50

TCP/SNA update

June 2003

In this issue

TCP/SNA Update

Published by

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

North American office

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

Subscriptions and back-issues

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

Editorial panel

Articles published in TCP/SNA Update are reviewed by our panel of experts. Members include John Bradley (UK), Carlson Colomb (Canada), Anura Gurugé (USA), Jon Pearkins (Canada), and Tod Yampel (USA).

Editor

Fiona Hewitt

Contributions

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

TCP/SNA Update on-line

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

Implementing an OSA-Express card using the QDIO mode

When migrating from a 9672 server to a zSeries server, system engineers, network administrators, and system programmers will have to plan the replacement of their OSA-2 (Open Systems Adapter-2) card by an OSA-Express feature.

The OSA-Express card provides significant enhancements over OSA-2 in function, connectivity, bandwidth, data throughput, network availability, reliability, and recovery.

Each OSA-Express card has one port for 9672 G5/G6 servers, and two ports for zSeries, which can be attached directly to a LAN or ATM network. Each port can be configured using HCD as one of the following channel types:

- *OSD (Queued Direct I/O)*. QDIO is a highly efficient data transfer mechanism that satisfies the increasing volume of TCP/IP applications and increasing bandwidth demands. SNA support is not native in QDIO mode, but is provided through the use of Enterprise Extender (a TCP/IP encapsulation technology that carries SNA traffic from an endpoint over an IP network).
- *OSE (non Queued Direct I/O)*. Non-QDIO channel, like any other channel-attached control unit and device, executes channel programs (CCW chains) and presents I/O interrupts to the issuing applications. Non-QDIO mode supports SNA/APPN/HPR and/or TCP/IP traffic simultaneously through the OSA-Express port.

This article reviews the benefits of using a QDIO channel, and describes how to install and use a QDIO channel for a TCP/IP connection.

QDIO IMPROVEMENTS

QDIO implements several major improvements:

introduced with the IRD Service Policy Server. It sorts outgoing IP message traffic according to the service policy you have set up for the specific priority assigned in the IP header.

LPAR-to-LPAR communication

Access to a port can be shared among the system images that are running in the logical partitions to which the OSA-Express channel path is defined to be shared.

When port sharing, the OSA-Express features working in QDIO mode can send and receive IP traffic between logical partitions (LPARs) without sending the IP packets out to the LAN and then back to the destination LPAR.

QDIO INSTALLATION

HCD definition

HCD operations are required to define the OSA-Express feature to the I/O hardware configuration.

In the following example, we define:

- One OSA QDIO CHPID: 3F
- One OSA control unit: 0CF0
- Sixteen OSA devices: 0F00-0F0F

CHPID definition

First, you need to define the OSA QDIO channel, as follows:

```
.-------------------- Add Channel Path Definition ---------------------.
|                                                                       |
|                                                                       |
| Specify or revise the following values.                              |
|                                                                       |
| Processor ID . . . : CPCC                                            |
| Configuration mode : LPAR                                            |
|                                                                       |
| Channel path ID  . . . . 3F   +                                      |
| Channel path type  . . . OSD  +                                      |
| Operation mode . . . . . SHR  +                                      |
| Managed  . . . . . . . . No (Yes or No)  I/O Cluster ________  +     |
| Description  . . . . . . ________________________________            |
|                                                                      |
| Specify the following values only if connected to a switch:          |
|                                                                      |
| Dynamic entry switch ID  __  + (ØØ - FF)                             |
| Entry switch ID  . . . . __  +                                       |
| Entry port . . . . . . . __  +                                       |
|                                                                      |
.----------------------------------------------------------------------.
```

The channel type is OSD when implementing QDIO (it is OSE for a non-QDIO CHPID).

Control unit definition

Once the CHPID has been defined, you need to define an OSA control unit to support QDIO devices, as follows:

```
.---------------------- add Control Unit Definition -------------------.
|                                                                      |
|                                                                      |
| Specify or revise the following values.                              |
|                                                                      |
| Control unit number  . : ØCFØ         Type . . . . . . : OSA         |
| Processor ID . . . . . : CPC                                         |
|                                                                      |
| Channel path IDs . . . . 3F   __   __   __   __    __    __    __    +
| Link address . . . . . . ____ ____ ____ ____ ____  ____  ____  ____  +
|                                                                      |
|                                                                      |
| Unit address . . . . . . ØØ   __   __   __   __    __    __    __    +
| Number of units  . . . . 255  ___  ___  ___  ___   ___   ___   ___   |
|                                                                      |
| Logical address  . . . . __  + (same as CUADD)                       |
|                                                                      |
| Protocol . . . . . . . . __  + (D, S or S4)                          |
| I/O concurrency level  . _   + (1, 2 or 3)                           |
|                                                                      |
.----------------------------------------------------------------------.
```

The control unit type must be OSA, the unit address must be set to 00, and the number of units must be 255.

Device definition

At this point, you need to define QDIO devices, as follows:

```
.-------------------------- Add Device ----------------------------.
|                                                                  |
|                                                                  |
| Specify or revise the following values.                          |
|                                                                  |
| Device number  . . . . . . . . ØFØØ  (ØØØØ - FFFF)               |
| Number of devices  . . . . . . 16__                              |
| Device type  . . . . . . . . . OSA          +                    |
|                                                                  |
| Serial number  . . . . . . . . __________                        |
| Description  . . . . . . . . . ________________________________  |
|                                                                  |
| Volume serial number . . . . . ______  (for DASD)                |
|                                                                  |
| Connected to CUs . . ØCFØ   ____  ____  ____  ____  ____  ____ ____  +
|                                                                  |
'------------------------------------------------------------------'
```

The device type must be OSA.

Note that any OSD CHPID requires at least three devices for each TCP/IP stack using it: one for read, one for write, and one for the datapath.

Missing Interrupt Handler (MIH) for QDIO

The OSA QDIO devices should have an MIH value of at least 15 seconds, or 30 seconds if running as a guest system on VM.

You should modify the IECIOS00 member in SYS1.PARMLIB to add the following line:

```
MIH TIME=ØØ:15,DEV=(ØFØØ-ØFØF)
```

QDIO USAGE BY COMMUNICATION SERVER

Once the OSA devices have been defined, they can be used to implement a TCP/IP interface:

- Three devices will be used: 0F00 for read, 0F01 for write, and 0F02 for the datapath.
- The IP address of the OSA interface is 192.168.142.18
- The VIPA address of the MVS system is 128.161.45.18

VTAM definitions

TCP/IP uses a VTAM interface to run the OSA-Express in QDIO mode.

A Transport Resource List (TRL) major node must be defined and activated before TCP/IP starts its QDIO device, as follows:

```
         VBUILD TYPE=TRL
MPCØ1OØØ TRLE  LNCTL=MPC,                                      *
               MPCLEVEL=QDIO,                                  *
               READ=ØFØØ,                                      *
               WRITE=ØFØ1,                                     *
               DATAPATH=ØFØ2,                                  *
               PORTNAME=OSAØØ          <- must match TCP/IP DEVICE name
```

The READ and the WRITE device numbers should be a even/odd pair. The READ device number must be the even number of the device pair. The WRITE device number must be the odd number of the device pair. The READ and the WRITE devices are used only to exchange control information.

This DATAPATH device is used for data transfer in both directions.

In a PRSM configuration running multiple LPARs sharing the same OSA card, one TRLE is needed in each LPAR, containing only one datapath address. The portname in the TRLE must be the same across the LPARs. Each TCP/IP DEVICE name must match the portname in the TRLE.

You can activate the TRL using the following VTAM command:

```
V NET,ACT,ID=MPCØ1OØØ
```

After activation of the TRL, the status of the TRLE is NEVAC or INACT until TCP/IP refers to it.

TCP/IP requires an active TRL before starting its device, so the VTAM major must be activated before starting TCP/IP.

TCP/IP definitions

In the TCP/IP profile, you need to code DEVICE and LINK definitions corresponding to the portname in the VTAM TRLE.

```
;
; OSA LINK - ETHERNET - PORT Ø - QDIO mode
;
DEVICE OSAØØ   MPCIPA  NONROUTER  <- Device name must match the portname
;                                    in the TRLE
LINK   LNKOSAØØ    IPAQGNET  OSAØØ
;
HOME
   128.161.45.18   VLINKØ
   192.168.142.18  LNKOSAØØ
;
```

The IPAQENET link type should be used for OSA-Express Fast Ethernet and Gigabit Ethernet features.

TCP/IP activation

At this point, you can start the TCP/IP stack and check the status of the VTAM and TCP/IP resources.

The following commands can be used to control the VTAM TRLE:

```
D NET,TRL
ISTØ97I DISPLAY ACCEPTED
IST35ØI DISPLAY TYPE = TRL 424
IST1314I TRLE = ISTTØPØF  STATUS = ACTIV       CONTROL = XCF
IST1314I TRLE = IUTLØØ8Ø  STATUS = ACTIV       CONTROL = TCP
IST1314I TRLE = IUTLØØ6Ø  STATUS = ACTIV       CONTROL = TCP
IST1314I TRLE = IUTIQDIO  STATUS = NEVAC       CONTROL = MPC
IST1314I TRLE = IUTSAMEH  STATUS = ACTIV       CONTROL = MPC
IST1314I TRLE = MPCØ1OØØ  STATUS = ACTIV       CONTROL = MPC
IST1314I TRLE = MPCØ1Ø2   STATUS = ACTIV       CONTROL = MPC
IST1454I 7 TRLE(S) DISPLAYED
IST314I END

D NET,TRL,TRLE=MPCØ1OØØ
ISTØ97I DISPLAY ACCEPTED
ISTØ75I NAME = MPCØ1OØØ, TYPE = TRLE 597
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
ISTØ87I TYPE = LEASED             , CONTROL = MPC , HPDT = YES
IST1715I MPCLEVEL = QDIO       MPCUSAGE = SHARE
IST1716I PORTNAME = OSAØØ      LINKNUM =   Ø   OSA CODE LEVEL = Ø326
IST1577I HEADER SIZE = 4Ø96 DATA SIZE = Ø STORAGE = ***NA***
IST1221I WRITE DEV = ØFØ1 STATUS = ACTIVE     STATE = ONLINE
IST1577I HEADER SIZE = 4Ø92 DATA SIZE = Ø STORAGE = ***NA***
IST1221I READ  DEV = ØFØØ STATUS = ACTIVE     STATE = ONLINE
IST1221I DATA  DEV = ØFØ2 STATUS = ACTIVE     STATE = N/A
```

```
IST1724I I/O TRACE = OFF  TRACE LENGTH = *NA*
IST1717I ULPID = TCPIP
IST1815I IQDIO ROUTING DISABLED
IST1757I PRIORITY1: UNCONGESTED PRIORITY2: UNCONGESTED
IST1757I PRIORITY3: UNCONGESTED PRIORITY4: UNCONGESTED
IST18Ø1I UNITS OF WORK FOR NCB AT ADDRESS X'ØC92DØ1Ø'
IST18Ø2I P1 CURRENT = Ø AVERAGE = Ø MAXIMUM = Ø
IST18Ø2I P2 CURRENT = Ø AVERAGE = Ø MAXIMUM = Ø
IST18Ø2I P3 CURRENT = Ø AVERAGE = Ø MAXIMUM = Ø
IST18Ø2I P4 CURRENT = Ø AVERAGE = 1 MAXIMUM = 3
IST314I END
```

The following command can be used to control the OSA TCP/IP inferface:

```
D TCPIP,,NETSTAT,DEV

EZZ276ØI DevName: OSAØØ            DevType: MPCIPA
EZZ2766I   DevStatus: Ready        CfgRouter: Non  ActRouter: Non
EZZ2761I   LnkName: LNKOSAØØ         LnkType: IPAQENET    LnkStatus:
Ready

EZZ2762I     NetNum:  Ø   QueSize: Ø   Speed: ØØØØØØØ1ØØ
EZZ282ØI     BytesIn: 21868552             BytesOut: 3789597
EZZ2764I     IpBroadcastCapability: No
EZZ2767I     ArpOffload: Yes  ArpOffloadInfo: Yes
EZZ2821I     ActMtu:  1492
EZZ2768I   BSD Routing Parameters:
EZZ2769I     MTU Size: ØØ576            Metric: Ø1
EZZ277ØI     DestAddr: Ø.Ø.Ø.Ø          SubnetMask: 255.255.255.Ø
EZZ281ØI   Multicast Specific:
EZZ2811I     Multicast Capability: Yes
EZZ2812I     Group              RefCnt
EZZ2813I     -----              ------
EZZ2814I     224.Ø.Ø.6          ØØØØØØØØØ1
EZZ2814I     224.Ø.Ø.5          ØØØØØØØØØ1
EZZ2814I     224.Ø.Ø.1          ØØØØØØØØØ1
```

Systems Programmer (France) © Xephon 2003

Mass disconnections from VTAM dumping?

We recently encountered a problem that seemed to be caused by VTAM dumps, and that created a lot of confusion and extra work. First, there was a DB2 abend, and then the Help Desk reported that there was no on-line mainframe response for over 10 minutes. To make matters even worse, after the 10 minute lapse, everyone found themselves disconnected. The Help Desk checked with Operations and was told that the only unusual console log messages involved three VTAM system dumps.

But surely three VTAM dumps can't really cause a 10 minute outage – this problem was addressed years ago by IBM. A system dump interrupts VTAM service only long enough for a memory-to-memory copy of the VTAM region; a parallel process is started that does the actual dump from memory to DASD without interrupting normal VTAM processing.

But if the VTAM dumps didn't cause the problem, what happened? Like most z/OS sites, we use a multi-session manager that allows users to be logged on to multiple applications simultaneously. This was getting only sporadic access to the processor, and, at one point, it got no CPU cycles for over two and a half minutes. This was because DB2 had a higher Dispatching Priority, and, before the abend, was busy failing, and then during the restart was attempting to back out all outstanding transactions. This was CPU-intensive stuff – especially considering that the abend was caused by a failure to recover from a previous failed attempt to recover from the memory and processor constraints caused by a single transaction's infinite loop in SQL.

For the record, we are a z/OS Version 1.2 environment, running DB2 6.1 and SOLVE:Access 4.1 (now called Unicenter SOLVE:Access Session Management) as a multi-session manager.

THE EVIDENCE

The console log records showing the VTAM dumps were as follows:

```
EPWØ4Ø1I EPWFFST: EVENT DETECTED BY VTAM FOR PROBEID ISTNACØ1 6Ø3
EPWØ4Ø6I DUMP DATASET IS: SYSTEM DUMP DATA SET
EPWØ4Ø2I PRIMARY SYMPTOM STRING FOR PROBEID ISTNACØ1 FOLLOWS:
EPWØ4Ø4I PIDS/569511701 LVLS/12Ø PCSS/ISTNACØ1 FLDS/ACDUNTNM
EPWØ4Ø4I PCSS/FULL RIDS/ISTNACTT PCSS/TIMEOUT PCSS/CLOSE FLDS/PTRLUCB
EPWØ4Ø4I ADRS/1798FØ98 VALU/CN11563
EPWØ4Ø2I SECONDARY SYMPTOM STRING FOR PROBEID ISTNACØ1 FOLLOWS:
EPWØ4Ø4I FLDS/LUCUSECT VALU/HØØØØØØØ1
EPWØ7Ø1I END OF MESSAGE GROUP
```

This set of messages was repeated three times in a row, with no intervening console records, at 09:27:59, 09:28:05, and 09:28:08. The three sets are identical except for the VALU/CN11563 field shown above for the first message set, with values of CN06262 and CN16742 for the second and third set of messages; all three values are VTAM terminal names.

The spooled messages from the VTAM Started Task (STC) for the period of the failure were as follows:

```
Ø9.24.Ø4 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT7 FOR ADNT7 522
   522   IST1Ø51I EVENT CODE = Ø2Ø2
   522   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ7ØØØ1Ø2Ø2ØF31ØØØØØØØ9
   522   IST314I END
Ø9.25.5Ø IST68ØI CONNECTION REQUEST DENIED – ID = ***NA***
                 PU GEN NOT SUPPORTED 542
   542   IST1354I DLUR NAME = ADNT.CP2216A       MAJNODE = ***NA***
   542   IST1394I CPNAME = ADNT.CP2216A      STATION ID = Ø2ØØ77A99ØØ
   542   IST314I END
Ø9.26.31 IEA794I SVC DUMP HAS CAPTURED: 544
   544   DUMPID=ØØ9 REQUESTED BY JOB (NET     )
   544   DUMP TITLE=NACCT –- CLOSE TIMER EXPIRED FOR N11563
   544
Ø9.26.54 IEA794I SVC DUMP HAS CAPTURED: 549
   549   DUMPID=Ø1Ø REQUESTED BY JOB (NET     )
   549   DUMP TITLE=NACCT –- CLOSE TIMER EXPIRED FOR NØ6262
   549
Ø9.27.Ø5 IEA794I SVC DUMP HAS CAPTURED: 567
   567   DUMPID=Ø11 REQUESTED BY JOB (NET     )
   567   DUMP TITLE=NACCT –- CLOSE TIMER EXPIRED FOR N16742
   567
Ø9.27.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT2 FOR ADNT2 594
```

```
   594   IST1Ø51I EVENT CODE = Ø2Ø2
   594   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ2ØØØ1Ø2Ø2ØF31ØØØØØØØ4
   594   IST314I END
Ø9.27.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT3 FOR ADNT3 595
   595   IST1Ø51I EVENT CODE = Ø2Ø2
   595   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ3ØØØ1Ø2Ø2ØF31ØØØØØØØ5
   595   IST314I END
Ø9.27.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT5 FOR ADNT5 596
   596   IST1Ø51I EVENT CODE = Ø2Ø2
   596   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ5ØØØ1Ø2Ø2ØF31ØØØØØØØ7
   596   IST314I END
Ø9.27.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT6 FOR ADNT6 597
   597   IST1Ø51I EVENT CODE = Ø2Ø2
   597   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ6ØØØ1Ø2Ø2ØF31ØØØØØØØ8
   597   IST314I END
Ø9.27.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT7 FOR ADNT7 598
   598   IST1Ø51I EVENT CODE = Ø2Ø2
   598   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ7ØØØ1Ø2Ø2ØF31ØØØØØØØ9
   598   IST314I END
Ø9.3Ø.42 IST53ØI AM GBIND PENDING FROM ADNT1 TO ADNT2 FOR ADNT2 77Ø
   77Ø   IST1Ø51I EVENT CODE = Ø2Ø2
   77Ø   IST1Ø62I EVENT ID = ØØØØØØØ1ØØØ1ØØØØØØØ2ØØØ1Ø2Ø2ØF31ØØØØØØØ4
   77Ø   IST314I END
```

Finally, the log kept by the multi-session manager software was as follows (note that, for readability, I’ve compressed the spacing, and removed the TERMINAL and USERID columns):

```
THU 2Ø-MAR-2ØØ3 2ØØ3.Ø79            SOLVE V4.1 SYSTEM ACTIVITY LOG
NMID NMOSP                                             PAGE    657
HH.MM.SS.TH TEXT                                           DD=LOG1
Ø9.25.3Ø.42 N2ØEØ8 LOCAL USER E97Ø574 FORCED OFF FROM NØ5Ø22
            BY TIME OUT.
Ø9.28.Ø2.37 N515Ø5 SESSION AIMS2 WITH APPLICATION PCICONE ENDED,
            USING ACB NØ6262B
Ø9.28.Ø7.84 NØ56Ø2 SESSION FOR N11563 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.11 NØ56Ø2 SESSION FOR N11563 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.65 NØ56Ø2 SESSION FOR NØ7Ø82 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.66 NØ56Ø2 SESSION FOR NØ6262 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.66 NØ56Ø2 SESSION FOR NØ6262 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.7Ø NØ56Ø2 SESSION FOR N16742 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.74 NØ56Ø2 SESSION FOR N16742 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.76 NØ56Ø2 SESSION FOR NØ84Ø5 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.77 NØ56Ø2 SESSION FOR NØ5295 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.8Ø NØ56Ø2 SESSION FOR N14262 FAILED. LOSTERM CODE X'14'.
Ø9.28.Ø8.8Ø NØ56Ø2 SESSION FOR NØ7Ø82 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.8Ø NØ56Ø2 SESSION FOR NØ84Ø5 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.86 NØ56Ø2 SESSION FOR NØ5295 FAILED. LOSTERM CODE X'18'.
Ø9.28.Ø8.87 NØ56Ø2 SESSION FOR N14262 FAILED. LOSTERM CODE X'18'.
```

```
Ø9.28.Ø8.9Ø NØ8HØ2 REGION <12Ø83> FOR E97Ø677 DISCONNECTED
Ø9.28.Ø8.9Ø NØ8HØ2 REGION <12Ø84> FOR E973271 DISCONNECTED
Ø9.28.Ø9.Ø1 NØ8HØ2 REGION <12Ø85> FOR E969ØØØ DISCONNECTED
Ø9.28.Ø9.11 N2ØEØ8 LOCAL USER E97Ø677 FORCED OFF FROM NØ7Ø82
            BY DISCONNECTION.
Ø9.28.Ø9.15 N515Ø5 SESSION AIMS WITH APPLICATION PCICONE ENDED,
            USING ACB NØ5Ø22A
Ø9.28.Ø9.15 NØ7ØØ3 CONNECT FAILED FOR TERMINAL N14645 –
            OPNDST FAILED RC=ØØ,FBK2=Ø9,SENSE=ØØØØØØØØ
Ø9.28.Ø9.15 N2ØEØ8 LOCAL USER E973271 FORCED OFF FROM NØ6262
            BY DISCONNECTION.
Ø9.28.Ø9.16 NØ55Ø1 CLSDST FAILED FOR N14645 –
            RC=14,FBK2=13,SENSE=ØØØØØØØØ
Ø9.28.Ø9.16 N2ØEØ8 LOCAL USER E969ØØØ FORCED OFF FROM N14262
            BY DISCONNECTION.
Ø9.28.Ø9.51 N227Ø5 LU N2Ø325 BEING PASSED TO APPLICATION PCICFOR
            USING LOGMODE SNX327Ø2
Ø9.28.Ø9.59 NØ84Ø1 NØØ952 CLOSED.
Ø9.28.Ø9.67 N2Ø2Ø2 REQUEST ACCEPTED – LU N2Ø325 PASSED TO
            APPLICATION PCICFOR
```

ANALYSIS

Beginning with the multi-session manager log, we analysed the problem by:

- Looking at the sequence of log records during the critical period, around the DB2 abend and three VTAM system dumps (excerpted above).
- Looking up the message codes in the software's manual, to understand what they meant.
- Looking through a 'normal' day of log records to determine how frequently some of these messages occur when things are fine.
- Trying to summarize what happened by writing short explanations for groups of consecutive messages.

The summary looked like this:

- The big gap (2:32 minutes) between the first and second message is 'normally'unheard of, but a 1:10 gap was seen the previous afternoon during the only other DB2 abend that had occurred so far this year.

- The N05602 messages show two LOSTERM codes, X'14' and X'18'. Both are a common occurrence, but never so many all at once; these all occurred within a period of 1.03 seconds.
- The N08H02 messages normally occur about once every half hour; we have three here within 0.11 seconds.
- Except for the last message (N20202), which is the start of a large number of messages (not shown) that indicate that normal operation has returned, the messages that follow the N08H02 series provide details of each of the three N08H02 disconnections.

The 2:32 gap was critical. Although the three VTAM dumps occurred during this period, the VTAM messages during the period indicated normal VTAM operations during much of this period. Add to this the fact that the DB2 region was using all the processor cycles it could get during this period. It began to look as though the multi-session manager was getting few or no processor cycles.

A quick look at SDSF DA (Display Active address spaces) later during normal operations indicated that CA-SOLVE:Access's Dispatching Priority was below that of the DB2 region that had abended during this period. That seemed strange, but one staff member said that it could make sense: after a DB2 abend, restart and recovery is the number one priority.

However, I still wondered whether a multi-session manager like CA-SOLVE:Access was designed to live without running for several minutes, and decided to contact the CA support centre, at

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

I clicked on Open a New Issue, and wrote:

> *"SOLVE currently has a lower dispatching priority than DB2. After DB2 abends, during DB2 recovery, SOLVE may not run for up to 2.5 minutes. As expected, users get no response during this period. But the issue is that when*

SOLVE does ‘come back’, it disconnects all users. The SOLVE log shows 14 N05602 errors and 3 N08H02 errors about 5-6 seconds after SOLVE regains control, but nothing else suspicious (to my eyes).

Preferred solutions: (1) a statement that SOLVE requires a higher dispatching priority; OR (2) a fix for this mass disconnections problem.”

Just over an hour later, I had an answer, despite having given this issue a low priority (of 3):

“I see your problem.

We have always said that SOLVE:Access should be in a dispatch priority just below VTAM but above normal applications.

You are correct in that I don’t see it documented in the manuals that way.

I will open a documentation request for you.”

Just over half an hour later, I received another note indicating that the documentation request had been opened and transferred, presumably to the people who write manuals at CA. On-line, between my original inquiry and the response I received, I could also see a note documenting the thought process of the support rep:

We have said that the dispatching priority for SOLVE:Access should be just lower than VTAM. The scenario that the customer lays out seems to justify this. In this case it appears that when DB2 abends because it has a higher dispatch priority than Access, terminal sessions are getting disconnected with a N05602 SESSION FOR luname FAILED. LOSTERM CODE probably a 14 or 18.

This seems logical since DB2 is getting to write the dumps, etc.

SOLVE:Access is being ignored until DB2 finishes.

SOLVE:Access has to appear to be a real terminal and therefore has to obey all the rules with regards to timeouts.

Checking the documentation to see if we have said that SOLVE:Access should have a High Priority, just below VTAM.

THE SOLUTION

The solution sounded simple: change the Dispatching Priority of the multi-session manager to just below VTAM. Unfortunately, however, a quick look at our z/OS WorkLoad Manager (WLM) showed that the definitions were unnecessarily complex because of excessive grouping. Worse still, the author of this mess was long gone, and everyone since had been afraid to touch it. A co-worker, also new to the organization, was already in the midst of a review of WLM because of erratic on-line response times and other symptoms. Both of us have been around long enough to know that you cannot just tweak one Dispatching Priority, or any other WLM setting, without a high risk of having something else going wrong. This meant that a change in Dispatching Priority for the multi-session manager would probably have to wait a few weeks until my co-worker's redesign of the WLM definitions had been completed.

The VTAM dumps also deserved attention, for three reasons:

- For customer service reasons, given the Help Desk as the customer and their original Trouble Ticket asking that the VTAM dumps be eliminated.
- On principle, since unnecessary VTAM system dumps are a waste of time, trouble, and resources.
- As part of a program that my co-worker and I wanted to start to eliminate 'all the noise' that made this DB2 abend problem so hard to diagnose – error messages that occurred regularly and in large quantities, but that no-one paid any attention to.

A colleague went straight to IBMLink's ServiceLink and found the answer within minutes. ServiceLink is available through IBM's VM-based customer network, and, in some parts of the world, through the Internet at http://www.ibmlink.ibm.com

On the Search for Service and Support Information Web page, I selected a Defined Search of General Search and hit the Go button. For the Search Argument field, I specified the PROBEID value shown on the EPW0401I message of the first console log record shown at the beginning of the article: ISTNAC01. When I hit the Search button, the results displayed were as follows:

```
Library                                             Number of Items
-------                                              --------------
MVS and associated products                                12
Program products                                           Ø
Problem diagnosis data                                     Ø
System Engineering Communication (SECOM)                   Ø
Flashes and bulletins from IBM                             Ø
Technical questions and answers                            Ø
Click on MVS and you will see the 12 items:
IIØ2613 CAN Ø2/14/Ø3 OPEN CLOSE ACB RECOMMENDED MAINTENANCE ACB APPL
                     IS HAVING AN OPEN OR CLOSE ACB PROBLEM WITH VTAM.
II13491     Ø2/1Ø/Ø3 ISTNACØ1 PROBE – FINDING THE PARTNER APPL.
II12578 CAN Ø1/2Ø/Ø3 INFO APAR FOR: DUMP TITLE = NACCT –- CLOSE TIMER
                     EXPIRED
OW44961 PER Ø8/Ø1/ØØ FFST PROBLEM ISTNACØ1 FOR CLOSE ACB TIMEOUT
OW44575 PER Ø8/Ø1/ØØ ISTNACØ1 PROBE DUMP TAKEN UNNECESSARILY FOR
                     FORCED CLOSE ACB FOR TELNET SESSION WITH TSO USER.
OW29338 PER Ø8/11/98 CLOSE ACB HANG. FMCB TSIP AND TSOP WAITING
                     ON EACH OTHER.
OW3Ø217 PER Ø3/Ø1/98 CLOSE ACB HANG. FMCB TSIP AND TSOP WAITING
                     ON EACH OTHER.
OW13826 PER Ø6/Ø3/97 AE APAR FIX FOR APAR OWØ9893
OW21ØØ1 PER Ø9/Ø2/96 ISTNACØ1 PROBE CLOSE ACB TIMEOUT FFST ALERT
OW192Ø2 PER Ø7/Ø1/96 CLOSE ACB TIMEOUT ISTNACØ1 ISTNACTT
OY66776 PER Ø7/19/95 PROBE ISTNACØ1 TRIPPED DUE TO A NON-ZERO LUCUSECT
OY65399 PER 12/22/93 PROBE ISTNACØ1 TRIPPED DUE TO A NON-ZERO LUCUSECT
```

Newer items are always the most interesting, since older items may have been solved in newer versions. This is especially true for this organization, since we are running a relatively recent release of z/OS (Version 1.2). The third item, II12578, was still only two months old when this problem occurred. Click on it and you will find that the January 2003 date is when it was last updated; the problem was originally closed in September 2001. You will also see the message field values:

```
Symptom         Symptom data  Explanation
--------------  ------------  -----------
PIDS/569511701  569511701     Component identifier
LVLS/411        411           Program product release level
PCSS/ISTNACØ1   ISTNACØ1      Software statement
FLDS/ACDUNTNM   ACDUNTNM      Data field name
PCSS/FULL       FULL          Software statement
RIDS/ISTNACTT   ISTNACTT      Routine identifier
PCSS/TIMEOUT    TIMEOUT       Software statement
PCSS/CLOSE      CLOSE         Software statement
FLDS/PTRLUCB    PTRLUCB       Data field name
ADRS/ØF4359AØ   ØF4359AØ      Storage address
VALU/CTAFØ1FØ9  TAFØ1FØ9      Error related character value
```

EXPLANATION AND SOLUTIONS

Even though a few values didn't match those in the EPW error messages shown at the beginning of this article, the important ones did, so this was clearly our problem. The explanation – at least, the busy system part of it – also matched our situation:

> *The above FFST probe (probe id ISTNAC01) is issued because a CLOSE ACB issued by the application identified in field VALU did not complete in a timely manner. This is almost always because it had a session with a same-domain application and that other application is not being dispatched in a timely manner. If it is known that the system was very busy at the time of the probe, the customer could choose to ignore the probe. Otherwise, the probe dump is needed to identify the partner application that is causing the delay.*
>
> *Please reference II13491 for info on how to find the partner application.*

Several solutions were offered. The most interesting ones were as follows:

> *Note that you can turn off this probe to prevent the dumps*

from being taken with the following FFST command:

```
MODIFY FFST,ACTION=DISABLE,PROBEID=ISTNAC01,VENDOR=IBM,AP=VTAM
```

Or the probe can be disabled using the FFST startup command list (DD name FFSTPARM, member name START00) with an entry:

```
P=ISTNAC01,V=IBM,A=DIS,AP=VTAM
```

If the probe is issued for an IBM TELNET application and the partner is known to be a same-domain TSO appl, then OW44575 will prevent this dump from being produced.

We chose the start-up command list option, and implemented it at the next IPL by putting the following job in the Hold queue:

```
//IR145    JOB (acct),'name',TYPRUN=HOLD,
//    MSGLEVEL=(1,1),MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID
//COPYMEM EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD  *
      REPRO IDS(FFST.V12ØESA.EPWPARM.IR145(STARTØØ)) -
            ODS(FFST.V12ØESA.EPWPARM(STARTØØ))
//*  WARNING:  IF MEMBER ALREADY EXISTS, IT WILL BE REPLACED
//*  WITHOUT ANY INDICATION.
//
```

Just before the next IPL, the operator was instructed to release the job with a $aj'ir145' command. However, a quick test showed that the command was risky: it will fail if multiple IR145 jobs exist in the JES*x* queues, even in the print hold queue from a previous test.

BACKOUT

The job simply replaces FFST's startup command list (START00) with one that includes the entry listed in ServiceLink. Because the START00 member didn't previously exist, the backout job is simple and foolproof:

```
//IR145B   JOB (acct),'name'
//    MSGLEVEL=(1,1),MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID
//DELMEM EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD  *
```

```
      DELETE FFST.V12ØESA.EPWPARM(STARTØØ)
   /* ADD THE IF AND SET MAXCC BELOW IF YOU NEED A ZERO COND  */
   /* CODE WHEN MEMBER DOES NOT EXIST:                        */
      IF LASTCC = 8 THEN -
         SET MAXCC=Ø
//
```

To avoid a second IPL, FFST is restarted after running the backout job by the following console commands and responses:

```
P EPWFFST.FFST
YES
S EPWFFST.FFST,SUB=MSTR
```

MULTIPLE COMPLEXITIES

The whole process got significantly more complicated when the decision was made to make the change to all four production z/OS systems (LPARs). Admittedly, all VTAM requests go through a single system, but there was a strong desire to keep all production systems in sync.

Five FFST parameter datasets were found, but only three were currently in use by the four z/OS systems. Three of the four systems share a single ICF catalogue, meaning that I had to create two FFST.V120ESA.EPWPARM.IR145 datasets and use two different job libraries to store these jobs.

Why three FFST.V120ESA.EPWPARM datasets when there are only two ICF catalogues? Because the dataset is catalogued without specifying a Volume Serial (VOLSER), reverting instead to the IPL volume.

In fact, that could have tripped me up, since the IPL volume was changed (by someone else) for two of the production systems during the same IPL. Since the IR145 job ran before the IPL, the change should have been applied to the dataset on the old IPL volume, not the new one. As it turned out, Operations had to IPL repeatedly and it all worked out.

But when I came in the next day, I also changed the two remaining FFST.V120ESA.EPWPARM datasets in preparation for future changes to IPL volumes. In retrospect, it would have made more sense to have IR145 and IR145B change all 5 FFST.V120ESA.EPWPARM datasets at the same time.

Jon E Pearkins
(Canada)

The standards enveloping Web services

Two years after the advent of the pivotal Web services' enabling technologies (WSDL and UDDI in September 2000, and SOAP a few months earlier), the World Wide Web Consortium (W3C) finally got round to proposing a formal definition for Web services. The W3C definition (not yet ratified) is contained with the Web Services Architecture specification that was put out in draft form on 14 November 2002.

The authors of this specification – from IBM, Software AG, Iona, and BEA – are acutely aware of the confusion about what actually constitutes a Web service. They therefore start off with the following caveat: "Although there are a number of varied and often seemingly inconsistent motivations for, and uses of, the term 'Web service', at its core, the following definition captures what we believe to be the shared essence of the term." They then define a Web service as follows:

> *"A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols."*

The term 'URI' in the above definition means a Uniform Resource Identifier – short strings that uniquely identify resources on the Web, with Uniform Resource Locators (URLs) of the form 'www.somename.com' being the best examples. 'Software systems' can be thought of as a broad, umbrella term that sets out to cover most things software-related, including applications, and possibly even small operating systems.

This definition clarifies and highlights two crucial aspects about Web services:

- They are innately XML-centric.
- They are software systems – ie functional software units – rather than just the underlying enabling protocols such as SOAP (as has been a mistaken belief in the past).

The inseparability from XML, combined with the reliance on technologies like SOAP, WSDL, and UDDI, goes on to confirm that Web services are only possible thanks to standards, in much the same way that it was universal conformance to a set of relatively simple standards that made the Web what it is today. Although it's sometimes easy to lose sight of the fact, Web services revolve entirely around XML – to the point that within some technical circles they are routinely referred to as 'XML Web services'.

Web services operate by interchanging data that is in the form of XML – or, to be even more precise, XML documents. The input parameter to a Web service therefore has to be in the form of an XML document. The output of a Web service will also always be a XML document. Furthermore, WSDL, the language used to describe the workings of a Web service in terms of what it expects as input, what it will return as output, and how you bind to it, is XML-based. SOAP, the preferred mechanism for invoking Web services and then exchanging the necessary XML documents back-and-forth with them, is also highly XML-oriented in order to ensure that it can be used with any programming language in a totally platform-independent manner.

The bottom line here is that it is the standards compliance aspect of Web services that makes them real and applicable, with the following four bedrock standards:

Standard or specification	Category	Purpose	Maintained by	Original vintage	Original sponsors
Core backbone standards					
XML	Description	Generalized data description scheme to facilitate data sharing	W3C	Feb 1998	Derived from SGML
SOAP	Messaging	Peer-to-peer exchange of messages using a remote procedure call mechanism	W3C	May 2000	IBM & Microsoft
WSDL	Description	XML derivative for describing Web-oriented communications protocols and messaging schemes	W3C	September 2000	IBM, Microsoft (& Ariba)
UDDI	Advertising/ publishing	Web-based registry mechanism to publicize and locate services	OASIS	September 2000	IBM, Microsoft & Ariba
Business process representation					
Business Process Execution Language for Web Services (BPEL4WS)	Business process	Notation for describing business process behaviour and interactions within the context of Web services		1st draft July 2002	IBM, BEA & Microsoft
Description					
Web Services Architecture	Description	Reference architecture that defines the key components and the relationships among them	W3C	1st draft November 2002	Software AG, IBM, Iona & BEA

Figure 1: Web services standards and specifications

Standard or specification	Category	Purpose	Maintained by	Original vintage	Original sponsors
Discovery					
Web Services Inspection Language (WS-Inspection)	Discovery	XML-based scheme that complements UDDI & WSDL, which allows a WS requestor to drill down into the services on offer by a provider		1st draft November 2001	IBM & Microsoft
Messaging					
Reliable HTTP (HTTPR)	Messaging	Guarantees reliable delivery of HTTP packets (which could include SOAP messages) between a server and client		June 2001	IBM
Web Services Attachments	Messaging	Extension to SOAP to facilitate attachments (eg images) without explicit XML encoding	IETF – draft RFC	1st draft June 2002	IBM & Microsoft
Direct Internet Message Encapsulation (DIME)	Messaging	Lightweight binary message format for encapsulating multiple payloads – and targeted to work with Web Services Attachments above	IETF – draft RFC	1st draft June 2002	IBM & Microsoft
Security					
XML Signature Syntax and Processing	Security	XML-based digital signatures	W3C	March 2001	Motorola, Citigroup & W3C

Figure 1: Web services standards and specifications (continued)

Standard or specification	**Category**	**Purpose**	**Maintained by**	**Original vintage**	**Original sponsors**
Security (continued)					
XML Key Management Specification (XKMS)	Security	XML-oriented scheme to integrate Public Key Infrastructure (PKI) with the Internet	W3C	March 2001	VeriSign, Microsoft & webMethods
Web Services Security (WS-Security)	Security	Enhancement to SOAP to provide message integrity, confidentiality & authentication		April 2002	Microsoft, IBM & VeriSign
WS-Security Addendum	Security	Clarifications to WS-Security (above) along with use of message timestamps, X.509 certificates & password transmittal		August 2002	Microsoft, IBM & VeriSign
WS-Security Profile for XML-based Tokens	Security	Framework to permit XML-based security tokens, eg Security Assertion Markup Language (SAML) & Extensible rights Markup Language (XrML), to be used with WS-Security (above)		August 2002	Microsoft, IBM & VeriSign
Web Services Trust Language (WS-Trust)	Security	Builds on top of WS-Security messaging to cater for security token exchange and multi-domain credential management		December 2002 VeriSign, & RSA	IBM, Microsoft,
Web Services Policy Framework (WS-Policy)	Security/ policy	Framework to describe and communicate the policies of a WS including service requirements, preferences & capabilities		December 2002 BEA & SAP	IBM, Microsoft,

Figure 1: Web services standards and specifications (continued)

Standard or specification	Category	Purpose	Maintained by	Original vintage	Original sponsors
Security (continued)					
Web Services Policy Assertions Language (WS-PolicyAssertions)	Security/ policy	Details general messaging-related assertions for use with WS-Policy (above) such as character encoding and preferred language		December 2002	IBM, Microsoft, BEA & SAP
Web Services Policy Attachments (WS-PolicyAttachments)	Security/ policy	Specifies three attachment mechanisms for using policy expressions with existing WSs		December 2002	IBM, Microsoft, BEA & SAP
Web Services Security Policy Language (WS-SecurityPolicy)	Security/ policy	Model and syntax for describing and communicating security policy assertions within the context of WS-Policy		December 2002	IBM, Microsoft, VeriSign, & RSA
Web Services Secure Conversation Language (WS-Secure Conversation)	Security/ policy	Builds on top of the WS-Security and WS-Policy models (above) to provide secure communications between services		December 2002	IBM, Microsoft, VeriSign, & RSA
Transaction processing					
Web Services Coordination (WS-Coordination)	Transaction	Extensible protocols for coordinating the actions of distributed applications especially in the context of completing a specific business process		August 2002	IBM, Microsoft, & BEA
Web Services Transaction (WS-Transaction)	Transaction	Works with WS-Coordination (above) to monitor the success or failure of short- or long-duration transactions		August 2002	IBM, Microsoft, & BEA

Figure 1: Web services standards and specifications (continued)

Standard or specification	Category	Purpose	Maintained by	Original vintage	Original sponsors
User interface					
Web Services for Remote Portlets (WSRP) (& Web Services for Interactive Applications (WSIA))	User interface	Standard set of user facing interfaces for facilitating the 'plug-and-play' integration of WSs with portals or interactive applications	OASIS	January 2002	IBM, Epicentric, & WebCollage
Web Services Experience Language (WSXL)	User interface	Component model for interactive WSs to facilitate commercial distribution via multiple channels and synthesis of new services		Late 2001	IBM

Figure 1: Web services standards and specifications (continued)

- XML – the basis for all input and output to/from a Web service.
- WSDL – the XML-based self-description of a Web service.
- SOAP – XML-oriented, peer-to-peer remote procedure call mechanism used by Web services.
- UDDI – the Web-based registry mechanism to publicize and locate services that can be used interactively or programmatically.

These four standards, however, are no longer the only ones associated with Web services. Over 20 new Web services-related specifications are now in existence, not even counting basic Web-related standards such as HTTP, URI, and TCP/IP that are used as utility protocols by the likes of SOAP. Figure 1 shows all of the standards and specifications associated with Web services as of spring 2003. It shows clearly that the four key enabling technologies are now just the cornerstones in a rising pantheon of Web services-related methodologies.

A RISING PANTHEON OF STANDARDS

This list of methodologies will, of course, continue to expand and evolve for a long time to come, with a group starting to talk about a brand new Web services management standard just as this article is being written. At present, there is no formal body that promises to maintain an accurate and up to-date list of these methodologies in an easy to reference, tabular basis. The problem is compounded by the inter-company (eg IBM versus BEA) or 'inter-faith' (ie Java versus .NET) politics that swirl around during the preliminary phases when such specifications are being formulated. The *de facto* standards bodies (eg W3C, OASIS, etc) understandably don't want to get embroiled in the early wrangling and wish to stay aloof and impartial until there's some consensus.

For example, in early summer 2002, IBM, Microsoft, and a consortium made up of BEA, Sun, and SAP were each vociferously promoting their own specific scheme for business process orchestration, with IBM's Web Services Flow Language (WSFL) being the most cited in the media as the front-runner for becoming the eventual standard. Nonetheless, in a few short weeks spanning July and August of that summer, IBM and Microsoft decided, quite suddenly, to bury their hatchets with regard to this issue, merge their competing schemes, and, moreover, collaborate, along with BEA, on creating the Business Process Execution Language for Web Services (BPEL4WS) – now one of the *de facto* standards in the still developing business process orchestration sphere.

Similar unpredictable dynamics are also occurring in the crucial and fast-growing Web services security arena – hence the difficulty in presenting an objective snapshot of the various standards in play without sometimes appearing to be backing the wrong horses.

Despite what we've just said about all the politics and rivalry, it's still illuminating to scan down the sixth column of Figure 1, to note how often IBM and Microsoft have collaborated in the Web services arena, despite the major differences of opinions they have on many other topics (including the best way to develop and deploy Web services). SOAP, WSDL, and UDDI, the prerequisite enabling technologies, were all joint productions involving IBM and Microsoft. This is also true of the nascent Web services architecture, the Web services inspection language, the messaging extensions, the transaction processing, and many of the security-related specifications.

All this said, IBM has been maintaining a very useful list of Web services specifications in the Web services section of its 'developerWorks' portal, which can be accessed by going to www.ibm.com and then selecting the 'developers' link that typically appears in one of the navigation bars. If you need an up-to-date list of Web services-related standards and specifications, this is the best place to start (always assuming, of course, that IBM continues to maintain and post this list). It's also worth consulting the following to see what these groups are portraying as the standards being adopted by the industry.:

- The Web Services Industry Portal at www.webservices.org
- OASIS at www.oasis-open.org
- The W3C at www.w3.org
- Possibly even the new, vendor-oriented Web Services Interoperability Organization at www.ws-i.org.

THE STANDARDS BEING DEVELOPED

Much of the new Web services-related standards work is focused on security, transaction processing, and user interface enablement. The latter is a particular hobbyhorse of IBM, as IBM is also a major provider of corporate portal-related technology. When Web services were originally conceived, they were meant to be purely programmatic – that is, they would be used program-to-program. In early 2002, IBM proposed a new standard referred to as Web Services for Remote Portals (WSRP), which advocates visual, user-facing Web services – that is, Web services that come complete with their own presentation services, ideally in the form of a contemporary graphical user interface (GUI).

WSRP's rationale, as implied by its name, is to make Web services easier to integrate into portals. Many functions desirable for inclusion within portals – for example, stock quote retriever, stock price ticker, weather report 'bug', search engine, local traffic report 'window' – will, in the future, be available in the form of Web services.

What WSRP and a related specification known as Web Services for Interactive Applications (WSIA), which is now being amalgamated into WSRP, contend is that having a built-in GUI will simplify and expedite the deployment of such services within portals, rather than portal developers having to locate and rely on additional software to implement an appropriate user interface front-end.

To be fair, there are certain portal-specific, as opposed to Web services-specific, methodologies that can also be used to tackle this user interface issue. Key among these are so-called 'XSL portlets' that will automatically render XML-defined content (and remember that the output of a Web service is indeed an XML document) within specific portal 'window panes', where a portlet is a piece of software that handles and controls a specific, autonomous 'window pane' within an overall portal view (or window, or even page).

It should be obvious that standards need to integrate Web services with transaction processing and business process 'representation'. The primary goal of Web services is to facilitate and expedite the development of the next round of e-business applications. These applications will reflect the latest e-business-related business processes being adopted by corporations.

Hence the emphasis on defining schemes whereby the invocation and execution of Web services can be orchestrated and managed to reflect the processes being addressed.

Security is the other ‘hot button’ when it comes to Web services. A rogue Web service could wreak havoc given that it is a remotely invoked piece of code – for example, a ‘Trojan Horse’ type Web service masquerading as a *bona fide* e-commerce utility could accept credit card information or passwords sent to it from an unsuspecting application. Hence the current efforts to provide mechanisms to validate and authenticate Web services to minimize the possibility of rogue Web services. Despite the growing list of security specifications, people are still worried about the potential security exposures of Web services. As with Web-to-host in the past, security has now become the major impediment slowing down the adoption rate of Web services.

THE BOTTOM LINE

Web services have brought out the best and the worst of today’s penchant for egalitarian open standards. There’s no question that open standards promote multi-vendor cooperation, foster competition as regards implementational options, and generally make sure that the consumer gets the best deal possible.

The democratic, open nature of the Web services-related specification also means that there’s no centralized leadership, control, master-plan, or business objectives. There are no ROI criteria. Talented developers employed by the large vendors are spending inordinate amounts of time and effort developing elaborate specifications without in any way being hampered by commercial pressures. That’s why, over the last year or so, we’ve seen more new standards and specifications rather than real, production-use Web services. Having said that, it’s true that the necessary standards, in particular those related to security and transaction processing, have to be in place before we can have meaningful Web services. Figure 1 gives you a baseline that you can use, going forward, to see what's happening when it comes to Web services-related standards.

Anura Gurugé
Strategic Consultant (USA)

Web services development options for the TCP/SNA world

For those of us who have been involved in nurturing TCP/SNA applications, the three vital questions that need to be answered when it comes to evaluating the options for developing Web services are as follows:

- What are the platform implications and, in particular, the role of mainframes, iSeries machines, and even Linux?
- What are the advantages of opting for the IBM solution and, in particular, the ever-expanding WebSphere repertoire, with its flagship WebSphere Application Server Version 5 and WebSphere Studio Application Developer (Integration Edition) offerings?
- What role do existing TCP/SNA applications have when it comes to creating new Web services? And what role will Web services play in the future for today's TCP/SNA applications?

The first two questions, and the WebSphere question in particular, also raise the issue of where Java and the Microsoft .NET initiatives fit in as regards Web services.

Fortunately, Web services' complete platform neutrality goes a long way towards helping us find the answers to these two vexing questions.

PLATFORM INDEPENDENCE LIKE NEVER BEFORE

Web services are truly platform-independent. What's more, this platform-independence applies to both sides of a Web services 'configuration' – that is, to the usage as well as the provision side.

A Web service is a run-time invoked external subroutine, running on a different machine, which functions by accepting input parameters in the form of an XML document and returning the required results also in the form of a XML document. Applications that use Web services can therefore run on any platform. Web services themselves are also totally platform-independent. They can run on any platform, whether mainframe or PC, and can also be developed on any platform with no regard to where or how they're going to be deployed. This means that there are no platform-related restrictions of any sort when it comes to any aspect of Web services.

It also means that there are no limitations as to which Web services can be used by which applications. For example, a Web application might be running on a Windows 2000 server that relies on three Web services: one running on a Sun Unix server, another on a Unisys mainframe, and the third on an IBM mainframe (or iSeries). A corollary scenario would be a new mainframe application that uses several Web services that are running variously on Linux partitions (possibly on the same mainframe as the 'calling application'), a Windows XP PC, and an Apple Mac Server. It would even be possible to have applications running on different platforms acting as Web service users and providers to each other at the same side – for example, a mainframe application using a Web service from a Windows 2000 server while at the same time providing a subroutine from the same application as a Web service to an application running on the Windows machine.

The platform-independence of Web services is, however, not in any way associated with or dependent upon Java, despite appearing to promulgate the same message popularized since the mid-1990s by the Sun-led Java camp. Since it's now one of the leading programming schemes of the Web era, Java inevitably has a major role with regard to Web services, particularly since the Web services initiatives of industry super-heavyweights such as IBM, H-P/Compaq, Oracle, BEA, and Sun are all highly Java-centric. Despite this, however, Web services are totally programming-language-agnostic – it would be quite possible for an application written in Visual Basic or C++ to use Web services functionality provided by a legacy program written in COBOL, PL/I, or even BAL. Needless to say, Java presents no problems, on either the usage or the supply side. Applications being developed in Java can use Web services provided by software written in any language, including Java, while Web services emanating from software written in Java can be used by applications written in any language.

Just as with platform-independence, the programming-language neutrality of Web services has no boundaries: it is truly any-to-any when it comes to programming language interoperability. It also provides a bridge between the old and the new when it comes to software programming methodology. This is another key reason why so many enterprises are interested in Web services: Web services provide a structured and regulated means by which mission-critical software functionality that was developed decades ago – before the advent of the PC, let alone the Web – can be 'brushed up' and profitably reused with brand new Web-centric applications.

Web services provide a means for 'modernizing' legacy applications, particularly those developed to run on mainframes and minicomputers (eg IBM AS/400s and HP3000s). With Web services, you can squeeze yet more ROI from these tried and trusted software workhorses, despite their having long-since earned their keep many times over. This legacy reusability aspect of Web services is an area of particular interest to IBM and other host access vendors (eg Jacada, SEAGULL Software), not to mention the Fortune 1000 companies that together have invested trillions of dollars in their application software portfolios.

Web services, in essence the new programming paradigm for the next iteration of Web evolution, has become the latest battlefield in the IBM versus Microsoft vendetta – even though, ironically, many of the pivotal Web services-related standards (eg SOAP, WSDL, UDDI, BPEL4WS, WS-Security, etc) are the result of open and close collaboration between IBM and Microsoft. Microsoft's Web services push revolves around .NET, and IBM's around the Java-based WebSphere Application Server and WebSphere Studio.

Microsoft .NET is made up of the following components:

- The .NET Framework Programming Model, which supports over 20 programming languages (including APL, FORTRAN, COBOL, Java, C++, C#, Perl, RPG, Pascal, and Visual Basic), enabling developers to create new Web-centric applications whether they be Web-based e-business applications, smart client programs (eg for palm and pocket devices), or XML Web services.
- A new set of developer tools, centred on Microsoft Visual Studio .NET 2003 (this is effectively the counterpart to WebSphere Studio).
- A repertoire of servers, including the nascent Windows Server 2003, Microsoft SQL Server, and the e-business-oriented Microsoft BizTalk Server, for integrating and executing new Web services and Web-based applications.
- The latest in client software, such as Windows XP, Windows CE, and Microsoft Office XP.

The one major downside to the .NET initiative is its lack of platform independence. While Web services developed by or running on a .NET server can be used with impunity by applications running on other platforms, including mainframes and iSeries, the .NET-based applications, Web services, and tools can run only on Windows systems – with the Windows servers in turn running only on ‘PCs’. Many, especially from the now increasingly egalitarian IBM camp, find this platform restriction galling.

The Java-based WebSphere offerings, in marked contrast, can and will run on multiple platforms – including Windows 2000/ 2003 servers.

Note, however, that platform independence isn’t necessarily that important. In some cases – possibly even most – it may not really matter where a Web service is running, provided that it meets the requisite reliability, resilience, security, and performance expectations. It makes sense not to get too religious about this issue. If you need to develop mainframe-specific or platform-independent applications, then you don’t really have any choice but to look at a Java-based approach. This might be WebSphere, but BEA, Sun, and Oracle (amongst others) also offer attractive Java-based solutions.

Java’s single biggest value proposition is platform-independence, but this isn’t necessarily the be-all and end-all. There’s a huge base of developers loyal to Windows, and this base is growing faster than the Java camp. What’s more, Web services’ platform-independence favours the Windows crowd – as long as they can claim that their services are mission-critical. Just as with Web hosting, a consumer of a Web service has no way of knowing the platform on which a Web service is being provided – just as you can’t tell whether a Web site is running on a Windows server, Unix server, or mainframe.

So the fact is that you really do have choices. Choose Java if your new applications have to be platform-independent – and even then you’ll have to choose between IBM, BEA, Sun, and so on. At the same time, you can think about .NET for Web services – even if those services are being specifically designed to be consumed by applications running on mainframes or iSeries machines.

SUPPLY SIDE	**SUPPLY SIDE**	**DEMAND SIDE**
New application developers	**Owners of previously developed applications**	**Enterprise class application consumers**
Obviate need to develop all necessary software.	Easily add new functionality to old applications through the use of third-party software modules.	Access to incisive, feature-rich applications.
Standardized, platform- and programming language-independent access to third-party software functionality.	Possibly remove platform dependencies of an old application by making the application a Web service that can then be invoked from a new platform-independent (eg Java) 'widget' (ie a micro application).	Faster availability of new applications, new features, and software upgrades.
Compress development schedules by being able to use third-party software functionality.		More reliable and resilient software.
Reduce software testing needs through the use of proven third-party software functionality.		Possibly more variety, options, and competition *vis-à-vis* different types of applications – and thus better pricing and service?
Reduce application development costs by minimizing development and testing requirements and schedules.		Availability of hitherto economically infeasible, specialized, 'niche' applications.
Expedite time-to-market.		
Offer specialized, additional functionality sourced from third parties.		
Enhance product competitiveness by offering value-added and best-of-breed functionality.		
Minimize lost opportunity costs caused by product delays, lack of functionality, or product instability.		

Ability to promote and market specific software functionality from existing, owned applications in the form of Web services.

Figure 1: Advantages made possible by Web Services

It's safe to say, without any fear of contradiction, that Web services transform everything to do with applications, whether they be TCP/SNA applications or new programs written to run on a Pocket PC. Web services positively impact everything associated with applications, on both the supply and demand sides. And this is key to all of us who look after TCP/SNA applications. Anybody and everybody who is exposed to software, in whatever role – whether it be as an application user, developer, financier, portal visitor, sales person, or sustainer – will enjoy some tangible benefits made possible by Web services. Web services go beyond just changing the paradigm when it comes to applications; Web services are truly iconoclastic!

Although you could think of Web services as being an application-development-specific methodology, this is far from being the whole picture. Web services also have a very productive role to play when it comes to existing applications, particularly the mission-critical TCP/SNA applications that sustain the operations of medium- to large-scale commercial, academic, and research enterprises. Web services enable problem solving logic from existing applications to be reused, and therefore offer a way to recycle software.

Web services are a standardized, universal, and sure-fire mechanism for isolating, capturing, and 'modularizing' software functionality so that it can be easily reused. In effect, therefore, they can make application owners into application software moguls! So it's not enough to say that Web services transform the application development process; they also completely transform application ownership. Figure 1 shows the main advantages that can be obtained by software providers and software owners through the availability of Web services.

In addition, the impact of Web services is not restricted to any one particular type or class of application. Although it's easy to assume that large enterprise class applications (eg ERP, CRM etc) are likely to be the main benefactors and thus consumers of Web services, this isn't necessarily the case. Down the road, most new applications, whether for single user desktop productivity or multi-user, pan-enterprise transaction processing, are likely to rely on one or more Web services. This will certainly be the case with Web-related applications, in particular e-business suites and portal-related applications.

SUMMARY

Although it may not be obvious at first, Web services are of particular interest and value to the TCP/SNA world. Web services provide a systematic way to yet again extend the life of these cash cow applications. Since Web services are truly platform-independent, it doesn't matter where the service is deployed; it can be on a mainframe, iSeries, or Windows 2003 server. The platform on which a Web service is running is of little consequence to the consumer of that service, provided that it meets performance, reliability, and security requirements. Finally, the TCP/SNA world has many options when it comes to Web services: you can be both a Web service provider and a Web service consumer, and you also have the option of using both Java and .NET offerings.

Anura Gurugé
Strategic Consultant (USA)

Printing management for TCP/SNA environments

Remote print management in the mainframe environment has always been expensive in both operational and maintenance terms. Most data centres use a combination of several products, printer emulator software, and various configurations to manage output distribution. There is usually also a variety of hardware involved, including:

- 3270 native printers
- Postscript, PCL, ASCII, and other printers
- Network LAN printer with different protocols (TCP/IP, decent, etc.).

While this variety of hardware justifies the presence of various software layers, it also complicates the management.

As recently as just a few years ago, this print management was concentrated almost exclusively with the SNA architecture. With the diffusion of IP protocols on mainframe platforms over the last few years, however, the management of remote print distribution has become more complicated. IP printer management is now needed, with products, software layers, and different modalities. And on top of the SNA concepts of type 1 logical unit (SNA Character String data) and type 0 or 3 logical unit (3270 data), we now need to cater for the Line Print Requester and Line Print Daemon characteristics of TCP/IP.

Although many data centres are managing print distribution in SNA and TCP/IP in parallel, there is a trend to migrate towards an IP-only solution.

This choice is motivated by the fact that IP protocols are now diffused on mainframe environments, and also because very few workstations (printers, display stations, and also network equipment) are still tied to the proprietary SNA architecture.

This article examines two different ways of managing remote print distribution.

CONFIGURATION 1

The first configuration uses the following components and subsystems:

- OS/390 Communication Server IP (TCP/IP IBM)
- OS/390 Print Interface
- OS/390 NetSpool
- OS/390 IP PrintWay
- OS/390 JES2 or JES3.

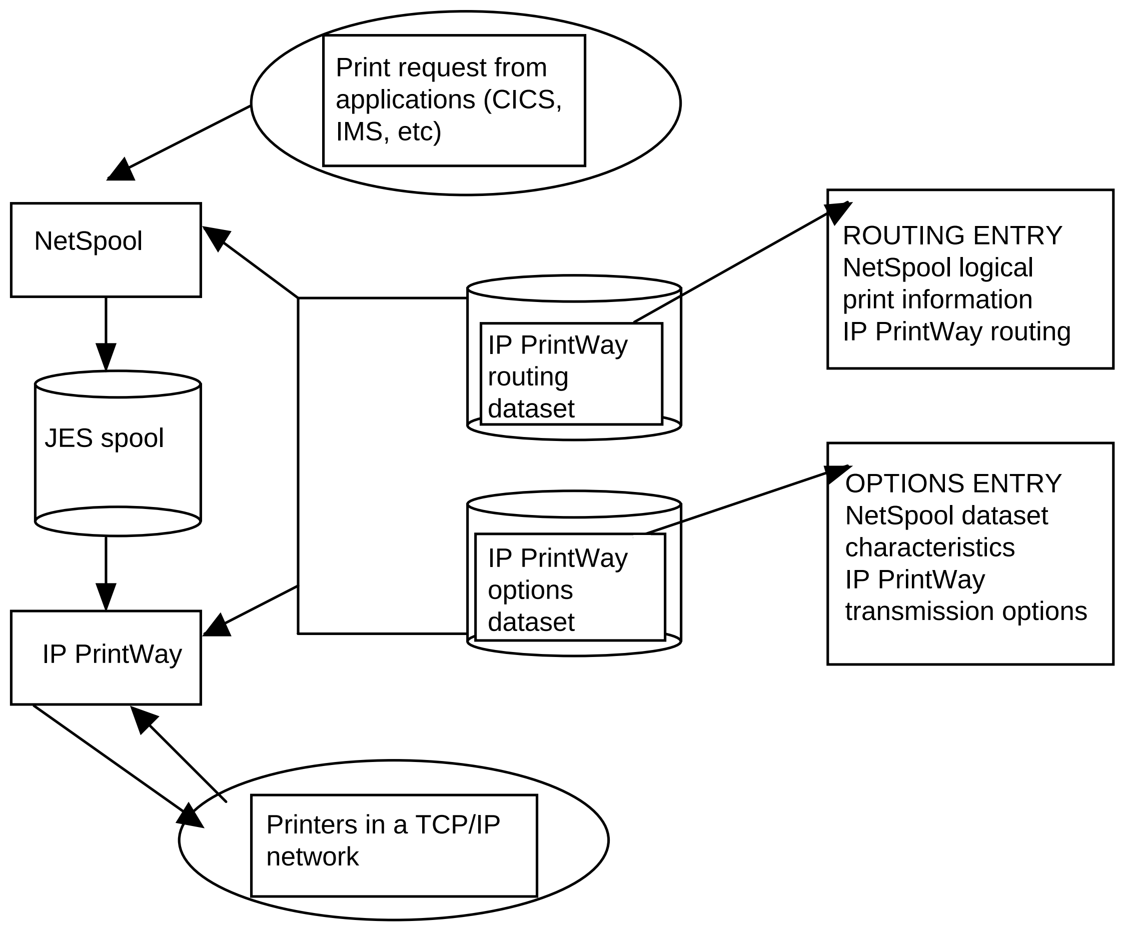
OS/390 Print Interface

The Print Interface is a component of the Print Server feature of the OS/390 system. It creates output datasets on the JES2 or JES3 spool for print requests received from remote workstations in your TCP/IP environment and from OS/390 Unix System Services.

IP PrintWay can then be used to transmit the output datasets created on the JES spool by the OS/390 Print Interface to printers in your TCP/IP network. In order to do this, you need to specify the printer routing information. This can be done either in the PrintWay routing dataset or in the OS/390 Print Interface Printer Inventory. The PrintWay routing information dataset can also be used for the output dataset created by other components (see the NetSpool feature, below).

OS/390 NetSpool

NetSpool creates output datasets on the JES2 or JES3 spool from VTAM print requests (print CICS, IMS, etc).



Print request from applications (CICS, IMS, etc)
NetSpool
ROUTING ENTRY
NetSpool logical print information
IP PrintWay routing
IP PrintWay routing dataset
JES spool
OPTIONS ENTRY
NetSpool dataset characteristics
IP PrintWay transmission options
IP PrintWay options dataset
IP PrintWay
Printers in a TCP/IP network

Figure 1: First configuration, with NetSpool and PrintWay

You can use PrintWay to transmit the output datasets that NetSpool creates on the JES spool to printers in your TCP/IP network. When you do this, you can define NetSpool logical printers in the PrintWay routing and options datasets, because all the information required by both NetSpool and PrintWay is in the same location. This makes maintenance easier. NetSpool and PrintWay can then share information in the PrintWay routing and options datasets (see Figure 1).

As the figure shows, NetSpool and PrintWay perform complementary functions. NetSpool converts print requests from VTAM applications into System/390 line data, and creates output datasets on the JES spool, using information in a NetSpool print-characteristics dataset.

NetSpool can be configured for your installation so that you don’t need to change existing VTAM applications. Existing VTAM applications will then be able to send print requests to NetSpool in the same way as they currently send print requests to SNA-network printers.

NetSpool, running as a VTAM application on the same or different OS/390 operating systems, simulates SNA network printers as logical printers, with each logical printer being defined to VTAM as an application logical unit. After NetSpool has created an output dataset on the JES spool, PrintWay can transmit it to another location for printing in a TCP/IP network.

Some of the benefits of NetSpool are as follows:

- *Data integrity*. By placing VTAM application output on the JES spool, NetSpool enables you to use JES security, checkpoint/restart, and reprint capabilities.

- *Routing flexibility*. JES selection parameters (ie destination, class, forms, etc) can be specified for routing NetSpool output to any supported printer or location.

- *AFP formatting flexibility*. Advanced Function Presentation parameters (page definition, form definition, etc) can be specified when printing on PSF-controlled printers.

- *Printer sharing*. NetSpool allows multiple VTAM applications to simultaneously direct output to the JES spool for printing on a single shared printer.

- *Print broadcasting*. NetSpool allows a single VTAM application to simultaneously direct output to the JES spool for printing on multiple printers.

- *Multiple instances of NetSpool*. NetSpool can run simultaneously in separate address spaces, with each instance of NetSpool receiving print requests sent to different logical printers.

- *Dynamic allocation of the OS/390 datasets on the JES spool*.
- *Operator commands and diagnostic messages to manage remote printers*.

OS/390 IP PrintWay

Acting as a Line Printer Requester (LPR), IP PrintWay can translate output datasets from EBCDIC to ASCII and transmit the datasets to any Line Printer Daemon (LPD) in a TCP/IP network. The LPD must adhere to the TCP/IP protocol between LPRs and LPDs as defined and specified in RFC 1179 and amendments.

PrintWay can transmit data to LPDs running on a printer or on a host system, including LPDs running on MVS, AIX, OS/2, OS/3 9 0 , OS/400, Windows, or Unix systems. If supported by the printer, IP PrintWay can also transmit the datasets to a direct socket printing connection and bypass the overhead of LPR/LPD processing.

PrintWay runs in its own address space and operates as an extension of JES. One PrintWay functional subsystem application (FSA) can support multiple printers, but multiple PrintWay FSAs could also be used.

The main functions of the output writer are as follows:

- Selecting datasets from the JES spool.
- Transmitting datasets to an LPD or another direct socket connection.
- Deleting the datasets from the JES spool.

The output datasets created on the JES2 or JES3 spool may contain System/390 (System/370) line data or formatted data such as PCL, PostScript, or MO:DCA-P data.

PrintWay offers the following advantages:

- *Distribution of data for use with workstation applications*. Transmitting datasets to a print queue on a workstation allows you to use applications running on your workstation not just to print data but to perform other functions as well (eg to view or archive).

- *Routing flexibility*. OS/390 job submitters can specify routing information in JCL – including, for example, the name of the target print queue or port number, and the IP address or name of the printer's host system. It's also possible to specify routing information for each target printer in a PrintWay routing dataset.

- *Ease of specifying options*. PrintWay supports the existing translation and formatting options of the MVS LPR command and also some additional options that affect the transmission of datasets. These options can be defined in a PrintWay options dataset. A job submitter can request a set of options in JCL, or this set of options can associate a set of options with each printer in the routing dataset.

- *Data integrity*. PrintWay can retry the transmission of datasets and can also verify that the data has been successfully transmitted before deleting datasets from the JES spool. PrintWay can also retain datasets on the JES spool for a specified amount of time after either successful or failed transmissions.

- *Simple integration with NetSpool*.

- *EBCDIC to ASCII translation*. PrintWay can translate single-byte and double-byte datastreams from EBCDIC to ASCII, using either standard translation tables provided by TCP/IP or user translation tables.

IP PrintWay is generally composed of the following datasets:

- *The routing dataset*. This is a VSAM file containing a routing entry for each target print queue or port. This file can be shared with OS/390 NetSpool.

- *The options dataset*. This is a VSAM file containing sets of

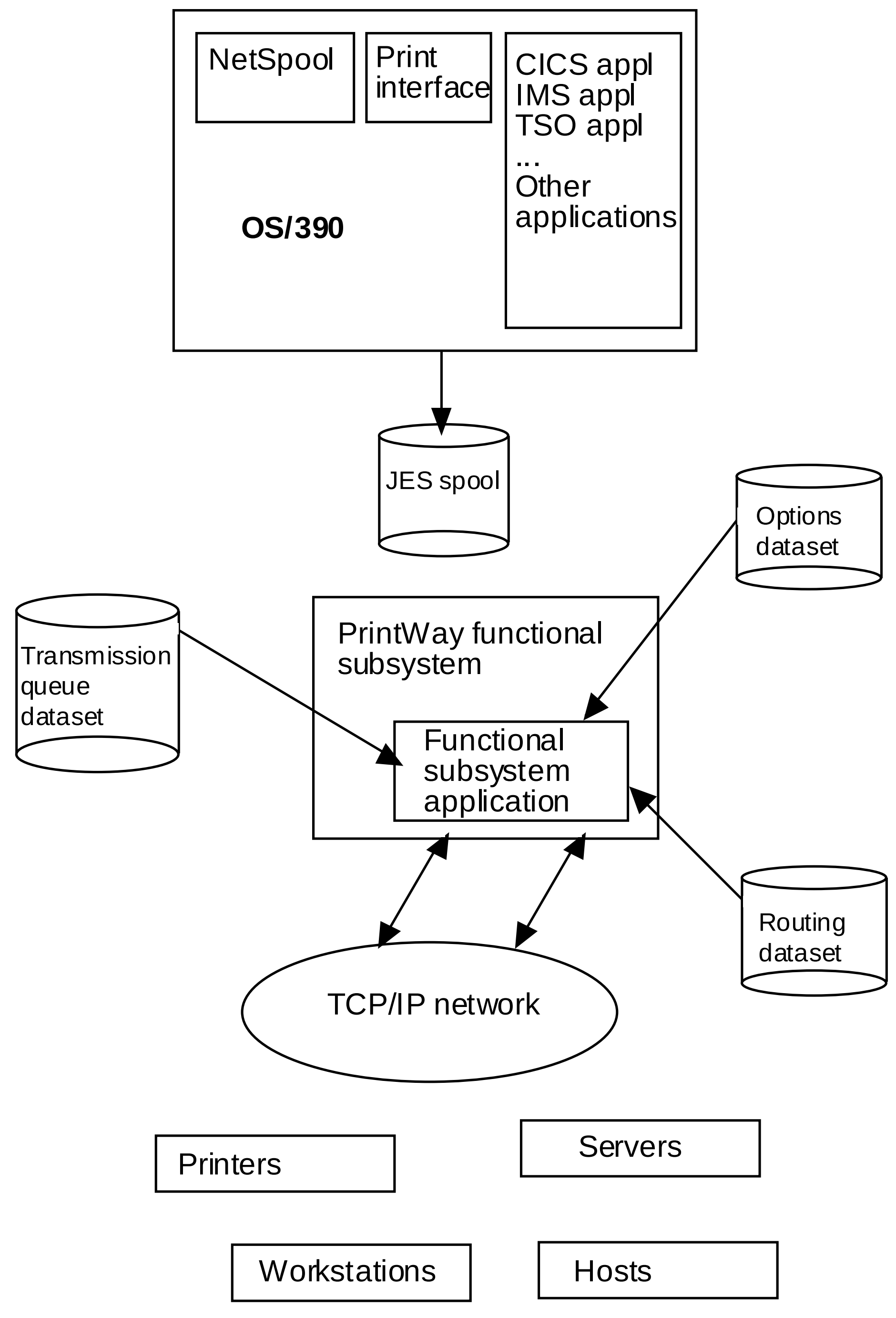


Figure 2: The components involved in remote print distribution

options called options entries. This file can be shared with OS/390 NetSpool.

- *The queue dataset*. This is a VSAM file containing an entry for each dataset on the JES spool being processed by the

PrintWay FSAs within one or more FSSs. A transmission-queue entry indicates the status of the transmission of the dataset and contains routing information and transmission options.

- *The message-log dataset*. This is a sequential file containing messages that track dataset transmissions.

Figure 2 shows the components involved in remote print distribution. What happens is as follows:

1 Using NetSpool or OS/390 Print Interface, a job (batch or on-line application) running on the OS/390 system creates output datasets on the JES2 or JES3 spool.

2 On the OS/390 system, PrintWay runs as an output writer (FSA) on JES2 or JES3. The PrintWay FSA selects output datasets from the JES spool, according to work selection criteria defined to JES2 or JES3. The work selection criteria correspond to JCL parameters, such as output class, specified in the JCL for each dataset.

3 PrintWay routes each dataset to an LPD or a direct socket printing connection, using routing information specified either in the JCL or in a PrintWay routing dataset.

4 PrintWay uses options specified in the PrintWay options dataset, or default options, when transmitting datasets.

5 PrintWay maintains a transmission queue to keep track of datasets being processed. This queue contains the status of each transmission, together with routing and option information.

6 PrintWay transmits datasets to the target print queue using the LPR/LPD protocol, or to a port using the direct socket printing protocol. PrintWay also transmits printing options, including the number of copies to print as specified in JCL, information for printing a separator page, and so on.

7 PrintWay retries unsuccessful transmissions the number of times requested either in the JCL or in the routing entry.

After successfully transmitting each dataset, or after completing the requested number of transmission attempts, PrintWay either deletes the dataset from the JES spool or retains the dataset on the JES spool for the period of time specified in the JCL or routing entry.

IP PrintWay provides an ISPF interface to create and manage the routing and options datasets, enabling transmission status to be monitored, datasets to be rerouted to other print queues or ports, and transmission options to be changed.

The administrator can also use a macro to create a batch job that adds, modifies, and deletes entries.

IP PrintWay notifies the job submitter, issuing messages to a PrintWay message-log file, when a dataset is successfully transmitted, or if the transmission fails, or when the dataset is deleted from the JES spool. Messages requiring operator action are issued to the operator console.

OS/390 JES2 or JES3

A functional subsystem must be defined for PrintWay in a JES environment, coding the FSS macros in the JES2 initialization parameters, or FSSDEF macros in the JES3 initialization parameters. You also need to define one PRT*nnnn* statement for JES2, or one DEVICE statement for JES3, for each functional subsystem application (FSA) under control of the FSS.

You then need to define the work selection criteria for each PrintWay FSA during JES initialization. These criteria determine which output datasets each FSA selects from the JES spool. Specify the work selection criteria – including, for example, output class, form name, and destination name – on the WS parameter of either the JES2 PRT*nnnn* statement or the JES3 DEVICE statement.

OS/390 Communication Server IP

IP PrintWay uses only the TCPIPJOBNAME and DATASETPREFIX parameters of TCPIP.DATA dataset that contains TCP/IP configuration statements used by TCP/IP applications.

The TCPIPJOBNAME parameter specifies the name of the TCP/IP program in your system. If you have several TCP/IP programs installed with different names, you can specify the name of the TCP/IP program that PrintWay is to use in the EXEC statement of the PrintWay start-up procedure.

The DATASETPREFIX parameter specifies the high-level qualifier for TCP/IP datasets. PrintWay uses this high-level qualifier when searching for TCP/IP translation tables.

Note that you should pay special attention to the parameters relating to the different types of TCP/IP buffer, namely:

- DATABUFFERPOOLSIZE
- SMALLDATABUFFERPOOLSIZE
- TINYDATABUFFERPOOLSIZE

You should also pay special attention to the parameters that specify the interval between keep-alive probes (KEEPALIVEOPTIONS).

CONFIGURATION 2

Configuration 2 uses the following products and subsystems:

- OS/390 TCP/Access (TCP/IP Computer Associates)
- OS/390 Enterprise Print Services (Computer Associates product)
- OS/390 JES2 or JES3.

OS/390 Enterprise Print Services

Enterprise Print Services (EPS) provides printer access across the network. It manages the print requests independently from where they arrive; in fact, OS/390 users can send data to printers attached to other systems on the TCP/IP network, and users on other systems on the TCP/IP network can send data to printers controlled by the OS/390 Job Entry Subsystem (JES subsystems).

EPS contains both client and server protocols compatible with the Unix remote print functions, Line Printer Daemon (LPD) and Line Printer Remote (LPR) protocol.

Enterprise Print Services can manage print services in a variety of environments and can also be defined to JES as an external writer using the SubSystem Interface (SSI).

For data sent by JES via the SSI, EPS maps the user-specified JES destinations into specific printers (queues) attached to systems on the TCP/IP network. For data received from TCP/IP hosts, EPS maps the print queue to JES destinations transmitted to JES via sysout dynamic allocation.

The Unix-like TCP/Access sockets facility provides the interface between EPS and the TCP/IP network, and the LPR protocol provides the interface to the remote printers on the network. The EPS LPD server uses the same interfaces, but in the opposite direction.

OS/390 users can send print jobs to EPS destinations known to JES with JCL, JES commands, or SSI commands.

EPS receives the print files from JES or NJE and transmits them in LPD format to the associated remote printers on the TCP/IP network. This is done by the LPD client of EPS.

When a user on a remote system sends print data to EPS, the printer designated in the LPR command is mapped to a JES destination. This can be a local OS/390 printer, a remote printer on the TCP/IP network, or a printer controlled by another JES node (NJE node).

Enterprise Print Services also provides VTAM printer emulation. The interface between an application program and EPS is provided by VTAM and the SNA virtual printer facility, either LU type 1 or LU type 3. EPS maps VTAM-defined printers to printers attached to remote systems defined on the TCP/IP network. The EPS virtual printer lets OS/390 applications using SNA printer requests access LPD printers on the TCP/IP network; existing VTAM applications therefore send print requests to EPS in the same way as they currently send print requests to SNA-network printers.

The main features of EPS are as follows:

- It implements the protocol used by LPR/LPD on Unix and other platforms using TCP/IP to provide printing compatibility with a wide variety of systems.
- It provides LPD client and server functions.
- It includes SNA printer emulation, letting OS/390 applications send print data through VTAM to a virtual SNA printer that is mapped to a remote printer.
- It uses Network Job Entry to interface with JES, allowing the flexible use of JES DESTIDs for both client and server functions.
- It uses SSI to interface with JES to receive data and map the DESTID to a remote printer.
- It uses sysout dynamic allocation with JES to send data from a remote printer.
- It supports various JES output parameters via the EPS LPD server for inbound and outbound print files.
- It supports the specification of ASCII and EBCDIC translation tables.
- It preserves server and client print requests across system outages and component failures.
- It provides logs of activity and error conditions.
- It provides operator commands to control the status and activity of the printers.

EPS is made up of a number of components. First, there are inprocessors, which are interfaces to specific types of input source. They deal with the unique transport, session, and application layers of the input source, and the creation of jobs from that input.

There are four types of inprocessor:

- LPD servers, which manage LPR client traffic. This requires the availability of a local TCP/IP stack and can be configured to control access by remote host, user, and/or printer.
- NJE Network Job Entry nodes, which manage incoming NJE traffic. The amount of the NJE job and dataset attributes provided during an NJE session are preserved in the resulting EPS job.
- VPRT SNA Virtual Printers, which manage incoming LU1 and LU3 traffic.
- SSI External writers, which use the JES subsystem interface to provide sysout datasets as input to EPS; sysout can be selected by any combination of SSI options (CLASS, DEST, WRITER, FORM). This interface provides less JES job and dataset information than the NJE inprocessor and results in a leaner EPS job.

Inprocessors place jobs on a queue, and the queue then provides the job and destination-specific information to a named outprocessor. The inprocessor selects the queue for a given job as follows:

- For LPD, remote printer specification provides the queue name.
- For NJE, the DEST contained in the NJE dataset header provides the queue name.
- For VPRT, the LU table provides queueing instructions.
- For SSI, the DEST associated with the sysout dataset provides the queue name.

Outprocessors are interfaces to specific types of output destination. They deliver a job to its intended destination and deal with the unique transport, session, and application layers of that destination. Outprocessors also perform protocol conversion and data translation.

There are three types of outprocessor:

- *LPR*. An outprocessor uses the LPR protocol to interface with remote LPD servers. This requires the availability of a local TCP/IP stack.
- *NJE*. A node generates outgoing NJE traffic.
- *JES*. A JES outprocessor uses dynamic allocation to create sysout datasets.

The remaining EPS components are as follows:

- The log task, which handles EPS logging.
- The spool, which is the repository for jobs.
- Jobs are the means by which inprocessors store the input they've received for subsequent delivery by an outprocessor. Jobs consist of attributes and data. Each job is assigned a unique identifier.

To perform print processing in conjunction with remote systems on the TCP/IP network, EPS implements the LPR protocol for client (outprocessor) and server (inprocessor) functions. The LPR protocol is a network application protocol for transferring print files between machines based on a client/server model.

Each LPD print request sent on the TCP connection consists of one or more data files and a control file. The data files contain input, which is printed. LPD data files contain a representation of the data to be printed as a typical text file sent to or received from a remote host (lines of ASCII text delimited by ASCII printer control characters).

The control file contains various commands, processing options, and information used to print the data files. Control files are sent to, and received from, TCP as ASCII files.

The LPD client of a host connects to a local TCP port number usually in the range of 721 to 731. It then connects to a remote host's LPD server, usually at well-known LPD server port 515.

The EPS LPD Inprocessor Server and EPS LPD Outprocessor Client function as follows:

1 The LPD inprocessor (server) implements the LPR protocol for receiving inbound print requests along with its control and data files. A job is created by the inprocessor and the job is then placed on a queue with the same name as the requested printer from the remote host.

2 A server process begins by starting the LPD inprocessor, which uses TCP sockets to listen for a remote connection request and to establish the requested connection.

3 The EPS LPD server requests data from the remote host and sends acknowledgements according to the LPR protocol. The data received by the LPD server is processed into a single OS/390 dataset called a spool file, which contains the untranslated spool file contents of the data files.

4 When the remote LPD client has sent all its files and all acknowledgements have been exchanged, it closes the connection. The event is received by the EPS LPD server. The EPS LPD server then begins its session end procedures and the connection is closed. The job is then placed in the queue, ready for processing by an outprocessor. When the session ends, the EPS LPD server generates a message to the log and performs its final clean-up for the session.

5 The requested printer queue name, the name of the data files, and the contents of the control file are saved in the checkpoint dataset.

6 The LPD outprocessor (client) implements the LPR protocol for sending outbound print requests to LPD servers on remote hosts. The EPS LPD client obtains its input for print requests from any of the possible inprocessors (NJE, JES,

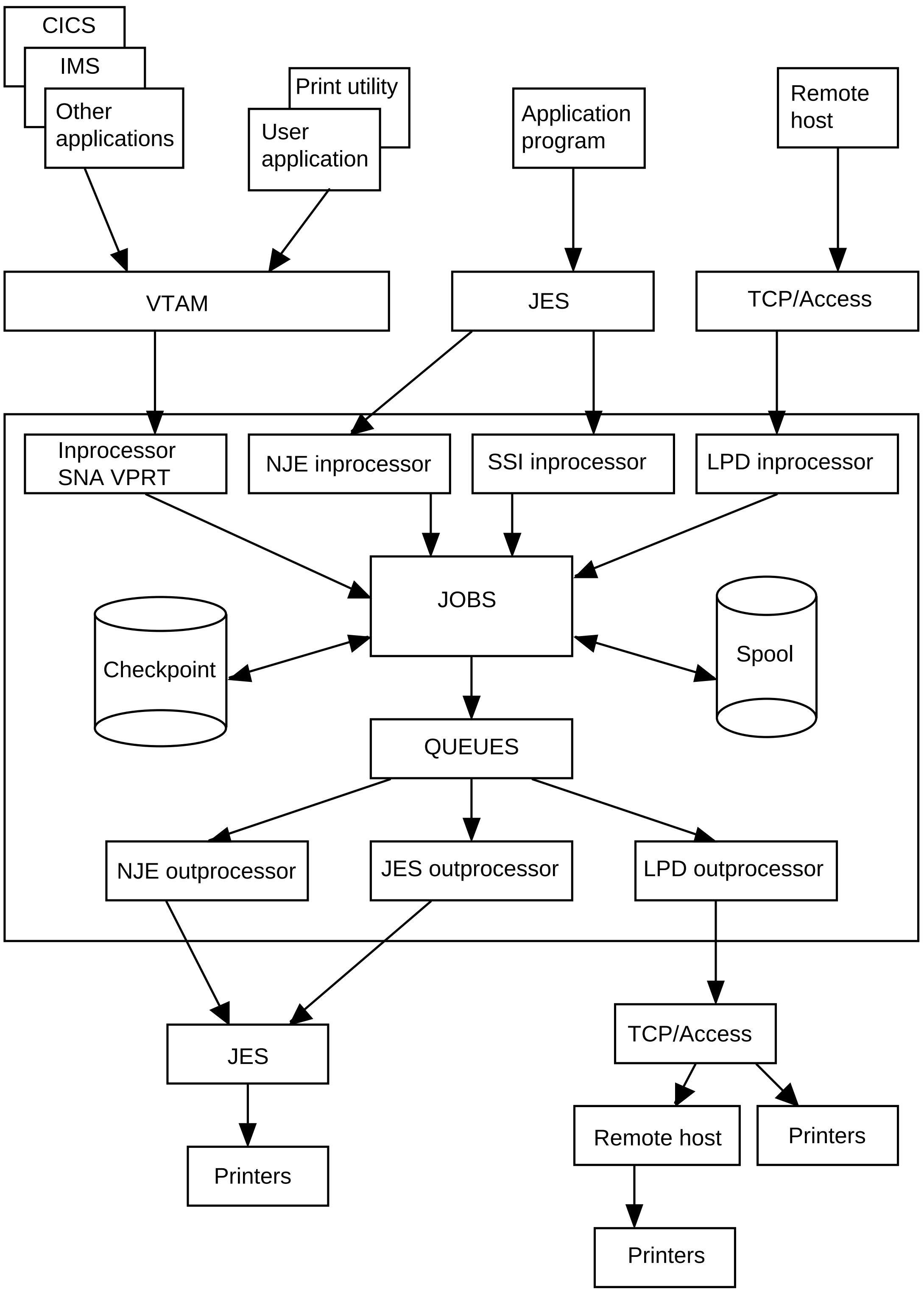


Figure 3: Second configuration

VPRT). The EPS LPD client passes the results of its processing to the remote LPD server using TCP sockets.

7 Various statements in the EPS configuration file (QUEUE statements) define destinations supported by the EPS client and the remote printer queue to which each is mapped.

8 The EPS LPD client begins when it receives notification from the queue dispatcher that a job is ready for sending.

9 The EPS LPD client determines the remote host, remote printer, remote port, and local port based on the queue settings.

10 The LPD client allocates a local port and establishes a connection with the remote host's LPD server using TCP sockets. The remote host address and the remote port number for the LPD server are derived from the configuration file (REMOTEHOST and REMOTEPORT parameters of the SET QUEUE command statement).

11 The print request with remote printer name is sent to the remote LPD server. The EPS LPD client waits for an acknowledgement from the remote server. On receipt, the client begins sending the LPD data files contained in the spool file followed by the control file (contained in the job), according to the LPR protocol.

12 When all of the data has been sent and everything is properly acknowledged, the EPS LPD client closes the TCP connection. The EPS LPD client then waits for the next print request.

The EPS inprocessor and outprocessor provide the interface to the TCP/IP network. They both use TCP sockets to do the following:

- Establish and terminate TCP connections
- Send and receive LPR protocol data
- Perform host name resolution

- Notify the operator of various TCP related events.

The inprocessor LPD server starts up as a separate OS/390 subtask and performs a listen socket function on the port specified by the EPS parameter. When a remote host has a print job to send, the listen function completes and the inprocessor attaches a worker subtask, passing the socket connection to the worker subtask. The inprocessor then listens. Meanwhile, the worker task receives the data from the remote host, writes the data to the EPS spool, and creates an EPS job.

The outprocessor LPD client starts up as a separate OS/390 subtask and waits for jobs to be dispatched to it by the EPS main task. When a job is dispatched to the outprocessor, it attaches a worker subtask. The worker subtask uses the information from the queue to determine which remote host to connect with. Once connected, it transmits the job in LPD format to the remote host. Data conversion and translation are done before transmitting to the remote host.

OS/390 JES2 or JES3

JES configuration statements need at a minimum to define the NJE node to be used by EPS and its connection to the NJE network, a JES APPL, NODE, and CONNECT statements.

The second configuration is shown in Figure 3.

Espedito Morvillo
(Italy)

Terminal emulation – 20 years of transformation

Mainframes have a long history in the corporate enterprise, and continue to play a critical role today. Over the past 20 years, access to mainframes has changed and improved significantly, adding to their continued value. Terminal emulation began with the advent of personal computers in the early 1980s, and has drawn on many technological innovations to make mainframe access quick, easy, diverse, and – most importantly – reliable.

The transformations in terminal emulation have been impressive. Who would have thought 20 years ago that terminal emulation would enable:

- Mainframe access through Windows, via a networked device.
- Mainframe access over the Internet, without an application installed on the user desktop.
- A user to take advantage of Web services.
- 3270 terminal emulation on a hand-held, wireless device to view and input host data from virtually any location.

Without the advances of terminal emulation technology of the last 20 years, the mainframe might well be dead and gone by now. Instead, it keeps finding new uses. This article reviews the key transitions in terminal emulation over the past twenty years, and considers what the future will bring.

A HISTORY

Putting terminals on the network – mid-1980s

In the mid-1980s, 'dumb' terminals had to be wired directly to the host via coax cables, and were not networked to other machines within an organization. With the onset of terminal emulation, users could communicate with the mainframe over a network adapter, and at the same time talk to file and print servers via a LAN. Networking vendors, such as Novell, created new ways of getting to hosts using network protocols such as Internetwork Packet Exchange (IPX). Users could share files and resources with each other, albeit in a rudimentary fashion compared with today.

DOS to Windows – early 1990s

In the early 1990s, Microsoft Windows began its domination of the user interface. The graphical interface brought personal computers to the masses, making everyday tasks like copying, pasting, and moving files less intimidating. This was also a period of prolific development and mass adoption of Windows-based terminal emulation products.

The Internet comes to the forefront – 1990s

Sparked by the Internet revolution, vendors like Netmanage and WRQ delivered TCP/IP as a single protocol for all networks and, more importantly, host communication. The single protocol not only simplified desktop configuration for networking but addressed the need to configure and maintain multiple protocol stacks, like NetBios, IPX, and DLC.

16-bit to 32-bit technology – mid-1990s

With the release of Windows 95 and the growing acceptance of Windows NT for the corporate desktop, terminal emulation vendors responded with 32-bit, multi-threaded versions of their Windows emulators. Attachmate was the first to deliver a 32-bit version, and other vendors, like WRQ, Wall Data, and IBM, followed suit.

Multi-host access – late 1990s

By the late 1990s, most major enterprises had a mix of Unix, Digital, and HP hosts because of differing user needs, mergers, and acquisitions, etc. Such heterogeneous environments generally required assorted terminal emulators from assorted vendors, and multi-host emulation solved that problem. The initial focus of multi-host emulation was ease of use and increased productivity; cost savings would become important in years to come.

Web-to-host – late 1990s

A new generation of terminal emulation began in 1996 with host access via a Web browser. At first, the functionality of Web-based emulation was somewhat limited, but its small footprint, easy deployment, and manageability were clear advantages. ActiveX, Windows Terminal Services (WTS), and Java were the platforms available for this new architecture. Some vendors picked Java; other vendors sided with ActiveX and WTS to provide thin clients via the Web browser.

Windows 2000 – February 2000

In February 2000, Microsoft introduced several important innovations with Windows 2000, including a new installer technology (MSI) and Active Directory services. As with the release of Windows 95, interoperability with the OS was a key issue for terminal emulation products. Microsoft established a strict set of requirements for Windows 2000 certification to ensure compatibility, validated by a neutral third-party vendor. Visual Basic for Applications (VBA) was also introduced to terminal emulation, and provided a highly sophisticated method for automating complex end-user tasks and customizing user interfaces.

Today – 2003

Today, new technologies are shaping the terminal emulation market and interesting trends are emerging around earlier innovations. Among these, three are notable:

- Multi-host access
- Web-based emulation
- Security.

Multi-host access, using a single emulation product, is becoming increasingly important as a cost-saving practice. For example, some companies are opting for multi-host access in order to reduce training costs or obtain volume purchase discounts by consolidating with one vendor.

With the global economic downturn of the early 2000s, corporations are being forced to do more with less and maximize their use of existing infrastructure. A good example of this is IKEA, a global retailer of home furnishings, which has applications running on mainframes, AS/400, Unix, and OpenVMS hosts. IKEA needed a solution that covered all of these systems with a single interface, and decided to standardize host connectivity on WRQ Reflection's Windows-based family of emulators. Keyboard mapping allowed its users, who were familiar with certain host applications, to continue to access data with familiar key combinations.

Web-based terminal emulation is becoming increasingly attractive. After several years of refinement, Web-to-host technology offers lower costs and greater manageability. At the same time, most companies need both Web- and Windows-based emulation. Web-based emulation suits many users' needs for day-to-day interaction with the host, and is easy for administrators to deploy and maintain. Other users still need the full functionality of Windows-based emulation because, for example, it offers complex scripting solutions and HLLAPI-based automation. Vendors that can offer strong products in both Web- and Windows-based emulation, and allow users to switch back and forth, are in the best position to address the needs of the current market.

Security remains a critical issue for Web-based emulation, which uses an uncontrolled network – the Internet – to access sensitive host data. Large companies today also understand that data encryption, long overlooked in Windows-based emulation, is important to ensure that their valuable data is safe.

ON THE HORIZON

Web services can extend terminal emulation with functionality provided by Web-based components. For example, Reflection users can configure Reflection to consume Web services using a built-in tool, Microsoft's Visual Basic for Applications (VBA). New Web technology is likely to be incorporated into terminal emulation in other ways as well, including access to terminal emulation configurations via a 'portal' Web page, and writing configuration information directly to XML.

Mainframes will continue to play a central role in the corporate enterprise well into the future. Because over 70% of all corporate data resides on mainframes, customers still have significant needs around these systems and terminal emulation vendors still have plenty of opportunities to address them. The vendor challenge is to find ways to incorporate new technologies, capitalize on transitions, and extend the functionality of existing solutions.

John Cahill
Product Manager, Reflection Business Unit, WRQ
(USA)

Information point – reviews

RESOURCE LINK – http://www.ibm.com/servers/resourcelink

Unlike most other IBM Web sites, Resource Link requires a user ID and password. Registration is free but is limited to customers, IBM employees, and business partners. It usually takes less than two business hours to receive e-mail confirmation.

As stated on the Sign In page: "Resource Link is a customized Web-based solution, providing everything you need to plan for, install, and maintain your IBM zSeries 800, zSeries 900, and IBM System/390 servers and associated software."

Once you've signed in, the home page is divided into four areas:

- Site news
- Key resource link areas
- Hints, tips, & FAQs
- What you'll need.

There's also a left sidebar with a range of options, amongst which several are worthy of note here. For example, 'Fixes' provides single Web page access to links to mainframe hardware and software maintenance. 'Problem solving' provides some overlap, but is a good single page set of links to major IBM support sites useful for mainframe hardware and systems software problem management, as well as public forums, such as IBM-Main.

Click on 'Planning' and then, in the page's Capacity subsection of Servers section, select Network capacity and planning. You'll find yourself on the publicly-accessible IBM page, which contains the latest mainframe network performance news:

`http://www.ibm.com/servers/eserver/zseries/networking/capacity_planning.html`

IBM freeware

In the left sidebar of the Resource Link home page, clicking on Tools and then the Product tools link that appears just below it offers you a choice of links in the 'z/OS and OS/390' section of the page:

- z/OS Unix System Services tools and toys
- z/OS and OS/390 download library
- XML toolkit for OS/390
- E-business sample code downloads
- z/OS and OS/390 data management tools
- z/OS planning and configuration wizards
- OS/390 planning and configuration wizards.

For example, following the second link, one of the programs listed is the Logrec Viewer. This provides ISPF access to the Logrec error log for those using a System Logger stream (LOGR) – great for debugging major problems with VTAM and other system software products.

zSeries System Programmer Portal (SPP) – http://rL11.gpL.ihost.com/zseries/spp

(Note that in the URL shown above I've used upper-case 'L' characters for clarity; any case is acceptable.)

When I first saw this site listed in Site News on Resource Link, I assumed that it was part of Resource Link – until I noticed the odd URL. In fact, IBM owns the ihost.com domain name, and SPP doesn't require a user ID and password.

The site leads you to existing publicly-accessible material on ibm.com. But, as its name implies, it selects the material that will be of interest to the mainframe System Programmer. It takes you a couple of clicks from the home page, but selecting Networking in the Performance section and then zSeries Network Performance in the Networking section gets you to the same

Network capacity and planning page that Resource Link took you to (see above).

Although there are exceptions, including the example just above, you'll generally find that Resource Link provides access to more information than SPP, but takes more clicks to get there. There's also considerable overlap between the two. I recommend, and plan to use, both when searching for IBM information as a mainframe System Programmer.

Jon E Pearkins
(Canada)

TCP/SNA news

Jacada has added a new connector to its Jacada Integrator for directly accessing IBM mainframe CICS and IMS transactions, paving the way for additional transaction connectors to CA-IDMS/DC, IBM TPF, HP3000, and Fujitsu Siemens BS2000 mainframes.

Integrator is used for integrating core legacy business systems, including the data and processes in those systems, with any front-office, call centre, CRM, or Web application, as well as with a wide variety of middleware.

URL: http://www.jacada.com/News/PR166 .htm

* * *

Fluke Networks has released OptiView Console 6.0, providing real-time views of enterprise activity, topology changes, errors, and alarms across an enterprise network.

It expands on its predecessor by providing much deeper views into the network, including views of LAN, WAN, and wireless access points. It integrates results from any connected OptiView network analyser and other network management platforms, including CiscoWorks and Distributed Sniffer Systems.

URL: http://www.flukenetworks.com/us/_ Promotions/ESV/ONAS+Splash+Page.htm

* * *

Compuware has begun shipping Version 8.3 of its Uniface development environment, with increased capabilities for integrating with existing applications through Web services support. Built-in functionality can now be made available as a Web service, enabling integration with J2EE, .NET, and legacy applications. Also, CORBA support has been enhanced to include the bundling of The Ace Orb (TAO), an open source object request broker.

URL: http://www.compuware.com/press room/news/2003/2003040701.htm

* * *

Plumtree Software has launched its Plumtree Enterprise Web Development Kit, a set of development tools for using the Plumtree Enterprise Web Suite to build personalized interactive applications from Web services running on different platforms.

URL: http://www.plumtree.com/news_ events/pressreleases/2003/press042803c. htm

* * *

Oracle has begun shipping its Oracle9*i* Application Server Java Edition for developing and deploying Java applications and Web services. It includes both an application server and an integrated development environment, and offers development and deployment capabilities for transactional business applications, Web services, dynamic Web sites, and e-commerce storefronts.

URL: http://www.oracle.com/ip/deploy/ias/ features/index.html?java_edition.html

* * *