RAS to the Rescue

Stand-alone dump best practices

Also in this issue:

- Migration for z/OS V1R9
- Diagnosis tools and techniques for:
  - Enterprise Extender
  - z/OS UNIX System Services
  - VSAM Record Level Sharing
- Mashups for the masses with IMS
- Java perks
- z/OS UNIX for the MVS system programmer
- Hot Topics “How Tos”
This issue of z/OS Hot Topics features even more RAS-ma-taz than usual! By RAS, of course, we mean the classic strengths of reliability, availability, and serviceability. Often the unsung hero of z/OS, RAS is easy to take for granted because it is designed into System z hardware and software. Today’s RAS standards are the result of more than 30 years of continuous improvement. Though we might sometimes neglect to sing its praises, RAS has been an important factor in making System z the premier operating environment that it is.

Fearlessly displayed on our cover is RAS champion, Bob Abrams, the zManagement Infrastructure Software architect. As Bob and co-author, Jodi Everdon, explain in our feature article “A SAD movie with a happy ending,” for those rare instances when you need to take a stand-alone dump (SAD), you can follow a set of best practices that make for happy endings.

The RAS theme runs throughout this issue, with plenty of tips along the way for a wide range of z/OS products and functions, like Enterprise Extender, SVC dump, z/OS UNIX System Services latch contention, VSAM, and more!

For those of you about to take the plunge into z/OS V1R9, come on in, the water’s fine! To help you get your feet wet, author Marna Walle provides an overview of migration actions and updates for IBM Migration Checker for z/OS.

We also shine the spotlight on the hardware heroes of System z with great articles on the benchmark performance of System z servers running Java workloads in “z/OS on a green machine” on page 66, and coupling facility basics in “Easy as A-B-CF” on page 44. For the I/O enthusiasts among us—who isn’t one of those?—we offer articles on recent enhancements to HCD and HCM.

Of course, z/OS provides the RAS-rich platform for all your subsystems and applications. Take IMS, for example. In this issue, IMS gets a fashion makeover in “Cool new threads for IMS” by William Li and Christopher Holtz and is unleashed on the Web in “Mashups for the masses,” by Jenny Hung and Shyh-Mei Ho.

Want more? We serve up another batch of our handy Hot Topics “How to” articles on Enterprise Extender and z/OS UNIX System Services.

By the way, how’s your JCL? If it’s as rusty as your Latin, see Amy Lander’s “Friends, Romans, Countrymen! We’ll lend you our JCL” (back cover) for an overview of the new reusable online JCL collection, available for your training needs at the z/OS Basic Skills Information Center.

We also continue to welcome newbies to the System z fold—see the Academic Initiative interviews with four new mainframe recruits “Future perfect” on page 59. And don’t forget to test your knowledge of IBM acronyms with our fun quiz “I-M? Or I-B-M?” on page 54.

Yes, this issue of z/OS Hot Topics will have you leaping over tall buildings! You’re sure to find what you’re looking for in this Super issue. Enjoy!

The Editors

newsletr@us.ibm.com

For information about obtaining future issues or hardcopy, go to:
ibm.com/systems/z/os/zos/bkserv/hot_topics.htm
Contents:

Featured articles

Best practices for stand-alone dump
BOB ABRAMS AND JODI EVERDON

Diagnosing basic problems in a z/OS UNIX System Services environment
JOHN C. SHEBEY III

Migration tips make for easier travel
MARNAL WALLE

use the new EEDIAG TEST command
MICHAEL FITZPATRICK, PATRICK BROWN, AND RANDALL KUNKEL

z/OS UNIX terminology made easier to swallow

MIGUEL FITZPATRICK, BOW WANG AND RANAN KUNKEL

Migrating to ICSF HCR7750
BOB PETTI AND PEGGY LABELLE

IBM Migration Checker for z/OS is ready for z/OS V1R9!
MARNAL WALLE

SVCDUMP is here!
JUDI BANK

how to diagnose basic z/OS UNIX latch contention
WILLIAM HUBER AND JOHN BROOKS

how to diagnose advanced z/OS UNIX latch contention
AHILAN RAJADEVA AND MICHAEL SPIEGEL

New VSAM RLS diagnostic commands show you what’s really going on
DOUGLAS LEHR, ROITI PRIETO, AND JASON LEE

Are your VSAM applications out to lunch?
LARRY L. LAW, DEBBIE BENJAMIN-DECKERT, AND TERI M. MENENDEZ

RACF ghost hunters
RUSS HARDGROVE AND JUN OGATA

Is that RACF in my Java?

Rediscover the magic of RACLST
JUN OGATA AND RUSS HARDGROVE

Using the IBM JZOS Batch Toolkit for z/OS SDKs
ROBERT FAIRLIE

The latest advice on tuning your Java heap
MIKE YOUNG AND KEN IRWIN

Working with our vendors to win
NICOLE M. FAGEN AND AUTUMN A. HOUSEKNECHT

zFavorites

What do we tell your ISVs about enablement assistance to System z?
ANDREW D. SCHMIDT AND SUSAN M. TUCZYNSKI

Parallel Sysplex concepts demystified
DAVID RAFTEN

WLM helps dispatch blocked workloads
JUERGEN SCHNEIDER, KARL-HANS HOLDER, AND ARMIN BUECHELE R

HCM I/O configurations
JOCHEN HOFFMANN

I/O timing facility can help
PURVI PATEL, JOHN STAUBI, AND HARRY YUDENFRIEND

Keeping perspective on complex I/O configurations
SIGRUN ELSENS

Here’s to the SAP
JOHN H. KETTNER

I-M? Or I-B-M?

Setting up OpenSSH
PAUL EVANS, ERIN FARR, AND C.T. WARE

move z/OS UNIX directory trees with pax
GARTH GODFREY

Studying mainframes
KATHY PFEIFFER

Unleash IMS with Info 2.0
JENNY HUNG AND SHYH-MEI HO

XML database and XQuery support
WILLIAM LI AND CHRISTOPHER HOLTZ

Championing performance and efficiency
MICHAEL BUECHELE, JOHN P. RANKIN, AND ELISABETH STAHL

Solving common riddles to good application performance
MARK WISNIEWSKI, ED MEKEL, AND JUDI BANK

Using Rational Developer for System z
JOHN DIAMOND AND RONALD TOWNSEND

Your communications serve z/OS Communications Server
Michele CARLO

IPCS class acts

New z/OS Information Center beta
JODI EVERDON AND WAYNE O’BRIEN

A fractured look at problem re-creation
JOEL MASSER

Using z/OS LDAP
VICTOR SACHAR, YVONNE CARDINAL, AND KEN MORGAN

z/OS C/C++ man pages now on the Web!
ADAM LICKEL AND BARBARA NEUMANN

Our contributors

Back cover:
The Reusable JCL Collection
AMY LANDER
A SAD movie with a happy ending

Best practices for stand-alone dump

BY BOB ABRAMS AND JODI EVERDON

Have you seen any good movies lately? Imagine an exciting new Hollywood thriller, one filled with surprising twists and turns. It features the plight of two intrepid system programmers caught in a harrowing race against time!

More than mere escapist fare, this movie uses themes drawn directly from real life experiences (ours). Though the names have been changed, the premise is all too painfully familiar.

Perhaps we would call this suspenseful film *Silence of the Business Phones*. Or *Dial M for Maintenance*. Or, even, *The Castle of Count Dumpula* (starring Bela Diagnosi).

In any event, with some careful preparation, you can help to ensure that a distressing scene like the one below never plays at your workplace:

No one in real life ever looks forward to taking a stand-alone dump. It usually means that your system is non-responsive and you haven’t a clue what is wrong. Eventually, you determine that you need to collect doc in the form of a dump and reIPL.

Truly, a SAD movie for the entire cast of characters!

**Expecting the unexpected**

A stand-alone dump is a rare occurrence, but if you ever do need to take one — especially for a system with fifty gigabytes or more of real memory — you must be able to create one correctly. In addition, you need to know how to send the dump to IBM Support for analysis.

The best way to be prepared for an urgent situation is by evaluating your installation’s current process and making the necessary changes. To help you begin, the folks in Reliability, Availability, and Serviceability (RAS) at IBM got together to draft best practices for creating a stand-alone dump. Here is the short version:

2. Create the multi-volume SADUMP (use the AMDSADD utility).
3. Choose the correct attributes for the IPCS dump directory.
4. Prepare the dump for further processing using the IPCS COPYDUMP command.
5. Compress data for faster transmission and analysis; use AMATERSE or TRSMAIN.

**Fade in:**

INTERIOR: Offices of Running-the-World, Inc. Friday 12:05pm.

Rob Barry sits at his desk, frowning at his system console. Jason Weber leans out of the cubicle across from Rob, eating, what appears to be, salami-on-rye.

Rob: What’s going on with the system, it’s not responding.

Jason: Is it hung? Do we need to reIPL?

Rob: Maybe—is this console even running? (Looks again at the screen.) It’s not doing anything! I even hit Enter a couple times, but it’s just staring at me!

Jason: Darn, it’s hung all right. We need to IPL.

Rob: Hang on; we’d better take a stand-alone dump first. The support people got annoyed at us for forgetting last time.

Jason: I haven’t had to do a stand-alone dump in three years. Last time I ran one, the console kept asking me for a device that I didn’t even know I had... Well, at least I found the run book! Oh, wait—look at the date!

Rob: 1989! Do you think there’s an easier way to do this by now?

Jason: There must be. Let’s call IBM Support to find out.
6. Use PUTDOC to transmit the data to IBM.
7. Set up remote access.
8. Test your stand-alone dump operations.
9. Automate the SADMP process.

You can find these best practices fully described in the IBM Techdocs Library at IBM.com/support/techdocs/atsmastr. Meanwhile, here is an overview to help you started on improving your installation's setup for stand-alone dump.

Planning the setup

Keep these guidelines in mind when planning for a multi-volume dump data set:

- Place the dump output data sets on an IBM System Storage DS8000™ Turbo (DS8000 Turbo, for short) for better performance.
- Use more DASD striping for stand-alone dump, up to the maximum of 32 volumes, for better performance.
- Use more DASD striping for stand-alone dump, up to the maximum of 32 volumes, for better performance.
- Use FICON® channels, instead of ESCON®, to yield the best transfer rates.
- Use the AMDSADDD utility to define a stand-alone dump data set, and be sure to catalog the data set to prevent accessing the "wrong version" afterwards.
- Do not mirror the SAD volumes using disk remote copy. Instead, use a separate set of volumes, pre-allocated and formatted for your disaster recovery site.
- Choose the correct attributes for the IPCS dump directory. Reduce the number of control interval (CI) splits during the initialization and analysis of dumps.
- Place the dump output data sets on an IBM System Storage DS8000™ Turbo (DS8000 Turbo, for short) for better performance.
- Use more DASD striping for stand-alone dump, up to the maximum of 32 volumes, for better performance.
- Use FICON® channels, instead of ESCON®, to yield the best transfer rates.
- Use the AMDSADDD utility to define a stand-alone dump data set, and be sure to catalog the data set to prevent accessing the "wrong version" afterwards.
- Do not mirror the SAD volumes using disk remote copy. Instead, use a separate set of volumes, pre-allocated and formatted for your disaster recovery site.
- Choose the correct attributes for the IPCS dump directory. Reduce the number of control interval (CI) splits during the initialization and analysis of dumps.

Preparing the output

A stand-alone dump by itself is unformatted. To make its contents readable, you need to use Interactive Problem Control System (IPCS) to format the dump. After IPCS post-processes the dump, you can view it with IPCS.

Sometimes, your IBM Support representative can diagnose a problem with just a subset of the stand-alone dump. If we ask you for a subset, you can use COPYDUMP, a powerful IPCS command, to create a subset dump data set to allow IPCS to access it properly.

The IPCS COPYDUMP command syntax looks like this:

```
COPYDUMP ODS('OUTPUT DATASET NAME')
  ASIDLIST(1:20,/JES _ ASID,Catalog _ ASID,Problem _ ASID)
  IDS('INPUT DATASET NAME') NOCONFIRM
```

Preparing the output

A stand-alone dump by itself is unformatted. To make its contents readable, you need to use Interactive Problem Control System (IPCS) to format the dump. After IPCS post-processes the dump, you can view it with IPCS.

COPYDUMP works faster than the IEBGENER utility, and produces a merged dump ordered by address space id (ASID). COPYDUMP processes the dump volumes in parallel, allowing for faster reading of the original dump. COPYDUMP also eliminates the need to reprocess the dump, allowing IBM Support to work with the dump sooner.

Here's how it works:

1. Use IPCS COPYDUMP (not IEBGENER) to produce a merged dump data set from the multi-volume stand-alone dump and a subset of the original stand-alone dump (ASIDs 1-20).
2. Ensure the output data set specified to COPYDUMP is DASD-striped, with at least eight stripes.
3. If you are running z/OS® V1R7, install PTF UA26080.
4. Catalog the output dump data set to allow IPCS to access it properly.

The plot thickens!

Returning to our movie, we see that our heroes have made a momentous discovery:

```
INTERIOR: Offices of Running-the-World, Inc. Friday 3:15pm.

Jason: Shoot! It's our anniversary tonight and my wife has tickets to see The Farnsworth Invention on Broadway. I'm never going to make it!

Rob: Don't worry, Tom at IBM told me about COPYDUMP and he said he could shoot the dump right now, if we have AOS or RSVSF... did you set up remote support?

Rob: Yeah, we have AOS.
```

Remote-assistance tools allow your IBM support representative to view your screen output. Here are some choices, but check with your representative to learn more about all of your options, based on your location:

- RSVSF: In the United States, you can allow IBM Support to analyze your dumps remotely through Remote Screen Viewing Support Facility (RSVSF), an SNA-based application.
- ONTOP: In Europe, you can allow IBM Support to analyze your dumps remotely through On-line Technical Operation (ONTOP), a TCP/IP-based application.
- Assist On-Site (AOS): Worldwide, you can allow IBM Support to view your IPCS data remotely through Assist On-Site (AOS), a TCP/IP-based application.
support representative connect through the Internet (your data is encrypted with 128-bit AES encryption). When you disconnect, your IBM Support representative can no longer see your screen or access your computer. For more information, see [ibm.com/support/assistonsite](http://ibm.com/support/assistonsite).

How to get to Broadway?

Practice, practice, practice. Rehearse taking a stand-alone dump whenever you run a disaster recovery drill on your Test system. Also, consider practicing when you migrate to a new z/OS release, or move to a new processor.

If you have a test LPAR that you use to train your operations staff, include stand-alone dumps as part of the training. During a practice drill, follow your local procedures (as you would in an actual situation) and prepare for reacting to stand-alone dump messages. Besides taking a stand-alone dump, practice using COPYDUMP to obtain a merged and subset dump. With these practices in place, you will be better prepared, should you ever need to take a stand-alone dump.

If we revisit our opening scene, we find that this SAD movie now has a happy ending:

Fade in:
INTERIOR: Offices of Running-the-World, Inc. Friday 12:05pm.

Rob: What’s going on with the system? It’s not responding.

Jason: Looks like we have to take a stand-alone dump.

Rob: No problem!

Jason: Yeah, it’s really not too bad now that we have those stand-alone dump procedures in place.

Rob: And, don’t forget that practice drill! That really helped!

If we ask you for a subset, you can use COPYDUMP, a powerful IPCS command, to create it.

It’s all in the details

How does our list of best practices compare to those in use at your installation? Are you set-up to take a stand-alone dump quickly?

Before the spotlight finds you unprepared, here are some resources to help you prepare for that rare moment when you must take a stand-alone dump — and center stage:

- z/OS V1R9 Diagnosis bookshelf at [publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/EZ2DGZ10](http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/EZ2DGZ10).
- IBM Education Center: Go to z/OS Operating System > General > Service > z/OS large stand-alone dump best practices at: [publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp](http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp).

Fade in:
INTERIOR: Offices of Running-the-World, Inc. Friday 12:05pm.

Rob: What’s going on with the system? It’s not responding.

Jason: Looks like we have to take a stand-alone dump.

Rob: No problem!

Jason: Yeah, it’s really not too bad now that we have those stand-alone dump procedures in place.

Rob: And, don’t forget that practice drill! That really helped!
any of the core business applications that your business relies on were originally written to use SNA networking. Over time, as your installation converts from SNA networking to IP networking and uses more applications that are written for IP, you might seek ways to consolidate your SNA traffic onto the IP network. Doing so can help reduce network complexity and reduce the number of physical links you need to support for network traffic. In the past, this is where you encountered a big speed bump: the impracticality of converting existing SNA applications to TCP/IP.

Now, with Enterprise Extender (EE), you can access these SNA applications across an IP network. To the end user, the session is SNA (with its predictable performance and high availability), but the SNA traffic is encapsulated as UDP datagrams and carried over an IP network without changing the SNA application.

With this obstacle to IP network migrations removed, let’s look at a new command that can help you diagnose problems in your IP network.

Can’t get there from here?
Even after you have read every manual on setting up Enterprise Extender, and created every required definition, you might still encounter a connection between two endpoints that is not working as expected. For example, you cannot activate the connection, or its performance is degraded. Naturally, you review your Enterprise Extender definitions, but you find that everything is coded correctly. Therefore, you conclude, the problem probably lies within your IP network.

Now, how do you determine which network paths are used by the connection? And, how do you locate the node in the network that is causing the problem? Read on—we have just the thing for diagnosing snags in your IP network!

Getting up to speed with EEDIAG TEST
In z/OS V1R8, we introduced a new operator command to help you diagnose problems with Enterprise Extender in your IP network: DISPLAY NET,EEDIAG,TEST=YES. Or, EEDIAG TEST for short.

<table>
<thead>
<tr>
<th>IST2130I</th>
<th>ENTERPRISE EXTENDER CONNECTIVITY TEST INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IST2132I</td>
<td>LDLC PROBE VERSIONS: VTAM = V1 PARTNER = UNKNOWN</td>
</tr>
<tr>
<td>IST1680I</td>
<td>LOCAL IP ADDRESS 10.10.10.100</td>
</tr>
<tr>
<td>IST1680I</td>
<td>REMOTE IP ADDRESS 10.10.10.200</td>
</tr>
<tr>
<td>IST2133I</td>
<td>INTFNAME: OSA1 INFTYPE: IPAQENET</td>
</tr>
<tr>
<td>IST2135I</td>
<td>CONNECTIVITY UNSUCCESSFUL SENSE: <em><strong>NA</strong></em> PORT: 12000</td>
</tr>
<tr>
<td>IST2136I</td>
<td>CONNECTIVITY TEST ENDED - MAXIMUM TIME LIMIT EXCEEDED</td>
</tr>
<tr>
<td>IST2137I</td>
<td>1 10.10.10.1 RTT: 3</td>
</tr>
<tr>
<td>IST2137I</td>
<td>2 * RTT: <em>N/A</em></td>
</tr>
<tr>
<td>IST2137I</td>
<td>7 * RTT: <em>N/A</em></td>
</tr>
<tr>
<td>IST2135I</td>
<td>CONNECTIVITY SUCCESSFUL PORT: 12001</td>
</tr>
<tr>
<td>IST2137I</td>
<td>1 10.10.10.1 RTT: 4</td>
</tr>
<tr>
<td>IST2137I</td>
<td>2 10.10.10.2 RTT: 7</td>
</tr>
<tr>
<td>IST2137I</td>
<td>3 10.10.10.3 RTT: 9</td>
</tr>
<tr>
<td>IST2137I</td>
<td>4 10.10.10.200 RTT: 11</td>
</tr>
<tr>
<td>IST2135I</td>
<td>CONNECTIVITY UNSUCCESSFUL SENSE: <em><strong>NA</strong></em> PORT: 12002</td>
</tr>
<tr>
<td>IST2136I</td>
<td>CONNECTIVITY TEST ENDED - MAXIMUM TIME LIMIT EXCEEDED</td>
</tr>
<tr>
<td>IST2137I</td>
<td>1 10.10.10.1 RTT: 4</td>
</tr>
<tr>
<td>IST2137I</td>
<td>2 * RTT: <em>N/A</em></td>
</tr>
<tr>
<td>IST2137I</td>
<td>7 * RTT: <em>N/A</em></td>
</tr>
</tbody>
</table>

Figure 1. Using EEDIAG TEST to locate a problem node
To probe the network, the command sends encapsulated SNA packets over the five UDP ports reserved for Enterprise Extender. The command verifies the ability of Enterprise Extender to reach the remote endpoint and displays information about each node on the network path traversed between the two endpoints.

Now let’s look at how you might use EEDIAG TEST to diagnose some common networking problems that cause Enterprise Extender connections to misbehave.

Traffic tie-up #1: Firewall is not properly configured

If you cannot activate an Enterprise Extender connection, it might be due to a firewall. If a firewall is located between the two endpoints, it must be configured to permit all five of the Enterprise Extender UDP ports.

You can use the EEDIAG TEST command to help pinpoint misconfigured firewalls. In Figure 1, we see the EEDIAG TEST command output for the area between endpoints 10.10.10.100 and 10.10.10.200.

For this example, we used a single network that is accessible through interface OSA1. As Figure 1 shows, the results of EEDIAG TEST point to a problem with the node at 10.10.10.2, which is not permitting UDP traffic for ports 12000 and 12002.

Traffic tie-up #2: Cannot reach the remote endpoint

If an Enterprise Extender connection is not performing as expected, the problem might be that multipath is enabled on your host and one of the network paths cannot reach the remote endpoint.

To review, multipath allows Enterprise Extender to send batches of UDP datagrams, round-robin style, across two or more local interfaces that traverse different network paths to reach the same remote endpoint. If one of these network paths cannot reach the remote endpoint, Enterprise Extender stops sending datagrams to the network. This action forces retransmission of the datagrams from the local host. If the retransmitted datagrams use an interface that reaches the remote endpoint, the Enterprise Extender connection remains active, but its performance is degraded.

To spot such problems, you can use the EEDIAG TEST command to probe all multipath routes used by the Enterprise Extender connection. In Figure 2, we see the output of the EEDIAG TEST command for the area between endpoints 10.10.10.100 and 10.10.10.200.

For this example, we used two networks, accessible through interfaces OSA1 and OSA2, as follows:

- OSA1 uses a network that can reach the remote endpoint (10.10.10.200)
- OSA2 uses a network that cannot reach the remote endpoint

The node at 10.10.10.22 cannot locate the remote endpoint 10.10.10.200, which is indicated by the ‘D-1’ flag. That is, the node returned a “host unreachable” message in response to the probe.

Back in the fast lane

The EEDIAG TEST command can help you diagnose a number of common networking problems related to Enterprise Extender. It can help take the guesswork out of determining which network paths the Enterprise Extender connection is traversing, and eliminate the need to “sniff” your network to locate the problem.

For more information about the EEDIAG TEST command, see z/OS Communications Server: SNA Diagnosis Volume I, Techniques and Procedures, GC31-6850.
Zombies and threads and forks! Oh, my!

Diagnosing basic problems in a z/OS UNIX System Services environment

BY JOHN C. SHEBEY III

Don’t let the world of z/OS UNIX System Services (z/OS UNIX) scare you. Armed with a little knowledge, you can overcome your fright and examine basic problems in a z/OS UNIX environment like a pro.

Help for messages and codes
Have you ever found yourself in the dark trying to decipher the meaning of a z/OS UNIX errno or errnojr (or errno2)? If so, rest assured, there’s light at the end of the tunnel.

A z/OS UNIX syscall typically returns three values:
• Return value: set to -1 (x'FFFFFFFF'), if an error condition is found
• Return code (errno): integer value associated with generic error condition
• Reason code (errnojr/errno2): four-byte value that provides specific information about failure

If the return value is -1, check the errno and errnojr values for details about the error. Usually, the errno values correspond to standard POSIX errors and aren’t very helpful. On the other hand, the errnojr values (CCCCRRRR) provide a more descriptive meaning of the failure.

The first two bytes (CCCC) are the reason code qualifier, which identifies the failing component. If the reason code qualifier is between 0-20FF, the reason code is from z/OS UNIX. The second two bytes (RRRR) are the reason code value.

BPXMTEXT
The BPXMTEXT utility can quickly interpret z/OS UNIX errnojr values. Find BPXMTEXT in SYS1.BPXEXEC and copy it to SYSEXEC or SYSPROC to use it. You can invoke it from TSO/E, IPCS, or the z/OS UNIX shell. In z/OS V1R8, you can use BPXMTEXT to interpret errnojr values from zFS (reason code qualifier=EFxx) and TCP/IP (reason code qualifier=7xxx). In z/OS V1R9, you can also interpret errnojr values from the C/C++ run-time library (reason code qualifier=Cxx). For more details about interpreting errno and errnojr values, see z/OS UNIX System Services Messages and Codes, SA22-7807.

Message prefixes
z/OS UNIX messages have the prefix BPX and are described in z/OS MV5 System Messages, Vol 3 (ASB-BPX), SA22-7633. z/OS UNIX shell and utilities messages are prefixed by either FSUM or FOM, and are described in z/OS UNIX System Services Messages and Codes, SA22-7807. Alternatively, you can use LookAt to locate the explanation for a message at ibm.com/systems/z/os/zos/bkserv/lookat.

Abends
BPXMTEXT cannot interpret z/OS UNIX abend reason codes, but you can find the explanations for abends and their associated reason codes in z/OS MV5 System Codes, SA22-7626. Here are two abend codes that you might encounter:
• ABENDEC6 during processing of a z/OS UNIX syscall request (the reason code provides more details about the specific error)
• ABEND422 with a reason code of x01zz (note the 01 in the third byte) is also z/OS UNIX specific.

Lurking doc questions
When you least expect it, a problem arises in your z/OS UNIX environment that haunts you. What documentation do you collect? How do you begin troubleshooting the problem?

INFO APAR II08038 contains some general guidelines for collecting documentation in the z/OS UNIX world. Typically, what you need is a dump of OMVS and its dataspaces, along with any problem address spaces, non-OMVS latch holders if there is z/OS UNIX latch contention, and active PFS address spaces that might be relevant to the problem.

Note that you might need to use the wildcard (*) for the jobname of forked address spaces because the jobname of the child is set to the parent’s base jobname with a number from 1 to 9 appended at the end. Here’s an example:

DUMP COMM=(description or dump name)
R xx,JOBNAME=(OMVS,other-related-processes*),CONT
R xx,SDATA=(PSA,SQA,LSQA,TRG,TRT,LSA,CSA,SUM,ALLNUC),CONT
R xx,SDATA=(‘OMVS’*),END

In a shared file system configuration, it’s often necessary to get sysplex-wide dumps of OMVS and its dataspaces. You only need to enter the following dump command on one system to initiate sysplex-wide dumps, for example:

DUMP COMM=(description or dump name)
R xx,JOBNAME=(OMVS),DSPNAME=('OMVS'.*),END
R xx,SDATA=(PSA,SQA,LSQA,TRG,TRT,LSA,CSA,SUM,ALLNUC),END
R xx,REMOTE=(SYSLIST=('OMVS'),SDATA,DSPNAME),END

Setting a SLIP
z/OS UNIX abends often generate a system dump. If no SVC dump is generated, you might need to set a SLIP trap for the abend. For a failing reason code, a SLIP is often required to diagnose the problem. You can SLIP on a z/OS UNIX errnojr. On z/OS V1R8 and above, you can also SLIP on a PFS errnojr. PTF UA30072 is required on z/OS V1R7 to SLIP on a PFS errnojr. Here’s an example:
where: ‘xxxxxxxxx’ is the eight digit (4 byte) reason code to be trapped, and ‘jobname’ is the optional jobname associated with the error.

It is often useful to collect OMVS Component Trace (CTRACE) data when recreating to SLIP on an abend or error

The CTRACE records are stored in OMVS dataspace SYSZBPXX. Use CTRACE as follows (max buffer size 64 MB):

```
SLIP SET,IF,A=SYNCSVCD,RANGE=(10?+8C?+F07+1F4?),
   DATA=(13R?+1B0,EO,xxxxxxxx),DSPNAME=('OMVS'.*),
   SDATA=(ALLNUC,PSA,CSA,LPA,TRT,SQA,RGN,SUM),
   J=jobname,JL=OMVS,END
```

Common problems in a z/OS UNIX environment

It’s important in both non-shared and shared file system environments to collect SYSLOG and EREP for most problems.

The following are some common problems you might encounter in a non-shared file system environment:

Hang in a non-shared file system environment:
1. Enter the D GRS,C console command to check for latch contention.
2. If there’s contention with latch set SYS.BPX.A000.FSLIT.FILESYS.LSN, enter the D OMVS,W console command to identify the file system involved and the associated PFS. For details, see the topic on “Diagnosing and resolving latch contention” in z/OS MVS Diagnosis: Reference, GA22-7588.
3. Dump OMVS and its dataspaces, along with any hung jobs or users, non-OMVS latch holders, and relevant PFS address spaces.
4. Enter the F BPXНикINIT,FILESYS=F console command on any system in the shared file system sysplex to look for any systems that might be performing serialized activity for an abnormal period of time.
5. If a system is identified as performing UNMOUNT or QUIESCE processing for an unusually long period, enter the F BPXНикINIT,FILESYS=F command to further identify which systems are causing the delay.

High CPU/Loop in a non-shared file system environment:
1. Enlarge the system trace table size through TRACE ST,999K, and enable the OMVS CTRACE.
2. Enter the D OMVS,A=ALL console command to review all of the active z/OS UNIX processes.
3. Enter D GRS,C to check for any latch contention.
4. Enter the D OMVS,W command to identify the file system and PFS associated with any latches in contention.

Hang in a shared file system environment:
1. Route the D GRS,C command (ROUTE *ALL,D GRS,C) to all systems in the sysplex to look for any latch contention.
2. Route the D OMVS,W command to all systems to identify the file system and PFS associated with any latches in contention, and to review any outstanding cross-system messages.
3. Route the D GRS,C command to all systems in the sysplex to look for any latch contention.
4. Route the D OMVS,W command to all systems to identify the file system and PFS associated with any latches in contention.
5. Capture sysplex-wide dumps of OMVS and its dataspaces, along with any involved PFS address spaces, non-OMVS latch holders and address spaces suspected to be looping or using high CPU.

Zombie buildup:

A buildup of zombie processes can be a real nightmare. And no, I’m not talking about the walking dead. When a dubbed process terminates, it remains in a zombie state until its status can be reported to the parent process. A large number of zombie processes can be a problem because each zombie takes up a slot in the process table, and therefore counts against the maximum number of processes allowed in the system (MAXPROCSYS).

If you suspect a buildup of zombies, enter the D OMVS,A=ALL command and look for processes with a state of 1Z, which is zombie process state. Determine the parent process ID (PPID).
If the PPID of the zombies is 1, it’s the responsibility of the BPXOINIT address space to cleanup the zombies. Enter the DJ,BPXOINIT console command to verify that BPXOINIT is swapped in and running. If so, you should collect a console dump of OMVS, its dataspaces, and the BPXOINIT address space.

If the PPID of the zombies is not equal to 1, it’s the responsibility of the parent process to cleanup the zombies. The likely reason for this type of zombie buildup is that the parent process did not issue a waitpid() to wait for the child processes to end.

Go forth, and diagnose!

As you can see, deciphering z/OS UNIX messages and reason codes and collecting the appropriate documentation can smooth the progress of working with the z/OS UNIX support center, helping to expedite problem resolution.

For more details, see SHARE Session #2920, Diagnosing Problems in a z/OS UNIX System Services Environment at ibm.com/servers/eserver/zseries/zos/unix/bpxa1pre.html.

z/OS UNIX terminology made easier to swallow

Let’s examine some basic z/OS UNIX terminology.

**Process** is a program, generally at the address space level, using z/OS UNIX. However, it’s not uncommon to see multiple processes in a single address space (multi-proc address space).

**Thread** is a unit of work (task) in a process.

**Dub** is establishing a z/OS UNIX environment for an address space, or more generally the creation of a process.

**Fork and spawn** are two ways that a parent process creates a child process. Forked child processes run in BPXAS initiators, which are WLM-managed initiators that remain active for 30 minutes after their last use.

**Syscall** is a request by an active process for z/OS UNIX System Services to perform a service.

**File** is an object with certain attributes (including permissions and type) that can be written to, read from, or both.

**File system** is a collection of files.

**Mounting a file system** makes it available for use.

**Physical file systems** (PFS) provide z/OS UNIX applications with I/O services. Physical file data resides on DASD (HFS, zFS), in memory (TFS), or is accessible through the network (TCP/IP, NFS). In a shared file system configuration, file system data is shared across the sysplex, and mount and unmount requests are sysplex-wide. XCF messaging is used to transfer data between the file system owner and client systems. The OMVS couple data set contains the common file system hierarchy for all participating systems in the sysplex.

MD John C. Shebey III

Signature John C. Shebey III
Now boarding for z/OS V1R9!

Migration tips make for easier travel

BY MARNA WALLE

To use all the great new enhancements in z/OS V1R9, you must first complete your migration! To get you started on the path to z/OS V1R9, let’s look at some of the required migration actions that you will likely encounter, plus some other actions that could cause your system to lose function if you skip them.

In addition, we offer some tips for actions you can take now to help make your planned migration sail more smoothly. Some tips involve running tools like IBM Migration Checker for z/OS to determine the applicability of certain migration actions to your currently running system. I discuss this tool in “Attention, passengers! IBM Migration Checker for z/OS is ready for z/OS V1R9!” on page 16.

By knowing what lies ahead, you can make your journey to z/OS V1R9 a pleasant experience!

**Action: Remove runtime dependencies on C/C++ IBM Open Class library**

z/OS V1R9 removes runtime support for the C/C++ IBM Open Class® library. Applications that rely on this support will not run on z/OS V1R9; you must update them to use the C++ standard library in z/OS Language Environment®.

**What you can do now:** Before migrating to z/OS V1R9, run the IBM Open Class Detection tool, iocchk, to analyze your programs for references to the removed IBM Open Class library DLLs. You can download this tool from [ibm.com/software/awdtools/czos/support](http://ibm.com/software/awdtools/czos/support).

For programs identified by the tool, update them to use the C++ standard library in z/OS Language Environment (or, if applicable, obtain updates from the software vendor). If, after migrating to z/OS V1R9, you find that you still require the IBM Open Class library DLLs, you can obtain them from the Web site, albeit without warranty, and for a limited time only.

**Action: Run the TN3270E Telnet server in its own address space**

In z/OS V1R6, we began allowing the TN3270E Telnet server to run in its own address space, separate from the TCP/IP stack. In z/OS V1R9, we no longer allow running the TN3270E Telnet server in the TCP/IP stack.

**What you can do now:** If you run the TN3270E Telnet server in the TCP/IP stack, you must now run it in its own address space. Follow the procedure described in z/OS V1R9.0 Migration, GA22-7499.

**Tip:** Use IBM Migration Checker for z/OS, program TN3270AS, to verify that you have done this migration action correctly.

**Action: Review your JES2 exit routines**

Migrating to z/OS V1R9 might require changes to your JES2 exit routines. Relax; these changes won’t be nearly as extensive as the exit changes required for z/OS V1R7.

Here are the JES2 exit changes for z/OS V1R9:

- Exit 31: In z/OS V1R9, you use the $XPL data area to map this exit’s parameter list. (Previously, there was no mapping macro to do so.)
- Exit 42: To accommodate problem state SSI callers, z/OS V1R9 adds more input and output fields to the exit’s $XPL data area.
- Exit 45: If your routine refers to any IAZSSSF fields, you must update the routine for input and output to this exit.

**What you can do now:** Review your JES2 exit routines to see whether they are affected by these changes. If so, add the changes to your z/OS V1R9 migration plan.

**Tip:** Use IBM Migration Checker for z/OS, program JES2EXIT, to determine whether JES2 exits 8, 31, 42, and 45 are active on your system.

**Action: Specify a DFSMSrmm control data set ID**

With z/OS V1R9, a DFSMSrmm™ control data set ID is required for starting DFSMSrmm; it was optional in previous releases.

**What you can do now:** Verify that you are using a DFSMSrmm control data set ID.

**Tip:** Use IBM Migration Checker for z/OS, program RMMCDSID, to verify that you have done this migration action correctly.
On z/OS V1R8 and later, consider updating the products that refer to one-byte console ID references. You can download the latest exclusion list and start the consoles tracking facility. You might need to unserialize callers of CNZMXURF, one-byte console ID references and serialization while accessing the UCME. A future z/OS release might require that CNZMXURF callers hold serialization ISV-oriented exit routines run in a cross-memory environment. As of z/OS V1R9, the following global resource serialization ISV-oriented exit routines run in a cross-memory environment.

**What you can do now:**

- Review the zNALC pricing alternative as a replacement for z/OS.e. Convert from z/OS.e to z/OS with zNALC pricing.

**Tip:**

Use IBM Migration Checker for z/OS, program ZOSEMIG, to verify that you are not using z/OS.e.

**Actions:**

Remove one-byte console IDs and identify unserialized callers of CNZMXURF.

- z/OS V1R7 removed one-byte console IDs from z/OS macros and commands, then z/OS V1R8 removed them from some other places. Some services, however, still contain toleration support for programs assembled before z/OS V1R7. The services treat the IDs differently, so those programs might behave unexpectedly. The CNZMXURF service allows callers to locate the console control block (UCME) that contains a specific console ID. With APAR OA19948, the console tracking facility now reports any unserialized callers of the CNZMXURF service. A future z/OS release might require that CNZMXURF callers hold serialization while accessing the UCME. The PTFs for APAR OA19948 are:
  - z/OS V1R9: UA36125
  - z/OS V1R8: UA36124
  - z/OS V1R7: UA36123
  - z/OS V1R6: UA36122.

**What you can do now:**

To identify one-byte console ID references and unserialized callers of CNZMXURF, start the consoles tracking facility. You can download the latest exclusion list from ibm.com/eserver/zseries/zos/downloads.

For the one-byte console ID references:

- Update the products that refer to one-byte console IDs.
- On z/OS V1R8 and later, consider adding the UNKNIDS routing attribute to console definitions, so that messages directed to one-byte console IDs are received by consoles requesting this attribute.

For CNZMXURF improper serialization references, e-mail the tracking output to consoles@us.ibm.com.

**Tip:**

Before and after migrating to z/OS V1R9, use IBM Migration Checker for z/OS, program ONEBCON, to verify the consoles track facility output and your usage of the UNKNIDS routing attribute.

**Action:**

Gauge the impact of new Language Environment runtime options.

- z/OS V1R9 adds a runtime option, CEEDUMP, which is used to control the processing of the z/OS Language Environment dump report, CEEDUMP. This option does not change the default behavior that existed before z/OS V1R9.
- z/OS V1R8 added the runtime option, DYNDUMP, which controls whether certain dumps are produced automatically. The DYNDUMP runtime option changes the default behavior that existed before z/OS V1R8.

**What you can do now:**

If you still use assembler language modules to specify your installation-wide runtime options (and want to continue to specify them in this manner), plan to update the modules for the new options. To make your migration work easier in the future (and allow for easier dynamic changes to the runtime options), you might want to convert to the CEEPRMxx parmlib member for installation-wide runtime option specification.

- If you use the CEEPRMxx parmlib member to specify your installation-wide runtime options and the defaults for the new options are acceptable to you, there’s no migration action to perform! If you want to change the defaults, however, plan to update your CEEPRMxx parmlib member for z/OS V1R9.

**Action:**

Ensure that the global resource serialization ISV-oriented exit routines run in a cross-memory environment.

As of z/OS V1R9, the following global resource serialization exit routines must be invoked in cross-memory mode to avoid a data integrity problem or system failure:

- ISGNQXITBATCH
- ISGNQXITBATCHCND
- ISGNQXITPREBATCH
- ISGNQXITQUEUEUDI.

**What you can do now:**

Determine whether you are using these exit routines. If so, update them as necessary to allow them to be invoked in cross-memory mode.

**Tip:**

Use IBM Migration Checker for z/OS, program GRSEXIT, to determine whether you are using these exit routines.

**Action:**

Do not enter commands for a copy pool from a pre-z/OS V1R8 system after the copy pool has been converted to z/OS V1R8 format.

As of z/OS V1R8, you can dump to tape copy pool versions residing on DASD. This enhancement changes the format of the DFSMShsm™ control data set (CDS) records. The first time you enter the FRBACKUP command for a copy pool, the system automatically converts the records to the new format. Afterwards, the only fast replication command you can enter successfully for a copy pool from a pre-z/OS V1R8 system is FRRECOV from DASD.

Therefore, before entering an FRBACKUP command for a copy pool on z/OS V1R8 or later system, ensure that your installation no longer needs FRBACKUP processing for copy pools from any pre-z/OS V1R8 systems.

**What you can do now:**

Plan on using only the FRBACKUP command for a copy pool on z/OS V1R8 or z/OS V1R9 when you no longer need to back-up copy pools from any pre-V1R9 system.

**Action:**

Perform ICSF migration actions.

For migration actions for ICSF, see “All aboard! Migrating to ICSF HCR7750” on page 14.

Lastly, remember to pack a good book.

For a complete list of z/OS V1R9 migration actions and more details on the actions described here, see z/OS V1R9.0 Migration, GA22-7499, which is available at the z/OS Internet Library Web site: publibz.boulder.ibm.com/epubs/pdf/e0z2m150.pdf.
The newest release of Cryptographic Support for z/OS V1R7 through V1R9, and z/OS.e V1R7 and V1R8 (HCR7750), is available for download from the Internet. When you plan to migrate to any of these new releases, keep in mind these two migration issues.

Increase the PKDS record size
Integrated Cryptographic Service Facility (ICSF) HCR7750 increases the maximum key size for Rivest-Shamir-Adleman (RSA) Chinese Remainder Theorem (CRT) key tokens from 2048 bits to 4096 bits. To accommodate this change to the token size, you need to increase the record length of the public key data set (PKDS). This change must occur before you start ICSF HCR7750 for the first time. Consider the following effects when you make this change:

- If the PKDS specified in the ICSF options data set does not have the proper record length, ICSF HCR7750 fails to initialize.
- If you share the PKDS with earlier releases of ICSF, you must apply toleration APAR OA21807 to those releases before migrating the PKDS to the larger record size.
- If you share the PKDS with earlier releases of ICSF, you must also perform all re-encipher operations on the HCR7750 systems. Otherwise, attempts to re-encipher the PKDS on older releases will fail.

Perform the following steps to accomplish this migration. Be sure to perform all steps on the system where ICSF HCR7750 is installed.

1. Create a backup copy of the current PKDS.
2. Allocate a new PKDS using the sample JCL provided in SYS1.SAMPLIB(CSFPKDS).
3. Use the sample JCL in SYS1.SAMPLIB(CSFPKDCP) to copy the original PKDS to the new PKDS.
4. Update the ICSF options data set to use the new PKDS.

If multiple systems are sharing the PKDS, update the ICSF options data set on all systems, then stop and restart ICSF. Remember that if the sharing system is using an ICSF version before HCR7750, you must apply toleration APAR OA21807.

Figure 1 is a sample of the JCL you can reference to allocate a larger PKDS. Figure 2 is a sample of the JCL you can reference to copy the original PKDS to a newly allocated PKDS with a larger record size.

---

**Figure 1. Sample JCL to allocate the PKDS with a larger record size**

```jcl
//CSFPKDS JOB <JOB CARD PARAMETERS>
//********************************************************************
//* Licensed Materials - Property of IBM                              *
//* 5694-A01                                                          *
//* Copyright IBM Corp. 2002, 2007                                     *
//*                                                                *
//* THIS JCL DEFINES A VSAM PKDS TO USE FOR ICSF                    *
//*                                                                *
//* CAUTION: This is neither a JCL procedure nor a complete JOB.     *
//* Before using this JOB step, you will have to make the following *
//* modifications:                                                   *
//* 1) Add the job parameters to meet your system requirements.      *
//* 2) Be sure to change CSF to the appropriate HLQ if you choose   *
//* not to use the default.                                          *
//* 3) Change xxxxxx to the volid where you want your PKDS to reside. The PKDS needs to be on a permanently resident volume. *
//*                                                                *
//*********************************************************************
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD  SYSOUT=*  
//SYSIN DD *
DEFINE CLUSTER (NAME(CSF.CSFPKDS) - VOLUMES(XXXXXX) - RECORDS(100,50) - RECORDSIZE(350,3800) - KEYS(72 0) - FREESPACE(0,0) - SHAREOPTIONS(2,3)) - DATA (NAME(CSF.CSFPKDS.DATA) - BUFFERSPACE(100000) - ERASE - CISZ(8192) - WRITECHECK) - INDEX (NAME(CSF.CSFPKDS.INDEX))
```

---

**Figure 2. Sample JCL to copy the original PKDS to a newly allocated PKDS with a larger record size**

```jcl
//CSFPKDCP JOB <JOB CARD PARAMETERS>
//********************************************************************
//* Licensed Materials - Property of IBM                              *
//* 5694-A01                                                          *
//* Copyright IBM Corp. 2002, 2007                                     *
//*                                                                *
//* THIS JCL DEFINES A VSAM PKDS TO USE FOR ICSF                    *
//*                                                                *
//* CAUTION: This is neither a JCL procedure nor a complete JOB.     *
//* Before using this JOB step, you will have to make the following *
//* modifications:                                                   *
//* 1) Add the job parameters to meet your system requirements.      *
//* 2) Be sure to change CSF to the appropriate HLQ if you choose   *
//* not to use the default.                                          *
//* 3) Change xxxxxx to the volid where you want your PKDS to reside. The PKDS needs to be on a permanently resident volume. *
//*                                                                *
//*********************************************************************
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD  SYSOUT=*  
//SYSIN DD *
DEFINE CLUSTER (NAME(CSF.CSFPKDCP) - VOLUMES(XXXXXX) - RECORDS(100,50) - RECORDSIZE(350,3800) - KEYS(72 0) - FREESPACE(0,0) - SHAREOPTIONS(2,3)) - DATA (NAME(CSF.CSFPKDCP.DATA) - BUFFERSPACE(100000) - ERASE - CISZ(8192) - WRITECHECK) - INDEX (NAME(CSF.CSFPKDCP.INDEX))
```
Update calls to CSNDPKG
Retained private keys are RSA keys that are stored in the cryptographic coprocessor instead of in the PKDS. You can use retained keys for key management or signature verification. ICSF HCR7750 continues to use RSA private keys that are stored in the cryptographic coprocessor regardless of the key usage type. You can create retained keys that are used for signature verification with a maximum key length of 2048-bits. However, with ICSF HCR7750, you can no longer create any retained private keys with the key management usage types of KEY-MGMT or KM-ONLY.

Review any of your applications that make use of the retained RSA key function to determine if they require an update. Look for the key generate callable service CSNDPKG that uses the RETAIN® keyword.

For information about migration actions for z/OS V1R9, see the article “Now boarding for z/OS V1R9!” on page 12.


---

```
//CSFPKDCP JOB <JOB CARD PARAMETERS>
//********************************************************************************
//* Licensed Materials - Property of IBM                                        *
//* 5694-A01                                                                  *
//* Copyright IBM Corp. 2007                                                   *
//* THIS JCL COPIES ONE VSAM PKDS TO THE LARGER PKDS                           *
//* CAUTION: This is neither a JCL procedure nor a complete JOB.              *
//* Before using this JOB step, you will have to make the following            *
//* modifications:                                                            *
//* 1) Add the job parameters to meet your system requirements.               *
//* 2) Be sure to change CSF to the appropriate HLQ if you choose             *
//* not to use the default.                                                   *
//********************************************************************************

//STEP1 EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//INDD DD DSN=CSF.CSFPKDS.OLD,DISP=SHR
//OUTDD DD DSN=CSF.CSFPKDS,DISP=SHR
//SYSIN DD *
REPRO INFILE(INDD) OUTFILE(OUTDD)
/*

Figure 2. Sample JCL to copy the original PKDS to the newly allocated PKDS
```
Attention, passengers!

IBM Migration Checker for z/OS is ready for z/OS V1R9!

BY MARNA WALLE

ike many of your peers, you have probably downloaded IBM Migration Checker for z/OS. This tool, which is provided “as is” on the z/OS Download Web page (ibm.com/servers/eserver/zseries/zos/downloads), can help you determine the applicability of particular migration actions for your currently running system. The tool is an automated way to do some of the checking you would otherwise do manually using the z/OS Migration book.

The popularity of this tool has been overwhelming; For z/OS V1R9, we give you more! Let’s look at the latest enhancements to this tool.

But first, a refresher on how it works
IBM Migration Checker for z/OS is composed of several REXX programs that you invoke individually through JCL. The programs, which run on your current system, tell you which migration actions you will need to do to migrate to a newer release.

For each migration action, the tool tells you either of the following:
• You’re “clear for take-off.” That is, the action isn’t applicable to your system or you have already performed the necessary tasks on your current system
• You’re not yet ready for take-off. Plan on performing the migration action.

After the installation of the release you can re-run the programs to verify that you have completed the migration action successfully. Running the tool before you’ve done your z/OS migration, as well as after you’ve completed the z/OS installation, is a good idea.

What’s new in the tool for z/OS V1R9
We have added many new migration programs to the tool for z/OS V1R9, in addition to the existing z/OS V1R8 migration programs. Although not all of the new z/OS V1R9 migration actions are covered with IBM Migration Checker for z/OS V1R9, many are. For a complete picture of what activities you need to do for migration, use the tool in conjunction with the z/OS Migration book.

For z/OS V1R9, the tool examines these additional migration actions:
• Discontinue use of the DIAGxx ISGERQA parameter.
• Ensure that the global resource serialization ISV-oriented exit routines run in a cross-memory environment.
• Migrate from z/OS.e to z/OS.
• Update JES2 exit routines 8, 31, 42, and 45 (four migration actions consolidated into this one migration program!).
• Ensure that _UNIX03 is not set in system profiles.
• Reassemble customized ISFPARMS.
• Specify a DFSMSrmm control data set ID.
• Migrate from a Sysplex Timer to IBM Server Time Protocol (STP).
• Migrate from Quality of Service (QoS) Traffic Regulation (TR) policy to Intrusion Detection Services (IDS) TR policy.
• Remove the Boot Information Negotiation Layer (BINL) Server.
• Remove the Dynamic Host Configuration Protocol (DHCP) Server.
• Remove any customization of the Simple Network Management Protocol (SNMP) sysObjectID MIB object in the OSNMPD.DATA file.

Upgrading to the new level of the tool
If you have the older level of IBM Migration Checker (the one we supplied for z/OS V1R8 migrations), understand that the newer z/OS V1R9 level of the tool completely replaces it. When downloading and installing the z/OS V1R9 level of the tool, you do not need to keep the previous level of the tool, unless you want to. Because the migration programs know which z/OS level you are currently running, the tool can report to you about which migration actions you can do now. For those migration actions you can’t do now, the tool will tell you the z/OS release for which you can do them, and whether the actions are required or just recommended for the release.

We welcome your feedback!
Because we offer the tool “as is,” there is no formal IBM Service support. However, we do accept questions, comments, and suggestions on our forum. For access to the forum, send a note to zosmigrate@us.ibm.com.

Also, be sure to see z/OS V1R9.0 Migration, GA22-7499, which is available at: publibz.boulder.ibm.com/epubs/pdf/e0z2m150.pdf

We hope that you will find this updated tool to be useful for your z/OS V1R9 migration activities.

So climb on board. Here’s to smooth sailing!
Have no fear!

**SVCDUMP is here!**

**BY JUDI BANK**

Some performance mysteries defy the best performance monitors. These obscure puzzles often require a super sleuthing tool — would you believe SVCDUMP?

Do you ever ask yourself these questions?

- Why does my application seem hung?
- What is it waiting for?
- Is it repeatedly calling a particular system service?
- What is the WLM service class and priority?
- Is there ENQ or latch contention?

If the target address spaces contain Java™, C or C++ middleware, or applications, dumping the OMVS data spaces is beneficial because they contain z/OS UNIX System Services file system information. CTRACE buffers are always dumped for dubbed address spaces. For programs running under z/OS UNIX, use the following dump parameters, where jobname is the address space you wish to analyze:

\[
\text{JOBNAME=(OMVS,jobname),DSPNAME=('OMVS'.SYSZBPX1,'OMVS'.SYSZBPX2), SDATA=(CSA,GRSQ,LPA,NUC,RGN,QOA,TXT)}
\]

Ensure that you have sufficient resources before collecting the dump. Dumping large numbers of address spaces, or very large address spaces, can cause a temporary depletion of memory and paging space.

The SELECT ALL IPCS command provides an easy way to find the ASID for a particular job or process if you didn’t already capture this information at the time of the dump.

You can find address space dispatching priority at offset X’2A’ of the ASCB or as field NDP in the SRMDATA report. From the IPCS main menu, select option 2, ANALYSIS, then option 6, COMPONENT followed by SRMDATA.

If the job of interest is not running, you can enter the IPCS command ANALYZE RESOURCE to determine whether it’s waiting for a resource.

**Global Resource Serialization report**

The GRSDATA component report, as shown in Figure 1, contains a more detailed source of ENQ information. Choose option 2 ANALYSIS, then option 6, COMPONENT, and select GRSDDATA. This report also provides information about GRS latch contention. A Slow Obtain means there is contention for the latch. Figure 1 indicates about .5 percent of the latch obtains encountered contention.

<table>
<thead>
<tr>
<th>Latch Set Name: SYS.BPX.A000.BDCI.PTY.LSN</th>
<th>Creator Jobname: OMVS</th>
<th>Creator ASID: 000E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latch Number</td>
<td>Fast Obtains</td>
<td>Slow Obtains</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>1,532,137</td>
<td>5,491</td>
</tr>
<tr>
<td>1</td>
<td>89,632</td>
<td>1,570</td>
</tr>
<tr>
<td>2</td>
<td>46,809</td>
<td>590</td>
</tr>
<tr>
<td>3</td>
<td>1,308</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1,024</td>
<td>17</td>
</tr>
<tr>
<td>Summary:</td>
<td>1,669,903</td>
<td>7,654</td>
</tr>
<tr>
<td>Total number of latches in above latch set: 1,025</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of latches with non-zero statistics: 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. GRS latch contention report
To determine if the address space in question is generating ABENDs, enter the IPCS command VERBX LOGDATA. You can also view master trace (MTRACE) to see if any console messages relate to your performance problem. This is option 2.7.3 on the IPCS main menu.

**Language Environment threads**

If you’re analyzing a contention problem in a process or application that runs under z/OS UNIX System Services, look at the snapshot of Language Environment threads.

From option 6 of the IPCS main menu, enter the command:

```
VERBX LEDATA ‘tcb(xxxxxx) asid(xxxx) nth(*)’
```

The `xxxx` field is the hexadecimal ASID for your job or process, which you obtain from SDSF or IPCS. Each thread snapshot will look similar to Figure 2.

For a complete list of Language Environment modules, see [ibm.com/servers/eserver/zseries/zos/le/assist/modr9nic.html](http://ibm.com/servers/eserver/zseries/zos/le/assist/modr9nic.html).

Most threads are similar to the one pictured in Figure 2. CEEOGLAT is the Language Environment latch service that’s called when contention exists for a latch. Therefore, Language Environment latch contention is causing our performance problem. CEVGTST is the Get Heap Storage service. In other words, our threads are trying to get storage because of an `operator new()` call and contention exists for the heap latch. Luckily, this problem is easy to solve.

Based on the symptoms, we assume that the HEAPPOOLS option is not enabled for this application. The command VERBX LEDATA ‘asid(xxxx) tcb(xxxxxxxx)summary’ displays the Language Environment parameters in effect. When we enabled HEAPPOOLS, voila, the performance problem was resolved!

**OMVSDATA command**

Another type of component data that can be helpful in solving your performance mystery is OMVSDATA, which displays address spaces and tasks for z/OS UNIX. Choose IPCS main menu option 2 ANALYSIS, then 6 (COMPONENT), and specify OMVSDATA. Select the Detail report, and then the Process report for the ASID of the target address space. The Detail report is quite verbose; you probably don’t want all the address spaces. You can see the Process ID (PID) and other general information about this address space including:

- ASID status (active) at the time of the dump
- Name of the last program that ran
- Number of threads in the process
- Files in use.

**Traceback:**

```
<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEEOGLAT</td>
<td>-7FFFFFD52</td>
<td>CEEPLPKA</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CEVGTST</td>
<td>+00000292</td>
<td>CEEPLPKA</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>operator new(unsigned int)</td>
<td>+000000E4</td>
<td>*PATHNAM</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>EventRetriever::handleRetrievedEvent(char*)</td>
<td>+00000506</td>
<td>*PATHNAM</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>EventRetriever::run()</td>
<td>+00000B9C</td>
<td>*PATHNAM</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ThreadEP</td>
<td>+00000080</td>
<td>*PATHNAM</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CEEPGTFN</td>
<td>+0000005A</td>
<td>CEEPLPKA</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CEEOPCMM</td>
<td>+00000908</td>
<td>CEEBINIT</td>
<td></td>
</tr>
</tbody>
</table>
```

**OMVS tracing**

Another instrument in our special performance toolkit for z/OS UNIX applications is OMVS component trace (CTRACE). Select 2.7.1 from the IPCS main menu. Use the Query option with a subcomponent name (for example, SYSOMVS) to display the start and stop times of any existing traces found in the dump. For example:

```
COMP(SYSOMVS)
START = 05/22/2007 20:12:49.659256 GMT
```

Some components always trace errors and significant events. These have a default setting of MIN instead of OFF. z/OS UNIX is such a component. Therefore, if your dump contains the OMVS data space with trace buffers, you can generally see some CTRACE in your dump. If you want to capture a full OMVS CTRACE, you must enable the trace using the TRACE CT,ON,COMP=SYSOMVS command. OMVS tracing can degrade your system, so make sure you disable it after collecting the dump.

To display the trace entries, select the DISPLAY (D option). Local timestamps are easier to interpret. FULL with kerninfo gives you a detailed description of each trace event and identifies the module that’s called for that syscall. Figure 3 shows how to specify the FULL option with kerninfo.

Next, let’s look at the syscall entry and exit events for a RECEIVE system call. P04 is the system name. If you compute the time between the syscall entry and exit, you can see that this syscall took only four microseconds. The hexadecimal data is specific to each type
of syscall and generally not documented although z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589, does contain some information about OMVS syscalls. In Figure 4, the syscall number is 86. Collecting an OMVS CTRACE is one way to determine if there are syscalls that have a long response time or a delay.

The other OMVS CTRACE option you might find interesting is count. There’s a general option called TALLY, which normally gives a count of each event type for most components. However, z/OS UNIX uses the option SCCOUNTS, which includes the per-second rate of each syscall. The syscalls that appear with great frequency are READ, WRITE and RECEIVE. Other high frequency calls might indicate a problem.

**System trace**
The last instrument in our performance toolkit is system trace. Be aware that this trace will degrade your system. The TRACE ST,999K,BR=ON command increases the trace table size and enables branch tracing.

It’s helpful to print the IPCS formatted trace to a data set so you can view it with TSO Edit. Use local timestamps when formatting the trace. Trace event types have varying formats. If you’re not familiar with trace events, see MVS Diagnosis: Tools and Service Aids, GA22-7589.

Look for gaps in timestamps for your address space as well as excessive calls to services. You can sometimes use the IPCS WHERE command to determine the module name of a suspicious caller. If the address is in the private area of storage you can use the BROWSE command to locate the calling module in the dump.

**Go forth!**
Now that you are a performance super sleuth, here is some documentation you might find helpful:

- z/OS MVS Diagnosis Reference, GA22-7588
- z/OS MVS Diagnosis Tools and Service Aids, GA22-7589
- z/OS MVS Interactive Problem Control System (IPCS) Commands, SA22-7594.

Wear your deerstalker (Sherlock Holmes hat) if you like, but please don’t smoke your pipe in the computer room!
Are you an expert in managing latch contentions when a hang situation occurs on multiple systems in a sysplex during z/OS UNIX System Services (z/OS UNIX) operations? For z/OS UNIX users, latch contentions can be a risky business. If you aren’t an expert in solving contentions, we are here to help! This article unravels new function in z/OS V1R8 that can help you diagnose and analyze latch contentions.

Have you ever gone into the bank on a Friday afternoon and discovered there’s only one teller?

- If you arrive when no one is at the window or in the line and nobody has to wait for you to finish your transactions with the teller, there is no contention.
- If you arrive when there is someone at the window, but that person leaves as you approach, there is also no contention.
- If you arrive when someone is at the window and you have to wait, there is contention.
- If there is a long or growing line when you arrive, while you wait, or by the time you leave, there is a lot of contention. The wait can become indefinite. In the case of latch contention, it’s the same if there is a deadlock or other problem in the system.

Meet the latches
Contention occurs when tasks have to take turns obtaining a latch. The mount latch is the number one latch of the z/OS UNIX latches. The mount latch is obtained only exclusively when mutually exclusive operations take place. By operations, we mean things like moving a file system, recovering a lost (dead) system, system initialization, and reading from or writing to a couple data set.

Mount latch
LFS uses the mount latch to serialize operations to one file system at a time. It’s the number two latch in the SYS.BPX.A000. FSLIT.FILESYS.LSN latch set. The mount latch is obtained only exclusively when mutually exclusive operations take place. By operations, we mean things like moving a file system, recovering a lost (dead) system, system initialization, and reading from or writing to a couple data set.

File system latch
When you mount a file system on a system, the file system gets a file system latch. You can view the latch numbers of the file systems by checking the rightmost column in the output of the DISPLAY OMVS,FILE command. The file system latch is obtained exclusively when the file system is unmounted, synchronized, and exported or unexported through Server Message Block (SMB) server or Distributed File Service (DFS™) server. The file system latch can also be obtained exclusively when the file system is moving, or recovering in a sysplex. On the other hand, the file system latch is shared when a file within the file system is being read from or written to. This is to prevent other users or programs from unmounting or moving the file system.

File latch
Each active file or directory is assigned a file latch that can be obtained either shared or exclusively. The file latch is in a shared mode, for example, when reading the names in a directory. The file latch for a directory is exclusive when creating or renaming a file within that directory.

Commanding contention
z/OS UNIX implements a new MVS™ command DISPLAY OMVS, WAITERS (D OMVS,W) to help you identify latch contentions. The D OMVS,W command displays the mount latch and file system latches, and includes outstanding cross-sysplex messages from both local and remote systems.

```
SY3 DISPLAY OMVS,WAITERS
SY3  BPX0631I 14.46.48 DISPLAY OMVS 648
OMVS  000E ACTIVE OMVS=(11,HR) MOUNT LATCH ACTIVITY:
USER ASID TCB REASON AGE
FUNCTION: File System Latch 46 EXCL
IS DOING: ZFS MountCall / Osi Wait
FILE SYSTEM: OMVS,LINUX.ZFS
HOLDING: File System Latch 46 EXCL
WAITER(S): NONE

Figure 1. BPX0631 MOUNT LATCH ACTIVITY
```
The D OMVS,W command is useful when you see one or more systems broadcasting these types of messages:

**BPX056I UNIX SYSTEM SERVICES LATCH CONTENTION DETECTED**

**BPXF042I** POSSIBLE CONTENTION FOR THE FILE SYSTEM MOUNT LATCH EXISTS ON SYSTEM xxx, LATCH NUMBER yyy

The BPXF042I message appears when the system detects a latch contention that persists for an excessive amount of time. Use the D OMVS,W command to help diagnose and relieve the latch contention by displaying who holds the mount, the file system, and the file latches.

Generally, two processes can hold a latch and potentially cause contention:

- **User programs**: When a user program runs a z/OS UNIX file operation, the system obtains the file system latch and possibly the file latch, and holds the latches for the duration of the operation.
- **z/OS UNIX (OMVS) task**: z/OS UNIX can hold the mount latch or a file system latch for extended periods for operations such as system recovery and moving a file system.

**Figuring it out**

The following figures show the output from D OMVS,W that displays in message BPXO063I. These message sections can help you identify and diagnose latch contentions in z/OS UNIX. Figure 1 is an excerpt of the MOUNT LATCH ACTIVITY output that shows what's holding the mount latch and which operations are waiting for the latch.

The HOLDER field shows that the OMVS.LINUX.ZFS file system is in the mount process and is holding the mount latch for 23 seconds. The mount latch holder also holds file system latch 46 exclusively, which tells everyone else to wait until this operation is complete. However, in Figure 1, there are no other tasks waiting to obtain the mount latch. This output includes the address space (000E) and task control block (008D7CD8) of the holder in case you want to trace where the holder resides in storage. Now, let's look at the next part, the FILE SYSTEM LATCH ACTIVITY output, in Figure 2.

In Figure 2, we identify OMVS.LINUX.ZFS as the file system associated with latch 46. As with the mount latch activity, the program is holding this latch exclusively for mounting the zFS file system. Unlike Figure 1, this latch has a waiter called MEGA, a z/OS UNIX program user, waiting for z/OS UNIX to release the file system latch.

Figure 3 shows the exchange of messages, across XCF, between systems in a sysplex. The OUTSTANDING CROSS SYSTEM MESSAGES field has two separate sections, RECEIVED SYSPLEX MESSAGES and SENT SYSPLEX MESSAGES. The received messages are the messages that the local system gets from the remote systems. The last section of Figure 3 includes the messages that the local system sent to the peer systems to let the users know the type of operations that are causing tasks to wait. On the remote system where the message is sent, enter the D OMVS,W command and look in the RECEIVED SYSPLEX MESSAGES section for a message with the corresponding REQID of the system that sent the message. The ASID and TCB fields will also match.

Figure 4 shows the last part of the D OMVS,W output, OTHER WAITING THREADS. These are threads waiting but are not associated with any mount latch or file system latch contention. Sometimes, when it comes to resolving latch contention, these waiting threads can give clues for diagnosing contentions.

To resolve latch contentions, you might need to terminate or cancel the holder (task) of the latch. For more on that subject, see “How to diagnose advanced z/OS UNIX latch contention” on page 22.
In z/OS V1R6, z/OS UNIX System Services (z/OS UNIX) provided additional latch purge capabilities to aid in the prevention of system outages caused by latch hangs in the z/OS UNIX component. It introduced a mechanism to detect unresolved z/OS UNIX latch contentions, and a way to resolve the latch contentions automatically or, if necessary, through a new console command MODIFY BPXOINIT, RECOVER=LATCHES.

Usually, the unresolved z/OS UNIX latch contentions result from one of the following scenarios:

- Jobs or processes that terminate without cleaning up the ownership of the z/OS UNIX global resource serialization (GRS) latches that they acquired
- User tasks that are currently active and holding a z/OS UNIX latch for an excessive period
- System tasks that are holding a z/OS UNIX latch and running in the physical file system (PFS).

Starting in z/OS V1R6, z/OS UNIX monitors all of its latch sets for latch contention. When z/OS UNIX detects latch contention and the latch causing the contention is held for an excessive amount of time (currently five or more minutes), z/OS UNIX automatically attempts to correct the problem that is not resolved. If the oldest latch holder’s job or process no longer exists, z/OS UNIX automatically resolves the latch contention by purging the latch holder’s information and releasing the latch. If the oldest latch holder’s job or process still exists, the following message indicates that further action might be necessary:

BPXM056E UNIX SYSTEM SERVICES LATCH CONTENTION DETECTED

If BPXM056E messages are not deleted from the console after a reasonable amount of time, you need to determine the cause of z/OS UNIX latch contention by using the DISPLAY OMVS,WAITERS and DISPLAY GRS,CONTENTION commands. For details about how to use the DISPLAY OMVS, WAITERS command to determine the cause of the z/OS UNIX latch contention, see the article “How to diagnose basic z/OS UNIX latch contention,” on page 20.

Using BPXOINIT

After z/OS UNIX reports the latch contention, you might need to enter the Modify BPXOINIT, RECOVER=LATCHES command to resolve the latch contention. This command aids in resolving latch hangs related to z/OS UNIX user tasks. BPXOINIT might not be able to resolve latch hangs related to internal system tasks in the OMVS kernel address space, if the owning system task is in critical system code that cannot be interrupted. When using the command, remember that it might abnormally terminate user tasks that are holding the latches for five
or more minutes. Usually, the command terminates only individual tasks, but be aware that it can terminate the entire process.

The MODIFY BPXOINIT command can take several minutes to resolve the latch contention. It first determines whether the same holder holds one or more z/OS UNIX GRS latches for five or more minutes. If so, it then determines whether the holder is still active in the system. If the holder is active and is a user address space task, the command abnormally terminates the task to resolve the latch contention. If the holder is not active (already terminated), the command releases the latch. If the holder is active and is an internal system task, the command further determines whether the internal system task is hanging in an open systems interconnection (OSI) wait or in a logical file system (LFS) operation. If the task is hanging in an OSI wait or in a LFS operation that you can cancel, the internal system task abnormally terminates with an error that causes a failing return code to be issued from the OSI wait service or by the LFS operation. For all other cases, the assumption is that it’s not safe to interrupt the internal system task.

**Those annoying abends**

It is important to note that abnormal termination with a 422-1A5 abend code causes the generation of a system dump. This is because the latch contention might indicate an internal operating system problem that the dump can help diagnose. Additionally, if more than one latch is in contention, multiple tasks can abnormally terminate and result in multiple dumps.

In z/OS V1R9, the MODIFY BPXOINIT,RECOVER=LATCHES command helps capture multisystem dumps for the 422-1A5 abend, if the user address space task that’s causing the latch contention is holding the mount or virtual file system (VFS) latch and is in a shared file system configuration (sysplex). In addition, the multisystem dump request for the 422-1A5 abend also captures the zSeries File System (zFS) address space in the local and remote dumps. Only one multisystem dump is requested per MODIFY BPXOINIT,RECOVER=LATCHES command regardless of the number of user tasks that are abnormally terminated. This sysplex-wide dump can aid in problem determination of a potential latch contention problem involving multiple address spaces and multiple systems.

If the latch contention is resolved by using the MODIFY BPXOINIT, RECOVER=LATCHES command, the BPXM056E message is deleted from the console and the following BPXM067I message displays:

```
BPXM067I UNIX SYSTEM SERVICES LATCH CONTENTION RESOLVED
```

If the command cannot resolve the latch contention within a reasonable time, message BPXM056E is deleted from the console and the following message displays to indicate that additional action might be needed:

```
BPXM057E UNIX SYSTEM SERVICES LATCH CONTENTION NOT RESOLVING
```

You can try entering the MODIFY BPXOINIT, RECOVER=LATCHES command again to see whether it resolves the latch contention. With these skills, you’ll be on the road to recovery, but if the latch contention continues, gather documentation and contact IBM Support for assistance.
Why the wait?

New VSAM RLS diagnostic commands show you what’s really going on

BY DOUGLAS LEHR, ROITY PRIETO, AND JASON LEE

With VSAM record-level sharing (RLS), your applications can access VSAM data sets — with full data integrity and serialization at the record level, not the control interval (CI) level — from any system in the sysplex. To provide this capability in z/OS, we start an SMSVSAM address space on each system to manage access to VSAM RLS data sets. Ensuring the correct integrity and serialization throughout the sysplex requires that the SMSVSAM address spaces constantly communicate with each other and with any CICS® regions or batch jobs that access VSAM RLS data sets.

Consider, however, what could happen if this communication breaks down in any way. A hang in RLS processing could potentially block access to any number of VSAM RLS data sets. And, if the hang occurs in a sysplex, with multiple systems in motion, you might find it that much more difficult to determine where the root problem lies.

Therefore, in recent releases of z/OS, we added two new operator commands — DIAG and DISPLAY,QUIESCE — to help you detect the location of hangs related to the SMSVSAM address space. While you already had some tools for diagnosing hangs, such as the command D GRS,C for showing enqueue contention, these tools did not always tell you everything you needed to know. Hangs can be complex, after all. For a hang related to VSAM RLS, the DIAG and DISPLAY,QUIESCE commands can help you determine which of your systems in the sysplex is experiencing the problem and which systems are impacted as a result.

Let’s look at how you might use the new commands.

Using the DIAG command

In VSAM RLS, latches provide system level serialization for shared control structures, ensuring that resources are either currently assigned to a process or are available for use. Because of their exclusive nature, latches tend to go into contention during VSAM RLS hangs.

Introduced in z/OS V1R8 (and retrofitted to earlier releases through APAR OA17556), the DIAG command provides users of VSAM RLS and transactional VSAM with a means of detecting a latch contention and its possible source. DIAG produces a console message that displays all latch contention, if any, for the system on which you enter the command.

Figure 1 shows the DIAG output for a system that is experiencing latch contention.

```plaintext
<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>TYPE</th>
<th>RESOURCE ID</th>
<th>JOB NAME</th>
<th>ASID</th>
<th>TASK</th>
<th>ASID</th>
<th>TASK</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMSVSAM</td>
<td>LATC</td>
<td>7BAD43B8</td>
<td>SMSVSAM</td>
<td>003A</td>
<td>008D5A48</td>
<td>003A</td>
<td>007F3000</td>
<td>00:22:09</td>
</tr>
<tr>
<td>IGW343I</td>
<td>LATC</td>
<td>07F1B1D0</td>
<td>SMSVSAM</td>
<td>003A</td>
<td>007F3000</td>
<td>003A</td>
<td>008D5A48</td>
<td>00:22:09</td>
</tr>
<tr>
<td>IGW343I</td>
<td>LATC</td>
<td>07F1B1D0</td>
<td>SMSVSAM</td>
<td>003A</td>
<td>008D64F8</td>
<td>003A</td>
<td>008D5A48</td>
<td>00:22:24</td>
</tr>
<tr>
<td>SMSVSAM</td>
<td>LATC</td>
<td>07F1B1D0</td>
<td>SMSVSAM</td>
<td>003A</td>
<td>008D6CF0</td>
<td>003A</td>
<td>008D5A48</td>
<td>00:23:30</td>
</tr>
</tbody>
</table>
```

In the DIAG output (Figure 1), RESOURCE ID is the address of the 8-byte area of virtual storage that represents a latch. For each held latch, the DIAG command also displays information about the job waiting for a resource (WAITER) and the process holding the resource (HOLDER), such as address space ID and TCB address. The value in ELAPSED TIME is the amount of time, in minutes, that the latch has been in contention.

Look carefully at Figure 1 and you will see a problem: two processes that are each holding a resource needed by the other.

Task1 (in blue) is holding a latch that Task2 (in red) needs. Conversely, Task2 is holding a latch for which Task1 is waiting.

In multiprocessing terms, we call this awkward entanglement a deadlock or (more poetically) a deadly embrace. To resolve it, you must cancel one of the two tasks involved.

That’s an extreme case, though. You might also find the DIAG command to be useful in certain scenarios that are described in z/OS DFSMSdfp Diagnosis, GY27-7618. For example, situations in which the holder of a latch is:

- Holding other threads and is a cross memory task from a client address space (such as a CICS transaction) that is also holding a latch and other threads. Here, you could simply cancel the holder, but what if this task or job is very important — more important than the waiters? If so, you might need to let the holder run to completion. Or, if the holder is processing critical work during the busy shift, you might choose to wait until the off-shift begins before canceling the job.

- Waiting for an ENQ that is held in another system. Here, you should...
consider entering the DIAG command on the other system to search for a possible problem there.

- Waiting for I/O or another system resource. Here, you should determine why the resource is not available or is slow to respond.
- An SMSVSAM (VSAM RLS) thread. Here, you might need to restart the SMSVSAM address space. Before doing so, however, you should consider the trade-offs between restarting the server immediately, versus waiting for the off-shift. Some work might be stalled, but is other, more important work still processing?

Using the DISPLAY, QUIESCE command

One key feature in VSAM RLS is the ability to shut down RLS access to a requested data set. This action, called quiescing the data set, allows non-RLS applications unhindered access to the files without interference from RLS applications.

To resume RLS access to the data set, you unquiesce it. You might, for example, temporarily quiesce a data set from CICS online transactions to allow it to be opened by nightly batch jobs, for example, or other non-CICS users, and later unquiesce it to resume VSAM RLS operations.

The DISPLAY, QUIESCE command has been around for several releases, but in z/OS V1R9, we enhanced it to provide more accurate data. We retrofitted the improved command to previous releases through APAR OA21101.

How does it work?

There are several ways to quiesce a VSAM RLS data set. Two common methods are:

- An explicit CICS request
- DFSMSdss™, when migrating or backing up a VSAM RLS data set.

In either case, CICS notifies all regions in the sysplex that use the data set. CICS then either relinquishes control of the data set immediately, or completes processing before giving up control of the data set.

After the batch processing or the data set migration is completed, the data sets are then unquiesced. Here, we notify CICS that the data sets are again ready to be accessed. This time around, however, we do not know which CICS regions currently need the data set or will need it in the future. To cover all the bases, we notify all CICS regions in the sysplex that are registered with a SMSVSAM address space and await their response. Even the lowly CICS test region must respond to an unquiesce event.

If any CICS regions fail to respond to the notification, whether because a CICS region still needs the data set, or that lowly test region does not have enough dispatch priority, the entire unquiesce event will hang.

If a system is running with a large number of CICS regions, it can be difficult to tell who responded to which events. To help ease root cause determination, we introduced the DISPLAY, QUIESCE command, which displays current quiesce and unquiesce events in progress for the system. If there is no activity, the DISPLAY output reflects this.

Figure 2 shows the DISPLAY, QUIESCE output for a system with VSAM RLS data sets.

DISPLAY, QUIESCE shows which data sets are being quiesced or unquiesced, along with the system on which request was entered, the time of the request, and how long it has been since the quiesce or unquiesce event began. DISPLAY, QUIESCE also shows the names and status of any subsystems that are involved in the quiesce or unquiesce event.

In Figure 2, notice that region C1AORP2 received notification and responded almost immediately, while C1AORP1 received notification but has failed to respond to the event for six hours. If a critical batch run needs to access this data set, and a CICS region has yet to respond to the event, the batch jobs will not be able to open the data set. Likewise, if a data set is to be unquiesced, but one or more CICS regions have not responded to the event, the data set will remain inaccessible to all CICS regions.

Some final thoughts

While DIAG and DISPLAY, QUIESCE both provide detailed information about hang scenarios, neither will take corrective action. It is up to you to decide the best course of action to resolve the hang. These commands can help you shorten the list of possible culprits and narrow the problem down to a single system or region in the sysplex.

Also, we continue to recommend that you dump SMSVSAM across the sysplex, along with any additional address spaces that show up in either display. Send the dump to IBM Customer Support before taking any action to resolve the hang.

Don’t let a wait in VSAM RLS processing leave you hanging. By using DIAG and DISPLAY, QUIESCE to help detect the location of hangs related to SMSVSAM, you might avoid the need for a sysplex-wide outage.

In multiprocessing terms, we call this awkward entanglement a deadlock or (more poetically) a deadly embrace.

Figure 2: Example of DISPLAY, QUIESCE output
Are your VSAM applications out to lunch?

Get them going with VSAM RLS

BY LARRY L. LAW, DEBBIE BENJAMIN-DECKERT, AND TERRI M. MENENDEZ

Are you feeling frustrated trying to improve the throughput of your mission-critical VSAM applications? Are your applications spending too much time seeking data? Does it sometimes feel like an episode of Mission Impossible?

If your problem with elapsed time is due to slow data retrievals, z/OS V1R7 VSAM record-level sharing (VSAM RLS) 64-bit buffering might be just what you are looking for.

A question of performance

Of all performance factors, how the buffers are allocated and managed for the VSAM data sets critically determines processing performance, affecting the I/O workload against all data sets that VSAM processes. Accessing data on DASD is a hundred times slower than accessing data from virtual storage or from a data space. Before z/OS V1R7, all VSAM RLS buffers resided in the two-gigabyte SMSVSAM data space. Users could tune the VSAM RLS buffering with the RLS_MAX_POOL_SIZE parmlib option to help determine how data-space buffers are reused, but the data space could still be too small for large applications to perform at optimum speed.

Beginning with z/OS V1R7, VSAM RLS makes the big leap over the two-gigabyte bar. It provides 64-bit-address buffering to keep the data and index buffers in virtual storage in the SMSVSAM address space.

“The Way We Were:” RLS before 64 bits

VSAM RLS ensures that no matter where you are calling from in the z/OS Parallel Sysplex®, you will get up-to-the-moment copies of records from a VSAM cluster. To accomplish this feat while meeting the performance requirements of the most critical production applications, RLS stores the data in three places in the storage hierarchy. A VSAM control interval (CI) resides on a DASD volume. The same CI can also reside in the coupling facility (CF) cache structure and — before RLS 64-bit buffering — in the two-gigabyte data space of the systems in the Parallel Sysplex.

When a transaction needs data, the shortest performance path is to find that data already in the system memory. RLS maintains buffer pools of data to facilitate quick transaction processing. An LRU task uses a “least recently used” (hence, LRU) strategy to determine which CI to retain in the data space and which is best replaced by newly incoming data. RLS can use most of the SMSVSAM data space, up to 1728MB, to store buffers. You are able to specify how much of that storage RLS is to use on each system in the Parallel Sysplex with the RLS_MAX_POOL_SIZE parmlib option. During workload spikes, RLS temporarily uses additional storage to hold buffers.

However, before RLS 64-bit buffering, no matter how much storage was requested or how intense the spike, the buffer pools could not exceed the two-gigabyte data space limit. Two gigabytes just weren’t enough for some large applications. Those transactions began to spend time waiting for a buffer to become available. During spikes, buffers that needed to remain in storage were reused on an emergency basis, impacting transaction performance as any CI had to be re-read. As more and more transactions were accessing data, the system processed more and more data at a slower rate.

“The New Deal:” 64-bit RLS

In z/OS V1R7, RLS went “above the bar”? In theory, up to two tera-pages (eight exabytes) of virtual storage in the SMSVSAM address space is now available for data buffering. The actual limit depends on the size of real storage. The data space is still there, but it is used to hold buffers temporarily. You can specify how much data space for RLS to use to keep buffers through the RLS_MAX_POOL_SIZE parmlib option and how much above-the-bar virtual storage to use through the RlsAboveTheBarMaxPoolSize parmlib option.
You specify whether buffers for a particular DATACLASS reside above or below the two-gigabyte limit. You can specify different buffer pool sizes for different systems. RLS maintains the buffer pools at the value you specify. As your data, workload, and system storage grow, the RLS buffer pools can grow too, returning improved performance where you need it most. In addition to making more storage available, RLS has made enhancements to manage the buffer pools more effectively.

For example, a monitor task now keeps watch on fluctuations in buffer pool sizes, ensuring a quick response to impending processing spikes. A new LRU task manages the above-the-bar pool. If update activity makes the buffers that reside above the bar no longer valid anywhere in the Parallel Sysplex, they are immediately available for reuse.

You can also specify that a portion of the buffer pool be permanently fixed in real storage. Using permanently fixed buffers means it takes less time to retrieve the data and hence help improve performance.

These enhancements allow RLS to keep data in the buffers longer. For widely accessed but seldom-updated data such as any key sequenced data set (KSDS) index CI, this translates to performance improvements. RLS can respond to processing spikes without prematurely refreshing buffers, reducing the need to re-read any CI when the spike subsides. Statistics found in the SMF and RMF™ records on the usage of the buffers can help you tailor your sysplex RLS buffer pool sizes to maximize the performance of your transactions.

“How to Succeed in Business without Really Trying:” Making use of 64-bit RLS to speed up your applications

Using VSAM RLS buffers above the bar for a data set only requires the following conditions:
- The system must be z/OS V1R7 or later.
- RlsAboveTheBarMaxPoolSize in the IGDSMSxx parmlib member must specify a number from 500MB through two terabytes for the system.

The data class for the data set specifies RlsAboveTheBar=YES.

The SETSMS, SET SMS=XX, and D SMS OPTIONS system commands also support the RlsAboveTheBarMaxPoolSize keyword, which means that you can dynamically change the RlsAboveTheBarMaxPoolSize values.

For detailed information about how to use 64-bit VSAM RLS, see z/OS DFSMS Using Data Sets, SC26-7410, at: publibz.boulder.ibm.com/epubs/pdf/dgt2d460.pdf.

Delivering the goods
An internal IBM performance study suggests that VSAM RLS 64-bit buffering allows for larger local buffer pools and increased buffer hits that result in better elapsed time.

To achieve the expected improvement in elapsed time, however, you must provide adequate RLS coupling facility cache structures to accommodate the larger local buffer pools. You must also provide adequate real storage to accommodate the larger buffer pools. Further, if your applications do not frequently retrieve existing records, 64-bit buffering might not be for you.

So, if you have the hardware, the required CF cache structures, and an adequate amount of real storage, what are you waiting for? Use VSAM RLS 64-bit buffering to exploit the hardware for improved throughput!

Acknowledgements
Thanks to Nelson W. Fincher of VSAM RLS Development for his contributions to this article.
RACF ghost hunters

BY RUSS HARDGROVE AND JUN OGATA

Is your RACF® database haunted by ghost profiles? Do you enter the SEARCH command, only to find (apparently) generic profiles that you are unable to change? You’re not alone. This problem has been haunting many security administrators, who have sought help from IBM support and the RACF-L forum.

Do you want to rid your database of these insubstantial pests? Is so, the RACF ghost hunters can help.

What happened is simply that a profile was created while generic command processing was turned off. Because of this, the profile is treated as a discrete profile although it looks like a generic profile.

How can you verify this?
In the SEARCH command output, check to see if there is a (G) after the profile name. Only true generic profiles are followed by a (G). For example:

```
GHOST.PROFILE1.*
GHOST.PROFILE2.* (G)
```

In this case, GHOST.PROFILE1.* is actually a discrete profile, even though it contains a generic character and so appears to be a generic profile. It’s one of the ghosts we’ve been hunting! On the other hand, GHOST.PROFILE2.* is a true generic profile.

How can you fix this?
To get rid of the ghosts, turn off generic profile checking and generic command processing for the class, and then delete the profile. If you use the example above and assume the profile is in the FACILITY class, the commands are:

```
SETROPTS NOGENERIC(FACILITY) NOGENCMD(FACILITY)
RDELETE FACILITY GHOST.PROFILE1.*
SETROPTS GENERIC(FACILITY)
```

Consider it open season on ghost generic profiles. Happy hunting!

Is that RACF in my Java?

Is that RACF in my Java? It could be, because there’s a new Java API for querying and administering RACF users and groups.

In z/OS V1R9, RACF introduced a set of Java interfaces that enable Java applications to access and manipulate user and group RACF data. This allows your installation to better leverage its existing security data.

This new Java interface:
- Can be run on or off the z/OS platform
- Is built on commonly used objects and interfaces in the Sun Java Servlet Development Kit (JSDK)
- Is easy to learn because it maps to other RACF user and group administration interfaces, such as the ADDUSER, ALTUSER, and CONNECT commands.

To get started using the new Java API, see the IBM Education Assistant module presentation Using Java to query or administer users or groups in RACF, which is available at:


Your feedback is welcome
You are encouraged to contact IBM with feedback on, or additional requirements for, this new interface.
security administrator and a wizard were in a bar one day. “I’m having all sorts of problems understanding RACLIST,” the security administrator was saying. “It’s like magic.

• Why do I bother to RACLIST stuff? It just seems like an extra step.
• I see the class descriptor table (CDT) entry for my new class states RACLIST=DISALLOWED. How can that be, when CICS needs it to be RACLISTed?
• What are the differences between the SETROPTS RACLIST command and the RACROUTE REQUEST=LIST macro?

Hey, you’re a wizard; maybe you can enlighten me about the magic behind RACLIST?”

The wizard, impressed that his friend was able to talk in bulleted list items, considered the problem. “There’s nothing truly magical about RACLIST,” he told the security administrator. “It just seems mysterious because you’re unaware of its history. Perhaps if we start at the beginning of RACLIST, we can slowly build a picture to help you understand it.”

What is RACLIST?
RACLIST is the placing of general resource profiles into system storage to satisfy security requests without requiring database I/O. Prior to RACLIST, all security requests went directly to the RACF database. This caused problems when there was a high volume of security requests, because all the requests had a single point of contention — accessing the RACF database. In addition, accessing the DASD on which the RACF database resided could be time consuming. When the speed of an application was crucial, as it would be for a Customer Information Control System (CICS) or an Information Management System (IMS™) application, the overhead of using RACF was impractical.

To address this problem, some clever person thought it would be smart to allow applications to place all the profiles of a particular class into memory. By doing this, a security request looks at the profiles in storage instead of going to the RACF database. This speeds up the response time for security requests, because applications avoid time-consuming I/O. This was the birth of RACLIST.

However, placing profiles of a particular class in memory meant the application would not pick up changes made to the RACF database for the class unless the application did something to refresh the data in storage.

“The need to refresh data in storage,” said the wizard, “is a very important thing to keep in mind. It’s important because it’s true for all forms of RACLIST. As we trace the history of RACLIST, you’ll see that, at every stage of its development, the need to refresh RACLISTed data has always been a concern.”

First, there was RACROUTE REQUEST=LIST
The initial approach of RACLIST was to place the profiles of a class in the application’s address space. The application would RACLIST the necessary classes using the RACROUTE REQUEST=LIST macro. This dramatically improved performance of the application, but the performance improvement came at a cost. If the class that was being RACLISTed had a large number of profiles, it needed a large amount of storage (all of it originally below the 16 MG line) to save that data in the application’s address space. This left less space for the application itself.

At this stage in the evolution of RACLIST, it was up to the application to create its own mechanism to detect when profiles in a RACLISTed class were updated, so that it could refresh the data in storage.

To illustrate this initial approach of RACLIST, the wizard waved his hands and, magically, a six-foot-tall hologram appeared over the security administrator’s head. The hologram, which rotated slowly in the air, illustrated how accessing profiles in an application’s address space compared with accessing the RACF database. Although it was a very impressive example of the wizard’s magic, reproducing a hologram of that size is beyond the resources of this newsletter (besides, this page is too small). Figure 1 is a rough approximation in two dimensions.
I/O required to access profiles in RACF database can be slow when there is a high volume of security requests.

I/O = slow

RACF DB

SETROPTS NORACLIST(<class-name>)

Once again, the wizard used his magic to conjure an illustrative hologram. This one compared accessing profiles in an application’s data space to accessing the RACF database. The best we can do here is a rough two-dimensional rendering of the wizard’s creation. We call it Figure 2.

“As long as we’re on the subject of the SETROPTS command,” said the wizard, “I should mention that it takes POSIT values (which are used to group similar classes) into account for any RACLIST processing. Whenever the SETROPTS command is entered with the RACLIST, RACLIST REFRESH, or NORACLIST parameters, the same action occurs for all related classes (in other words, those classes sharing the same POSIT value). It is very important to keep this in mind, especially when turning RACLIST off for a class. Because some applications might depend on certain classes being RACLISTed, don’t turn RACLIST off for a class unless you are certain you understand all applications that use that class (and all applications that use any class that has the same POSIT value).”

Although SETROPTS RACLIST improved upon the initial form of RACLIST by utilizing MVS data space technology, and providing commands and notification messages for easier administration, this support did nothing for those callers who were using the original RACROUTE REQUEST=LIST macro. The information they had RACLISTed was still in local private storage. What could be done?

SETROPTS RACLIST(<class-name>) REFRESH

If a class no longer needed to be RACLISTed, the security administrator could delete the data space into which the profiles had been copied by entering the SETROPTS command with the NORACLIST parameter.

Next, there was SETROPTS RACLIST

The next stage in the evolution of RACLIST was to place the profiles of a RACLISTed class into common storage to free the application’s address space. A secondary goal was to simplify the mechanism by which applications refreshed the stored profiles when changes were made to the RACF database.

A security administrator could now RACLIST a general resource class by entering the SETROPTS command with the RACLIST parameter:

SETROPTS RACLIST(<class-name>)

This places the profiles into common storage where multiple applications can access the same profiles. Originally, the profiles were in the common storage area (CSA) and the extended common storage area (ECSA), and later they were placed into an MVS dataspace (which is what is used today).

To simplify the process of refreshing a RACLISTed class, new messages informed the user when to refresh the profiles in storage because of changes made in the RACF database to a RACLISTed class. This way, when a user added, updated, or deleted a profile from the RACF database for a class that had been RACLISTed, he or she would be notified that the change would not be effective until the profiles in storage were refreshed.

To refresh a RACLISTed class, the security administrator could now enter the SETROPTS command with the RACLIST and REFRESH parameters.

Figure 1. Accessing profiles in an application’s address space versus accessing the RACF database

Figure 2. Accessing profiles in an MVS data space versus accessing the RACF database
Then came RACROUTE REQUEST=LIST,GLOBAL=YES

Seeing the benefits that the SETROPTS RACLIST command realized by leveraging MVS data space technology, the RACROUTE REQUEST=LIST macro was updated to take advantage of same thing. While calling applications could still use the local address space to store RACLISTed profiles, they now had the option of using an MVS data space. The GLOBAL parameter was added to the RACROUTE REQUEST=LIST macro.

However, if only the RACROUTE REQUEST=LIST,GLOBAL=YES macro is used instead of the SETROPTS RACLIST command, there are no messages to warn users when a profile in a RACLISTed class is updated.

Here are some other important differences between the RACROUTE REQUEST=LIST macro and the SETROPTS RACLIST command.

- Some classes are defined in the Class Descriptor Table (CDT) with the RACLIST=DISALLOWED option. This option is something of a misnomer, however, because these classes can still be RACLISTed. The RACLIST=DISALLOWED option simply means that the SETROPTS RACLIST command cannot be used to RACLIST the class. The class can, however, still be RACLISTed using the RACROUTE REQUEST=LIST macro.

- Unlike the SETROPTS NORACLIST command, RACROUTE REQUEST=LIST, ENVIR=DELETE,GLOBAL=YES will not delete the MVS data space. Instead, it merely removes the user’s access to the data space. Because the data space is not deleted, a subsequent RACROUTE REQUEST=LIST,GLOBAL=YES call connects to the existing data space (and possibly old profile data). This behavior enables the security administrator to maintain a consistent security environment until he or she changes it. The only way to delete the data space is to enter the SETR NORACLIST(<class>) command.

- Unlike the SETROPTS command, POSIT values are not taken into account when you RACLST a class using RACLST turned off, because it had not been RACLISTed to begin with.

Why RACLST?
RACLST helps provide much faster response time for security requests by taking a much quicker and faster path using system storage. For this reason, it is a general recommendation that you RACLST most general resource classes. Only classes that change often (for example, the TAPEVOL class), or whose changes need to be picked up immediately, should not be RACLISTed. Remember that only general resource classes can be RACLISTed. Because of this restriction, you cannot RACLIST the USER, GROUP, and DATASET classes.

“So,” said the wizard, “hopefully this conversation has helped demystify RACLST for you. There’s a lot more information we could cover, but, since it’s getting late, these other topics will have to wait for another day.” And with that, the wizard waved his hands and conjured two large margaritas out of thin air. Raising his glass to the system administrator, he offered a toast, “To RACLST!”

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.

Perhaps if we start at the beginning of RACLST, we can slowly build a picture to help you understand it.
Wrap your apps for console communication

Using the IBM JZOS Batch Toolkit for z/OS SDKs

BY ROBERT FAIRLIE

The IBM JZOS Batch Toolkit for z/OS SDKs is a package that simplifies running Java applications under z/OS. The basic function is included in IBM SDK for z/OS version 1.4.2 (31-bit) and versions 5.0 and 6.0 (31-bit and 64-bit), and consists of a batch launcher and a Java toolkit.

The batch launcher is an alternative to using BPXBATCH and BPXBATSL as a means to run Java programs in the MVS batch environment. The batch launcher enables you to:

- Run Java applications on z/OS seamlessly in an MVS batch job step or as a started task
- Configure the Java execution environment in a simple and flexible way
- Define access to data sets using JCL DD statements
- Send output directly to JES SYSOUT data sets with automatic codepage transcoding
- Pass condition codes between Java and non-Java job steps.

The toolkit is a Java Native Interface (JNI) library consisting of a Java archive (JAR) file and a native dynamic library. The toolkit enables Java applications to:

- Communicate with the MVS system console
- Read and write traditional MVS data sets from Java
- Control how MVS operator commands are handled.

In this article, we'll look at just a few of the classes available in the Java toolkit. Specifically, we'll show how a Java application can use classes in the Java toolkit to communicate with the system console, and respond to operator commands. Once we've shown how an application can do this, we'll then demonstrate how to use the batch launcher to run the application.

Echo—a simple Java application that performs I/O

Before we demonstrate an application that communicates with the MVS console, let's look at an application that performs simple I/O. Figure 1 shows a sample Java application, Echo. Echo simply has a loop that takes whatever is entered through stdin and writes it back out through stdout. Later, we'll show how to wrap this application so that input is redirected from stdin, and output is redirected to the MVS console using classes in the Java toolkit. For now, however, there is nothing MVS-console-like about it.

If you were to compile the Echo application and run it from the OMVS command line, it would direct your input strings right back to your monitor. Figure 2 illustrates execution of the Echo application.

The job name is unimportant when Echo is run from the command line, but it will be very useful later when the Echo application is wrapped by an application that redirects input from, and output to, the MVS console. In fact, let's take a look at that application now.

MvsConsoleWrapper—wraps a Java application to redirect I/O

The second sample Java application (MvsConsoleWrapper, shown in Figure 3) exploits JZOS console support, and can wrap any Java application that uses stdin and stdout. By doing so, the stdin and stdout are then redirected from and to the MVS console.

The MvsConsoleWrapper application uses the MvsConsole class from the Java toolkit to interface with the MVS console. The redirectSystemOut subroutine redirects output from the wrapped program to the MVS console using the MvsConsole.wto method. The redirectSystemIn subroutine uses the MvsConsole.startMvsCommandListener method to intercept MVS operator commands, while the MvsConsole.registerMvsCommandCallback method...
import java.io.*;
import java.lang.*;
import java.util.*;
import com.ibm.jzos.ZUtil;

public class Echo {
    public static void main(String[] args) throws IOException {
        BufferedReader br = null;
        int count = 0;
        boolean loop = true;
        boolean noPrompt = false;
        try {
            System.out.println("Echo: Enter your text at any time");
            System.out.println("Echo: jobname is <" +
                    ZUtil.getCurrentJobname() +
                    ">");
            while (true) {
                count++;
                String message = null;
                if (!noPrompt) {
                    br = new BufferedReader(new
                            InputStreamReader(System.in));
                    message = br.readLine();
                    if (message == null) {
                        noPrompt = true;
                    }
                } else {
                    try {
                        continue;
                    } catch (Exception e) {
                        e.printStackTrace();
                        break;
                    } // end catch
                }

                if (message != null) {
                    System.out.println("Echo: <" + message + ">");
                    message = null;
                } // end message != null

                if (count == 5) {
                    // escape clause after 5 iterations of while loop
                    System.out.println("Echo: count == 5. Breaking now.");
                    break;
                }
            } // end while
        } // end try
        catch (IOException e) {
            e.printStackTrace();
        } // end catch
    } // end invoke main
} // end Echo

Figure 1. Echo — a simple Java application that performs I/O

registers an MVS operator command callback (MvsCommandCallback interface) to enable customized response to operator commands. In this case, the handleModify method will enable the MVS operator to use the MODIFY command to direct input to the wrapped application.

Earlier, we saw how running the Echo application directed any input back to the monitor. Now let’s see what happens when we run it with the MvsConsoleWrapper application. To do this, we specify echo as an argument when invoking the MvsConsoleWrapper application (as shown in Figure 4).

Voilà! Though started from the very same OMVS command line, it’s off to the MVS console you go to communicate with the Echo application. The JZOS utility getCurrentJobname is used in Echo to spare you, gentle reader, the bother of figuring out your job name when at the MVS console.

Notice how the MVS operator MODIFY (F) command is used to communicate with the Echo application. That’s the JZOS handleModify method in action. If you were running this application, you could continue poking it with MODIFY commands until Echo ends naturally and returns, or you could run either the MVS CANCEL (C) or MVS STOP (P) command.

If you let the program reach its own conclusion, the execution stack unwinds gracefully. Echo returns, followed by MvsConsoleWrapper.

If you choose to stop MvsConsoleWrapper, JZOS provides the handleStop method for responding to the STOP command. Therefore, if you so desire, you can insert Java code into the handleStop method section in the MvsConsoleWrapper application to allow for a more detailed exit for your application. After the STOP command has been entered at the console and stopping procedures are completed, the command line returns to your OMVS session.

A nicer way to start a long-running Java application from the OMVS command line is to use the & character at the end of the command to have the task run in the background. Your application
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PipedInputStream;
import java.io.PipedOutputStream;
import java.io.PrintStream;
import java.io.Writer;
import java.lang.reflect.Method;
import com.ibm.jzos.MvsCommandCallback;
import com.ibm.jzos.MvsConsole;

public class MvsConsoleWrapper {
    // The main method accepts as arguments the name of the
    // wrapped class followed by its arguments.
    public static void main(String[] args) throws Exception {
        if (args.length < 1) {
            System.err.println("Usage: MvsConsoleWrapper <class> <args>...");
            System.exit(8);
        }
        redirectSystemOut();
        redirectSystemIn();
        invokeMain(args);
    }

    static void redirectSystemOut() throws Exception {
        PipedOutputStream pos = new PipedOutputStream();
        PrintStream ps = new PrintStream(pos);
        PipedInputStream pis = new PipedInputStream(pos);
        final BufferedReader reader = new BufferedReader(new
        InputStreamReader(pis));
        new Thread() {
            public void run() {
                try {
                    String line = null;
                    while ((line = reader.readLine()) != null) {
                        MvsConsole.wto(line,
                                    MvsConsole.ROUTCODE_SYSPROG_INFORMATION,
                                    MvsConsole.DESC_JOB_STATUS);
                    }  // end while
                }  // end try
                catch(Exception ioe) {
                    // Pipe breaks when shutting down; ignore
                }  // end catch
            } //end run()
        }.start();
        System.setOut(ps);
    } // end redirectSystemOut()

    static void redirectSystemIn() throws Exception {
        // This starts the MvsConsole listener if it's
        // not already started
        // (by the JZOS Batch Launcher)
        if (!MvsConsole.isListening()) {
            MvsConsole.startMvsCommandListener();
        }
        PipedOutputStream pos = new PipedOutputStream();
        // use default file.encoding
        final Writer writer = new OutputStreamWriter(pos);
        MvsConsole.registerMvsCommandCallback(
        new MvsCommandCallback() {
            public void handleStart(String parms) {
            }
            public void handleModify(String cmd) {
                try {
                    writer.write(cmd + "n");
                    writer.flush();
                } catch (IOException ioe) {
                    // Pipe breaks when shutting down; ignore
                } // end catch
            } //end run()
            .start();
            System.setIn(pis);
        } // end redirectSystemOut()

        static void redirectSystemIn() throws Exception {
            // This starts the MvsConsole listener if it's
            // not already started
            // (by the JZOS Batch Launcher)
            if (!MvsConsole.isListening()) {
                MvsConsole.startMvsCommandListener();
            }
            PipedOutputStream pos = new PipedOutputStream();
            // use default file.encoding
            final Writer writer = new OutputStreamWriter(pos);
            MvsConsole.registerMvsCommandCallback(
            new MvsCommandCallback() {
                public void handleStart(String parms) {
                }
                public void handleModify(String cmd) {
                    try {
                        writer.write(cmd + "n");
                        writer.flush();
                    } catch (IOException ioe) {
                        ioe.printStackTrace();
                    } // end catch
                } //end run()
            } //end run()
            .start();
            System.setOut(ps);
        } // end redirectSystemOut()
    }

    static void invokeMain(String[] args) throws Exception {
        String[] mainArgs = new String[args.length-1];
        System.arraycopy(args,1,mainArgs, 0, mainArgs.length);
        Class mainClass = Class.forName(args[0]);
        Method mainMethod = mainClass.getMethod("main", new
        Class[]{});
        mainMethod.invoke(mainClass, new Object[]{});
    } // end invokeMain

    public class MvsConsoleWrapper {
        // The main method accepts as arguments the name of the
        // wrapped class followed by its arguments.
        public static void main(String[] args) throws Exception {
            if (args.length < 1) {
                System.err.println("Usage: MvsConsoleWrapper <class> <args>...");
                System.exit(8);
            }
            redirectSystemOut();
            redirectSystemIn();
            invokeMain(args);
        }

        static void redirectSystemOut() throws Exception {
            PipedOutputStream pos = new PipedOutputStream();
            PrintStream ps = new PrintStream(pos);
            PipedInputStream pis = new PipedInputStream(pos);
            final BufferedReader reader = new BufferedReader(new
            InputStreamReader(pis));
            new Thread() {
                public void run() {
                    try {
                        String line = null;
                        while ((line = reader.readLine()) != null) {
                            MvsConsole.wto(line,
                                        MvsConsole.ROUTCODE_SYSPROG_INFORMATION,
                                        MvsConsole.DESC_JOB_STATUS);
                        }  // end while
                    }  // end try
                    catch(Exception ioe) {
                        // Pipe breaks when shutting down; ignore
                    }  // end catch
                } //end run()
            }.start();
            System.setOut(ps);
        } // end redirectSystemOut()

        static void redirectSystemIn() throws Exception {
            // This starts the MvsConsole listener if it's
            // not already started
            // (by the JZOS Batch Launcher)
            if (!MvsConsole.isListening()) {
                MvsConsole.startMvsCommandListener();
            }
            PipedOutputStream pos = new PipedOutputStream();
            // use default file.encoding
            final Writer writer = new OutputStreamWriter(pos);
            MvsConsole.registerMvsCommandCallback(
            new MvsCommandCallback() {
                public void handleStart(String parms) {
                }
                public void handleModify(String cmd) {
                    try {
                        writer.write(cmd + "n");
                        writer.flush();
                    } catch (IOException ioe) {
                        ioe.printStackTrace();
                    } // end catch
                } //end run()
            } //end run()
            .start();
            System.setIn(pis);
        } // end redirectSystemIn

        static void invokeMain(String[] args) throws Exception {
            String[] mainArgs = new String[args.length-1];
            System.arraycopy(args,1,mainArgs, 0, mainArgs.length);
            Class mainClass = Class.forName(args[0]);
            Method mainMethod = mainClass.getMethod("main", new
            Class[]{});
            mainMethod.invoke(mainClass, new Object[]{});
        } // end invokeMain
    }
}

Figure 2. Running the Echo application

> java Echo
Echo: Enter your text at any time
Echo: jobname is <G0210184>
1
Echo: <1>
2
Echo: <2>
IBM
Echo: <IBM>
I never said this is rocket science
Echo: <I never said this is rocket science>
z/OS RULES!
Echo: <z/OS RULES!>
Echo: count == 5. Breaking now.

Figure 3. MvsConsoleWrapper — can be wrapped around any Java application that performs I/O
will run just fine with the console interface, and you still have the command line for other tasks:

```bash
> java MvsConsoleWrapper Echo &
> java MvsConsoleWrapper Echo
```

### Running Java applications from a batch job

The next logical step for a long-running Java application is to run it as an MVS batch job. This is where the batch launcher comes in. The batch launcher is an MVS batch program that configures and launches the Java Virtual Machine (JVM). In the JCL shown in Figure 5, the Java class called by the batch launcher is MvsConsoleWrapper. The Echo class is passed as an argument with the JZOS_MAIN_ARGS variable in the STDENV DD statement.

Now we have a long-running Java application with console interaction running as a batch job. The best of all worlds!

### The grand finale

The batch launcher does not require your Java application to make use of any of the toolkit APIs. On the Java development team, we have used the batch launcher to run many Java tasks including the execution of JUNIT test suites for other Java functional testing. In addition, the technique of Java application wrapping demonstrated in this article was used to successfully deploy the IBM Encryption Key Manager on z/OS as a started task.

I hope this small sampling of the toolkit and the batch launcher has piqued your interest in the IBM JZOS Batch Toolkit capability. If it has, you can refer to the following Web sites for documentation and additional samples:

- [alphaworks.ibm.com/tech/zosjavabatchtk](http://alphaworks.ibm.com/tech/zosjavabatchtk)

By visiting these sites, you’ll find that the IBM JZOS Batch Toolkit capability provides many handy functions that enable a Java programmer to tap the power of z/OS.
Trash your assumptions!

*The latest advice on tuning your Java heap*

**BY MIKE YOUNG AND KEN IRWIN**

Garbage collection (GC) is so misunderstood. Countless sources of advice differ over such matters as frequent GC, long GC, compaction, fragmentation, and other terms that fill system administrators with dread. We in Java Level 2 want to set the record straight. We intend to dispel many of the rumors and myths you might have heard about heap tuning in the past.

This article provides a set of standard recommendations regarding tuning of the z/OS Java 1.4.2 heap. Version 1.4.2 of the Java Virtual Machine (JVM) is used in current IBM CICS and IBM DB2® releases as well versions 5.1 and 6.0 of IBM WebSphere® Application Server. The next release of Java, version 5.0, has a completely different garbage collector and completely different tuning. We’re going to leave that for a future article. On to the recommendations.

**Update your JVM**
Recent service refreshes of the JVM have included numerous improvements in stability and performance in JVM storage management and the garbage collector. There are specific updates and corrections to heap expansion, improvements to heap marking, and support for large object tuning. If you haven’t updated recently, check your Java PTF level or issue the command `java -version` at a UNIX System Services prompt and check the results. Informational APAR II13519 indicates the most up-to-date service refresh available. For the recommendations in this article, we suggest service refresh level SR9A or later.

**Verbose GC (with st_verify!) is your friend**
If you can accommodate the slight overhead, run your application (even in production) with the garbage collection tracing option `-verbosegc`. This produces detailed information about every garbage collection cycle that takes place. If you have already turned on this option, consider adding the JVM property `ibm.dg.trc.print=st_verify`. This additional property adds no noticeable overhead, but provides useful information about compaction and pinned classes. When setting this property on the Java command line, add `-D` to the beginning: `-Dibm.dg.trc.print=st_verify`.

**Minimum and maximum should be different**
The number one myth we hear in Level 2 is that you are supposed to run the JVM with a static heap size (one where `-Xms` equals `-Xmx`). This is simply not true. A long time ago (well before version 1.4.2) large, long-running applications could benefit from setting the minimum and maximum to the same value. Today, the Java team recommends using different values. So how do you know what values to specify?

**Setting the maximum heap size (-Xmx)**
This one is easy. Set your maximum heap size to a value that at least holds your entire application at its highest load. By default, the maximum heap size for the JVM is set to 64 MB; middleware applications such as WebSphere Application Server and CICS have their own defaults. Setting a value that is too small will cause java.lang.OutOfMemory errors. Setting a value that is too large will, at best, waste space within the Java heap. At worst, it will cause address-space-related memory errors because the Java heap is allocated from Language Environment (LE) heap storage (subpool 2) as a single, contiguous area based on the maximum heap size (-Xmx) requested.

**Setting the minimum heap size (-Xms)**
Set your minimum heap size to the smallest amount of memory your application needs to start. By default, the minimum heap size for the JVM is set to 1 MB. How can you tell what your application needs? After you start your application, but before any workload begins, look at your verbose GC log. The size of your heap at this point is the value you should use.
Java tries to keep your heap at the most efficient size throughout the run time of your application. As your workload increases, the heap expands. As workload decreases, the heap contracts. Maintaining an efficient size is very important to avoid fragmentation.

When `Xms` is set to the same value as `Xmx`, you prevent the JVM from managing its heap as the designers intended. Although the JVM might not need to perform garbage collection as frequently, when it does, the time spent in GC can be quite long. When GC takes a long time, the user can seem to be hung. Furthermore, if the JVM isn’t allowed to compact the heap as an application is being loaded, certain objects that are stored in fixed locations in the heap can cause fragmentation and potential OutOfMemory failures later in execution.

**Large objects and the LO ratio**

Large Java objects are becoming more and more common as applications start working with complete sets of business data, creating PDFs, parsing XML, and doing things no one considered when Java was first developed. The designers of the JVM considered anything over 64 KB in size a “large object.” However, it is quite common to see applications creating objects that are 2, 10, and even 40 MB in size.

These super-large objects cause problems for the normal Java heap. Even the slightest bit of fragmentation can cause allocation of a large object to fail; the larger an object gets, the more likely fragmentation will cause a failure.

Never fear, the JVM has a solution! There is an area in every Java heap called the large object area (LOA). This is a special reserved space that small objects aren’t allowed to use. Unfortunately, the default size of this area isn’t large enough to accommodate all application environments.

You can tell if your application is exceeding the LOA by checking the action code in verbose gc tracing. Action=2 means that the JVM attempted to use the LOA but was unsuccessful. If you have many of these action=2 allocation failures, your LOA is too small.

**Customizing the LOA**

The JVM now has an option to configure the size of the LOA. (Before you use this option, be sure to update to the latest service refresh. There have been some recent fixes to this feature.) To configure the size of this area, use the JVM option `-XloratioN` where `N` is a decimal number less than one but greater than zero representing the portion of the heap devoted to the LOA. For example, we commonly recommend `-Xloratio0.3` as a starting point for applications with objects 10 MB in size. The default size of the LOA is 0.05.

Tuning this value is a repetitive affair. To avoid problems with small objects, you should avoid reserving too much of your heap. Instead, slowly increase the value in 0.05 increments and restart Java to pick up the change.

When you use the `-XloratioN` option to manually configure the size of the LOA, you will see a new field in verbose GC representing the used space in both the normal area and the LOA.

---

![Figure 1. Sample output from verbose gc tracing](image-url)
heap layout:
(151332640/375762688)
(159412144/161041152)

The first set of numbers displays the used/total values for the normal heap area; the second set provides the same values for the LOA.

As you increase the LO ratio, you will notice a reduction in action=2 failures. The goal is to replace as many as you can with action=1 while keeping the frequency of GC about the same. (A rule of thumb is that as the action code increases, the more time is spent within the GC cycle.)

If you increase the LO ratio too far, you will notice frequent GCs due to exhaustion of the normal heap area. Additionally, you might notice a large amount of the LOA sitting unused. In the preceding example, the LOA is practically empty and the LO ratio could probably be reduced to give more space to the normal heap.

Clusters
Remember when we told you to activate ibm.dg.trc.print=st_verify in the beginning of this article? Configuring your clusters is where this setting comes in handy. The JVM has two primary clusters: the pinned object cluster (p) and the class cluster (k). Experience has shown that in many environments, the greatest benefit can be made by tuning the k cluster. Fortunately, adjusting the k cluster is easy.

Load and run your application with a healthy workload and then examine your verbose GC logs for lines like those in the preceding example. These are the st_verify output. Set your k cluster to about 10% more than the value of classes in the first line. In the preceding example, we would recommend -Xk1045. The settings for clusters, -XkNNNN and -Xp, are typically most needed with large applications. WebSphere Application Server often needs a setting of -Xk in the thousands. (Experience has shown a good starting point is -Xk6000.)

Conclusion (and further reading)
Because every application is different, the GC tuning options used for each environment must be evaluated individually. No single document can provide perfect tuning for everyone. Plenty of additional options that are not discussed here might be needed on your system. You might even need to think about the design of your applications. The final word in every case should always be the Java 1.4.2 Diagnostics Guide available at ibm.com/developerworks/java/jdk/diagnosis/142.htm. Chapter 2 deals exclusively with the garbage collector, and Chapter 31 discusses the GC diagnostic options you can use.

We intend to dispel many of the rumors and myths you might have heard about heap tuning.
Game plan for success

Working with our vendors to win

BY NICOLE M. FAGEN AND AUTUMN A. HOUSEKNECHT

To deliver innovative products and outstanding client support, IBM creates many opportunities for interaction with independent software vendors (ISVs). Our interactions with ISVs cover a broad spectrum of activities including:

• Discussion of future changes
• Planning for future changes
• Testing ISV products on the latest release of z/OS
• Understanding requirements to request enhancements to z/OS
• Resolving problems for our mutual clients.

A healthy partnership
IBM works with ISVs to ensure a successful product integration. IBM ISV Technical Services and Support programs are available to any member of PartnerWorld® (ibm.com/partnerworld) and include a broad range of programs designed to prevent problems for our mutual clients.

Technical disclosure meetings
Biannually, we hold technical disclosure meetings with ISVs to discuss changes to IBM products. These meetings give ISVs the opportunity to learn about the latest enhancements in z/OS directly from the people who make it happen: our product designers and developers. We share certain information about z/OS so that ISVs can make appropriate plans for their products to support the future environment. In addition, we make certain that ISVs have the information and resources they need to successfully develop new products.

Vendor Participation Program
The IBM Vendor Participation Program (VPP) is a program designed to enable timely and direct notifications to ISVs of changes occurring within selected z/OS components that might affect their products. This program was created by IBM with the sole purpose of reducing the number of problems that might result from the interaction between ISV and IBM products. The program allows us to proactively address and resolve these issues before the products ship.

ISVs can subscribe to this service by registering an interest in specific parts of z/OS. For example, if an ISV application interacts with part XYZ of z/OS, the ISV can register an interest in that part. When we make a change to part XYZ that can impact the ISV, we notify the ISV of that change early in the process. This notification enables ISVs to make and test the changes in their products to avoid the problems being discovered by a z/OS customer.

Additionally, when ISVs register their interest in parts, it helps us to better understand how ISV products interact with z/OS. As a result, we gain insight into how the changes we make might impact ISV products, arming us with the information we need to decide if we should consider other design options or if a given change is necessary.

IBM innovates…in Dallas!
IBM also provides ISVs with the opportunity to test their products on z/OS before the general availability of a new release. The IBM Innovation Center in Dallas, Texas is the facility where the magic happens! This test environment provides ISVs with an opportunity to ensure their existing products integrate smoothly with the new release of z/OS. It also serves as a place for ISVs to test upgraded products or new products that use new z/OS functions.

Where can I find out more?
If you’re an ISV or if you know one, please do get involved with IBM ISV Technical Services and Support programs by visiting:

• IBM Destination z (ibm.com/systems/z/destination)
• System z™ ISV Support (ibm.com/systems/z/solutions/isv).

Prime-time problem solving
Collaboration between IBM and ISVs doesn’t stop after the z/OS release and ISV products are generally available. Enter the IBM z/OS Support Center, which has one goal: to ensure our clients’ success by resolving problems as quickly as possible! When a problem involves both an ISV and IBM, we work closely with our client and the ISV as a team to identify the cause of the problem.
The truly great debuggers and problem solvers embrace and attack client problems with all the energy, vigor, and excitement of the final minutes of an amazing sports event. If you thought the Cowboys / Bills Monday Night Football finale was exciting last season, you should be a fly on the wall at the IBM z/OS Support Center during a 2:00 am critical situation conference call! The great ones always play to win!

Building the team

The client is like a coach; it is up to the client to select the players. Will the players be only IBM, only ISVs, or a mixture of IBM and ISVs? Will there be two independent teams with one coach? As a part of building the team, these are the decisions that our client/coach must make. It is our client’s decision to determine if all the players will play in a particular game. Further, it is the coach that establishes communications between the players whoever they are. It is a win-win situation when our client remains present for all exchanges between IBM and the ISV. Having all of the participants present makes it easy to ensure that everyone is kept informed of the progress of the problem analysis and the solution to follow. We wouldn’t want our coach to stop coaching in the middle of a game, nor would we want one player running some random pattern that jeopardizes the game plan!

Warming up

After the team is established, it’s time to warm up. We begin with an exchange of information, starting with a problem summary that offers a technical description of the issue and an understanding of the impact of the problem on the client’s business. The current status of the problem is crucial: is the problem still occurring or has it cleared? If the problem is currently occurring, the team collects the appropriate documentation, reviews it, and takes the necessary actions. If the problem has already cleared, we try to understand the details of the documentation that the team has collected to initiate the debugging effort.

Let’s huddle

After warm-up, the team is ready to huddle. Here, every team member tries to eliminate extraneous potential causes of the problem so that only possible root causes remain. The huddle is necessary to ensure that the investigation of the problem is accurate and efficient.

Everyone benefits when this huddle is held immediately. It is also important to understand that during the course of the “game,” the players can change and additional huddles could be required. For example, IBM might need to engage other IBM components or eliminate some components from the discussion as new information is uncovered. And the same is true of the ISV team members. The huddle is a natural part of the problem determination process.

Sharing the details of the analysis and the current understanding of the problem allows team members avoid duplicating efforts.

Knowing your role in the game plan

The players who remain in the game after the huddle are those responsible for chasing down potential root causes of the problem, workarounds, or fixes. Often this means that each player needs to understand his or her individual part in the overall play to find additional information for the team to analyze as progress is made toward the “end zone.” The new findings might indicate that some players gained significant yardage (uncovered a new possibility that needs to be investigated), been sidelined (eliminated themselves from the game entirely), or have scored a touchdown (identified the root cause).

Looking forward to win

IBM is dedicated to every client’s success. As a part of our efforts to live true to this core value, we work very closely with ISVs to do our best to ensure that our products work in conjunction seamlessly and error-free. We value and support our close relationship with ISVs. IBM will continue to work hand-in-hand with ISVs to ensure our clients’ success.

IBM is dedicated to every client’s success.
It’s the zFavorites for System z credit card CD! You’re gonna love this! It has all sorts of helpful Web links, like those for:

- Hardcopy
- Operating systems
- Software
- Language and tools
- ISV development and applications
- Product documentation
- Marketing information
- Education
- Support
- Links to FREE downloads
- IBM Redbooks® sampler
- WebSphere Application Server
- XML

To use the CD insert it in any standard CD device, and it should start automatically. If it does not, click **Start > Run**, and then type `x:\index.htm` (where `x` is your CD drive letter) and press **Enter**.

Additional copies of zFavorites CD (GK3T-4331-12) are separately orderable.

zFavorites:

The secret is out!

What do we tell your ISVs about enablement assistance to System z?

BY ANDREW D. SCHMIDT AND SUSAN M. TUCZYNSKI

Did you know that if you search the IBM Business Partner Application Showcase (ibm.com/software/showcase), you can find thousands of solutions from our System z independent software vendors (ISVs)? Moreover, these solutions are spread across all industries (automotive, financial, government, and the like) as well as solution types (applications, middleware, infrastructure support, and more), and all support the System z platform. Yet, sometimes we in the System z ISV enablement team hear from our System z customers that ISVs do not know how to enlist IBM’s support to help enable their products on the System z platform. And that’s why you, as a System z customer or developer, should know about our PartnerWorld program and options. Let us help you to get the most out of your ISVs by getting them in touch with the System z information they need.

IBM PartnerWorld
The first thing you should do is point your ISVs to the IBM PartnerWorld program (ibm.com/partnerworld). PartnerWorld is an IBM-wide program that allows Business Partners access to solutions, offerings, and skills that will help them win in the marketplace; it’s the first step in working with our System z team. Although PartnerWorld membership itself is offered at no charge and contains a lot of no-fee resources, some specific programs within PartnerWorld are fee-based.

Support for IBM Business Partners
Within PartnerWorld, we in the System z ISV support team have created two specific “landing pages” that our ISVs can use to gain access to our programs for z/OS.

The higher level is the “System z - Benefits, resources, and support for IBM Business Partners” page (ibm.com/partnerworld/systemz). This page is a master collection of resources ranging from technical information to training, enablement facilities for hardware resources, white papers, and educational opportunities, ISV Advocate Program details, and so on. It’s a wealth of knowledge spanning the many operating systems that System z supports, not just z/OS.

Also new in 2007 is the “z/OS - Resources and support for IBM Business Partners” Web page (ibm.com/partnerworld/zos) that, as the name suggests, provides focus strictly on z/OS items such as technical z/OS education for our ISV developers, Destination z information (for the ISV focus), road maps for solution enablement, and the like.

IBM Innovation Centers
Key to the technical enablement activities of which ISVs can avail themselves are the IBM Innovation Centers for Business Partners. They are worldwide in scope. Although not every center supports every flavor of IBM hardware and operating systems, these centers can offer partners from virtually any geography access to the appropriate people and skills to support ISV porting and enabling activities. They provide education, consultation, system access, and more. Some of these offerings are available on site, and some are provided through remote access.
**System z for ISVs initiative**

A relatively new program introduced in 2006 is the “System z for ISVs” initiative. This is a no-charge, worldwide program that specifically supports ISVs that are interested in porting their z/OS-based applications to either WebSphere Application Server or DB2 on System z. The program provides ISVs with technical consulting with IBM IT architects and remote System z access to develop, port, and test their application. After the actual enablement, we provide the ISVs with follow-on sales and marketing resource support (such as Global Solution Directory listings, IBM Sales Connections for System z, membership in the PartnerWorld Industry Networks program, advertising discounts, lead generation tools, support for direct mail campaigns, and so on)—all to help increase the visibility of the ISV’s product in the marketplace.

**Continued support for ISVs**

But of course, even if the ISV’s application doesn’t qualify specifically for the “System z for ISVs” initiative, other portions of our PartnerWorld programs are available, just as they were before the initiative was created in 2006.

So, once the ISV is on the platform, IBM continues to provide support with other marketing and technical programs as well. Some of these are confidential in nature and require specific confidentiality agreements to be put in place. So it’s well worth it for your ISV to enroll in PartnerWorld and for you to introduce them and yourself to the services and offerings of our PartnerWorld team! We provide assistance through a special e-mail ID (zisvq@us.ibm.com) that any ISV can use for additional guidance.

**Bottom line?**

Our teams in Poughkeepsie, other worldwide System z centers, and the Innovation Centers all work with ISVs to get them to the correct program within IBM to accomplish their goals. And, with that said, we circle back to our original premise of this article: to explain what it means to you — satisfying your own needs as a System z customer or developer by providing your ISV with the right information to exploit the System z platform.
In 1994, customers were telling IBM hardware and software designers:

- We want higher availability
- We want protection against unplanned outages, and fewer planned outages
- We want to pay less for MIPS
- We want our server growth to match the pace of our business growth as we merge with other companies.

By that time, we had already reduced the APAR rate — thanks to much-improved programming tools and techniques. Higher availability continued to be an important focus for us, however; we still needed a means of ensuring that changes to your environment would not render an existing application unworkable. We believed that a component designed for running simple transactions and batch in 1990, for example, should work as well (and hopefully better) on a ten-times faster server running WebSphere Application Server, z/OS UNIX System Services, and so on, today.

In revisiting our list of customer requirements, we now find that:

- Higher availability is achieved through the automatic selection of an available server to handle the work.
- Price per performance is enhanced by allowing applications to run on more, but possibly slower servers, including what was (back in 1994) the “new” CMOS server. Of course, server speed has increased so much since the introduction of Parallel Sysplex that price per performance is vastly improved. Further, MSU-based variable workload license charges (VWLC) pricing renders it even less of an issue.
- Scalability is attained by allowing you to cluster up to 32 servers.

All of this was designed with the overall goal of minimizing the need for your installation to change its applications.

How availability is maintained
You — and your end users and business applications — can view the entire Parallel Sysplex cluster as a single logical resource. Just as work can be dynamically distributed across the individual CPs within a single symmetric multi-processor (SMP) server, so, too, can work be directed to any node in a Parallel Sysplex cluster with available capacity. This approach avoids the need to partition data or applications among individual nodes in the cluster, or to replicate databases across multiple servers. If one node becomes unavailable, the Parallel Sysplex makes it possible to redistribute workload dynamically to the remaining resources.

With Parallel Sysplex, we introduced the concept of rolling IPLs, that is, the recycling of one logical partition (LPAR) at a time to apply maintenance or upgrades, while keeping production applications available on other logical partitions, which remain active.

By removing single points of failures in the environment, we help to maintain availability for your business applications.

Introducing the coupling facility
The heart of the Parallel Sysplex cluster is the coupling facility, or CF, for short. Although the coupling facility control code is loaded from the server microcode, it behaves more like an operating system: It runs in an LPAR, it executes transactions (CF requests), and it manages resources, such as CF links and storage.
Because a CF runs in an LPAR, it can reside either on a stand-alone server by itself, or as one LPAR on a server with other LPARs containing z/OS systems.

The storage in a CF is formatted to contain various CF structures, each of which is designed for a specific purpose:

- Cache structures provide buffer coherency, as used by DB2 buffer pools.
- Lock structures provide inter-system locking, such as that used by GRS and the integrated resource lock manager (IRLM).
- List structures allow for communication and passing data, for example, by z/OS components such as XCF, JES2, and system logger.

By sharing resources, a CF can help simplify system management by reducing the need for extra, specialized hardware.

**How serialization is managed**

In a Parallel Sysplex, serialization is provided by IRLM for both DB2 and IMS. IRLM can track locks on records within an LPAR.

IRLM manages serialization across images by hashing to a location within a lock structure in the CF to insert lock information. If there is contention, the CF notifies the different instances of IRLM and together they manage the lock in conflict to determine whether it is a real conflict (the transaction waits until lock

To obtain even higher availability, you can *duplex* the structures by maintaining identical copies of the structures on different CFs.

**Configuring a CF**

A CF can use a general purpose CP, but it is usually configured to use a special purpose CP called an Internal Coupling Facility (ICF). How you configure a CF is usually determined by your installation’s business requirements. If redundancy is a priority, for example, IBM recommends that you configure two CFs for each Parallel Sysplex. A specific CF structure can reside in one CF or the other, and each CF can back up the other.

To obtain even higher availability, you can *duplex* the structures by maintaining identical copies of the structures on different CFs. Of course, you must balance the availability and configuration benefits of CF duplexing with the performance cost of doing it.

**A good team player**

A CF shares resources with the rest of the sysplex in these ways and more:

- Uses a GRS star for file serialization performance
- Uses the OPERLOG data set for merged system logs
- Shares tape drives between system images

The use of the cache and lock structures within the CF allows the multiple transactions to perform direct read/write access to the same database at the same time.

**Example of a CF in use**

For transactions running in parallel, the main concerns are how to provide data integrity and serialization. Data integrity is an issue because database managers (DB2, IMS, and so on) must maintain an in-storage buffer pool (a record that is read once will be read again at some point). Referencing the record from storage can save response time and CPU time compared to an I/O operation from disk.

When multiple DB2 subsystems exist, however, with each having read a page and one having made a change to it, how do we ensure that another DB2 subsystem copy of the data reflects the change?

For this, we need the CF. The first time a DB2 subsystem reads a page, it registers interest with the CF. Basically, it tells the CF, “Hey, I need to know if anyone else changes this page.” Thereafter, each time a DB2 subsystem re-uses a page in its local buffer pool, it checks the page validity through a bit map in the Hardware System Area (HSA). If the bit indicates that the page is still valid, the DB2 subsystem can safely reference the page again.

Further, each time a DB2 subsystem updates a page, it sends a copy of the new page to the CF, which determines whether another DB2 subsystem has registered interest in the page. If so, the CF directs that DB2 subsystem’s server to flip the HSA bit. If a DB2 subsystem needs to re-read a page and finds that the bit was flipped, it gets an updated version from the CF (avoiding a sequential I/O) or from disk.

As you might have guessed, other database managers in the Parallel Sysplex (such as IMS) do something similar to provide data consistency.

**Find out more**

Need a good read on the subject of coupling facilities and other aspects of Parallel Sysplex technology? See these links:

- Our home on the Web: [ibm.com](http://ibm.com)/systems/z/psos

In addition, see IBM publications for information about setting up and using a Parallel Sysplex:

- [z/OS MVS Setting Up a Sysplex](http://www.redbooks.ibm.com/redbooks/pdfs/sg246373.pdf), SA22-7625
- IBM Redbooks: [www.redbooks.ibm.com](http://www.redbooks.ibm.com)
Roadblocks holding you back?

WLM helps dispatch blocked workloads

BY JUERGEN SCHNEIDER, KARL-HANS HOLDER, AND ARMIN BUECHLER

Generally, self-optimization means self-tuning. That is, the ability of the system to recognize that some function can help improve how it works and take the appropriate action. In z/OS V1R9, Workload Manager (WLM) continues to help address the requirements of blocked workloads. If your system runs for extended periods at 100% processor utilization, blocked workloads can benefit from new function run through parameters in the IEAOPTxx parmlib member. With the new support, blocked workloads can proceed even when the system is 100% active for an extended period.

During periods of 100% processor utilization, discretionary workloads (workloads that your installation defines with a lower dispatch priority) might not be dispatched. These discretionary workloads can hold serially reusable resources that other workloads require, thereby blocking the progress of workloads with a higher dispatch priority. In z/OS V1R9, you can specify that work in a given address space or enclave that is ready to run, but that does not get processor service within a certain period of time (starved work), is temporarily promoted to a higher dispatch priority.

Various enhancements such as improved enqueue and lock promotion already exist. Nevertheless, all these methods require resource monitors to recognize the contention situation and signal it to WLM. This is not always possible, because the resource managers require comprehensive instrumentation. It is possible to have a priority inversion loop, which means one resource holder blocks the work that is waiting for another (different) resource holder. These are cases where WLM does not know what work is holding the important resources and where the new function can help. WLM gives a small amount of processor service (called a trickle) to starved work.

This support periodically examines the work coming into the queue and identifies which work units are “starved” for processor service over an extended period. WLM then promotes these work units to the ENQUEUE HOLD dispatch priority for the length of a time slice. When the time slice is complete, the work units have to wait for a defined time interval before they are considered for another promotion. Two new parameters in IEAOPTxx call this support:

- The BLWLINTHD parameter specifies the amount of time blocked work in an address space or enclave must wait before it is considered for promotion. The default for this parameter is 60 seconds.
- The BLWLTRPCT parameter specifies how much of the processor capacity to use to promote blocked workloads. This means, every second, the system uses a percentage of the processor capacity you define to help blocked work. The default for this parameter is 0.5%.

RMF supports this function by reporting relevant measurements. The Postprocessor CPU Activity report provides a new section with the following information about blocked workloads:

PROMOTE RATE the number of blocked work units that can be promoted in the dispatching priority per second. This value results from the BLWLTRPCT option settings.

PROMOTE RATE - USED (%) the utilization of the defined promote rate during the reporting interval. This rate is calculated per RMF cycle and averaged for the whole RMF interval. It demonstrates how many trickles were given away (in percent of the allowed maximum) for the RMF interval.

WAITERS FOR PROMOTE average number of address spaces and enclaves found blocked during the interval that were not promoted according to the BLWLINTHD option.

WAITERS FOR PROMOTE - PEAK the maximum number of address spaces and enclaves found blocked during the interval according to OPT parameter BLWLINTHD. The AVG value might be quite low even when there were considerable peaks of blocked workload. Thus, the peak value is also listed.

When WAITERS FOR PROMOTE is greater than zero, this means that work is being blocked for a longer period than BLWLINTHD specifies. In this case, you might want to increase BLWLTRPCT. If you have a problem with blocked work holding resources for too long and you see no waiters in the RMF data, you might want to decrease BLWLINTHD.

The Postprocessor Workload Activity report (WLMGL) indicates the amount of processor time used by transactions of a certain service or report class that were running at a promoted dispatching priority. This new field is called PROMOTED and is displayed in the SERVICE TIMES field.

It’s important that you enable this function for all production systems and examine the new information in RMF at regular intervals. The default settings guarantee a small portion of the available processor capacity is spent helping blocked workloads.
When you use z/OS Hardware Configuration Manager (HCM), your I/O configuration exists in two files: the input/output definition file (IODF) on the z/OS host, and the HCM configuration file on the Windows™ client. It’s nice to have the HCM file on your laptop and to be able to take it anywhere: you work without a connection to the host, viewing the information and even changing parts of the configuration.

Suppose you’re back from vacation, and want to know what’s changed in your I/O configuration. Your co-worker Peter is out of town for a few days. You see that a modification to the configuration file on the LAN was made on Tuesday last week and an update to the IODF three days later. Is a newer version somewhere else, or did Peter forget to resynchronize the HCM configuration with the IODF? Can you safely work with what’s on the LAN, or will there be a big merging effort later?

Here, the different locations of the two files can be painful. Sure, the IODF is well protected by the strict validation done on the host by z/OS Hardware Configuration Definition (HCD) and by the host’s reliable backup and recovery. Potential damage from mixing up HCM configurations is limited, but losing non-IODF data and having to merge conflicting changes back into a consistent state is no laughing matter. At times like this, you might wish that the HCM configuration file was stored on the host, safely, and always in sync with the IODF.

We have news for you

In z/OS V1R9, HCM introduces the concept of a master configuration file (MCF). The MCF is an HCM file stored on the host, right next to your IODF. HCM still works on the local copy of your configuration file, but that file is kept synchronized. When you open your configuration file and HCM finds a newer version on the host, it downloads the updated MCF and replaces your local file, pending your confirmation. Likewise, when you update your local configuration file, HCM uploads your changed file to the host, replacing the MCF.

You can use the MCF even if you’re the only one working with HCM. You’ll sleep better knowing that now all parts of your business-critical I/O configuration are safely stored on the host. Better still, when sharing work on the configuration instead of passing updates around on the LAN, you now get them automatically from the big iron. You’ll find that sequential sharing of configurations is much easier and safer, thanks to the MCF.

You might also appreciate seeing fewer IODF resynchronizations. As long as everybody uses HCM, you normally just need the automatic MCF download to get updates. This is considerably faster than IODF resynchronization.

We kept the local copy of the configuration file, so you can keep taking it anywhere. Stand-alone work actually gets easier with MCF. For example, after taking your laptop to the raised floor and recording information about cables, HCM is aware of the changes made in stand-alone mode. When HCM reconnects to the MCF and finds updates done by somebody else in the meantime, it shows you the differences in the new Physical Mismatch Resolution dialog (see Figure 1). You simply select your cabling updates and tell HCM to merge them, making the updated master file available to everybody.

You define MCF usage for individual configurations. For example, you might decide to use the MCF only for important configurations and not for configurations you are experimenting with, which don’t need sharing or extra safekeeping. With MCF-disabled configuration files, HCM works as it always did. You can MCF-enable an existing configuration file at any time: HCM will automatically notify your co-workers who access the same IODF. To enable or disable a configuration file for MCF, open the file in connected mode and use the new menu option provided in z/OS V1R9 (File > Master Configuration File).

Learn more

We think the MCF is big step forward for shared work with HCM and for keeping configuration files safe.

Find out more in z/OS V1R9.0 and z/VM V5R3.0 Hardware Configuration Manager User’s Guide, SC33-7989.

Figure 1. MCF Physical Mismatch Resolution dialog
Stop the wait

I/O timing facility can help

BY PURVI PATEL, JOHN STAUBI, AND HARRY YUDENFRIEND

The life of an I/O operation is measured from the time the application issues the I/O request to the time that the application is posted back with completion status. This includes I/O queue time, I/O active time, as well as retries by an error recovery procedure (ERP).

Some transaction-oriented applications require timely completion of their I/O requests and prefer termination of their I/O requests instead of excessive recovery and retries. To fulfill these requirements, the I/O timing facility was implemented in MVS/ESA V3R1.

Using the I/O timing facility, you can limit how long an I/O request can remain in the system before it is stopped. When an I/O time interval is specified, the I/O supervisor (IOS) ends an active or queued I/O request when that request exceeds the timeout value. IOS ends the request by posting the application with a permanent-error post code, issuing a message, and writing a LOGREC record. In contrast to the missing-interrupt handler (MIH), which measures only the time the I/O request is active, the I/O timing facility measures the interval from the time the application issues the I/O until the application is posted that the I/O is complete. (See Figure 1.)

The I/O timing facility is disabled by default. To enable it, use the SETIOS console command or the IECIOSxx parmlib member. Currently, the I/O timing facility is supported only for direct access storage devices (DASD) that contain only non-paging data sets. It can be enabled for a single device, multiple devices, a range of devices, or an entire device class. After enabling the I/O timing facility, you can disable it by specifying IOTIMING=00:00 on the SETIOS console command or the IECIOSxx parmlib member.

You can set both an I/O timing interval and an MIH time interval for the same device. If the I/O timing interval is less than or equal to the MIH time interval, the I/O request might end without giving MIH a chance to correct the problem. To avoid this situation, you should take the MIH time interval for the device into account when setting an I/O timing value.

Message-only feature

You might want to use I/O timing to determine how long an I/O request has been in the system, but without terminating the I/O request. To do this, you can take advantage of the message-only feature of the I/O timing facility. Message-only processing allows the system to detect I/O timeout conditions, but leaves the I/O request queued in the system. With the message-only feature enabled, a timed-out I/O request causes message IOS080I to be issued and a LOGREC entry to be written.

The message-only feature can help you pinpoint workload issues or the cause of slow I/O response times. For example, you might choose to automate based on the messages issued in order to signal alerts, trigger some recovery processing, or gather additional documentation. Table 1 compares the I/O timing facility without and with message-only processing.

Figure 1. MIH versus I/O timing
Time for some examples

1. You can enable I/O timing for the entire device class. The following operator command establishes an I/O timeout value of 35 seconds for the entire DASD device class.

\[ \text{SETIOS MIH,IOTDASD=00:35} \]

2. You can use a single statement to specify different I/O timing values for devices by using the IOTDASD, DEV, and IOTIMING keywords. The following MIH statement in the IECIOSxx parmlib member establishes an I/O timeout value of 30 seconds for all DASD devices except devices 410 through 41F. For devices 410 through 41F, ten seconds will be used as the I/O timeout value.

\[ \text{MIH IOTDASD=00:30,DEV=(410-41F),IOTIMING=00:10} \]

3. You can enable an I/O timing limit of 25 seconds with message-only processing for device 185 by using the IECIOSxx parmlib member.

\[ \text{MIH DEV=(0185),IOTIMING=00:25,MSGONLY=YES} \]

Can't wait to learn more?
For more information about using the I/O timing facility, see:
- z/OS MVS Initialization and Tuning Reference, SA22-7592
- z/OS MVS System Commands, SA22-7627.
A map is not the territory

Keeping perspective on complex I/O configurations

BY SIGRUN ELSNER

Let me guess — you probably like using z/OS Hardware Configuration Manager (HCM) for its simple graphical navigation technique, as you display and define your I/O configuration. HCM lets you create a graphical diagram of your entire configuration and save it as a file. Or maybe you actually print it as an impressive wallpaper for your office — until it becomes outdated or you simply get tired of that particular decoration.

We also like these diagrams with zillions of processors, switches, and controllers and the dazzling number of connections between them. They do provide some overview of connectivity in your data center, and they can impress some visitors.

But otherwise, they are of limited use. Many people are involved in bringing a large I/O configuration to life and in keeping it alive. Often, they aren’t much interested in the graphical view. Instead, they need reliable and complete reports in order to quickly make accurate assessments about a configuration’s ability to meet certain requirements. Or perhaps they need to compare two complex configurations from different angles. To help meet these needs, we did a lot to improve the HCM reporting facilities since z/OS V1R7.

Formatting and viewing textual reports

Previously, HCM reports could be sent only to a Windows printer driver, either for a real printer, or to a driver allowing you to create output in some graphics-capable format such as Adobe Acrobat or PostScript. Starting with z/OS V1R9, HCM offers more flexibility in specifying the report format and destination:

- All textual reports can be saved to delimiter-separated files (CSV files). This allows you to easily load the reports into a spreadsheet program for further processing and analysis. See Figure 1 for an example.

![Figure 1. HCM configuration (CSV format)](image-url)

Figure 1. HCM configuration (CSV format)
• When selecting the new XML/HTML output format, you can view the generated reports in a Web browser. HCM provides appropriate XSL and CSS style sheets as well as XCD schema definitions for this purpose.

To avoid cluttered reports (and to save paper), you will appreciate seeing the much-improved limitation capabilities available in the z/OS V1R9 version of HCM: they allow you to flexibly restrict reports to a certain scope of interest.

Since z/OS V1R7, HCM offers more reports in a condensed format. For example, you might know the accurate and time-proven reports created by z/OS Hardware Configuration Definition (HCD). They have evolved over the years, and they keep getting better as we take into account feedback from customers like you. We now make them available within HCM: all report type selections and scope limitations that you know and rely upon in the HCD ISPF interface are now available through HCM dialogs.

Comparing two configurations

Previously you had no way of comparing two configurations directly and efficiently within HCM. As of z/OS V1R8, this has changed:

• For a long time, HCD has been able to compare two IODFs. The resulting reports are now available in HCM as well. All comparisons provided by HCD and all scope limitations you are familiar with from ISPF can be accessed from within HCM.

• In addition to exploiting the logical comparison capabilities of HCD, you can now also compare the physical aspects of two configurations: cables, patchports, cabinets, controllers, user-defined description fields — all the things that are not stored in the IODF and that only HCM knows about. (See Figure 2.) Scope limitations to restrict comparison to certain aspects or attributes of your configuration are just a few mouse clicks away.

Conclusion

If you've been using HCM mostly to navigate and define configurations, or just haven't looked at its reporting capabilities recently, give our new versions a try: we think they have much to offer.


Figure 2. HCM compare report (physical data)
Here’s to the SAP

The original mainframe specialty processor

BY JOHN H. KETTNER

With all of the recent focus on the new IBM System z specialty processors, zAAP and zIIP — see z/OS Hot Topics Newsletter Issue 17, August 2007, GA22-7501-13 — did you know that your mainframe already comes with a type of specialty processor installed? For many years now, IBM has included the System Assist Processor or SAP in its mainframes to help speed up certain types of input and output (I/O) operations. The SAP is standard on IBM System z servers.

The SAP processor is special because it is dedicated to handling complex I/O tasks that create the data frames for transmission to and from the I/O devices. By offloading these tasks to the SAP, we help to reduce the I/O processing overhead of the logical partitions (the operating system and the general purpose CPUs) on your mainframe.

Let’s take this opportunity to become better acquainted with the SAP.

What is the SAP?

Commercial applications perform I/O — a lot of it. When a computer’s CPU is forced to handle this logic, it can spend much of its time processing I/O operations. Some other platforms today devote general purpose processors to handling this I/O role, but that approach can become an expensive means of accessing data.

In contrast, IBM mainframes use the SAP, an asynchronous processor, to handle the preparation (set-up and scheduling) of I/O operations. The SAP is an I/O processor (IOP) running special licensed internal code.

The SAP drives the mainframe’s I/O channel subsystem, which serves a vast collection of more than a thousand high-speed buses. The SAP relieves the operating system and the general purpose CPUs of much of the work required to execute I/O operations. Specifically, the SAP schedules and starts each I/O request; that is, it locates an available channel path to the requested I/O device and starts the I/O operation. The SAP does not handle the actual data movement between central storage (CS) and the channel, however.

The SAP also serves a role in CP sparing. In the unlikely event that a CP fails, the SAP becomes a temporary work area to hold the current instruction while a new CP is reassigned.

How does it work?

For a given I/O operation, the SAP processes the Start-Subchannel (SSCH) instruction, which involves locating a subchannel or logical device in its initial work queue. The SAP processes requests in this queue, based on the I/O priority assigned by your installation’s z/OS Workload Management (WLM) policy, or through the Hardware Management Console (HMC) if the logical partition (LPAR) does not use Channel Subsystem I/O Priority Queuing.

Did you know that your mainframe already comes with a type of specialty processor installed?

Figure 1. At least one SAP is standard on every IBM System z

<table>
<thead>
<tr>
<th>Model Books</th>
<th>SAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S08</td>
<td>1</td>
</tr>
<tr>
<td>S18</td>
<td>2</td>
</tr>
<tr>
<td>S28</td>
<td>3</td>
</tr>
<tr>
<td>S38</td>
<td>4</td>
</tr>
<tr>
<td>S54</td>
<td>4</td>
</tr>
</tbody>
</table>

Model Books SAPs

1 2 4 6 8
defaults

Plain text representation:

With all of the recent focus on the new IBM System z specialty processors, zAAP and zIIP — see z/OS Hot Topics Newsletter Issue 17, August 2007, GA22-7501-13 — did you know that your mainframe already comes with a type of specialty processor installed? For many years now, IBM has included the System Assist Processor or SAP in its mainframes to help speed up certain types of input and output (I/O) operations. The SAP is standard on IBM System z servers.

The SAP processor is special because it is dedicated to handling complex I/O tasks that create the data frames for transmission to and from the I/O devices. By offloading these tasks to the SAP, we help to reduce the I/O processing overhead of the logical partitions (the operating system and the general purpose CPUs) on your mainframe.

Let’s take this opportunity to become better acquainted with the SAP.

What is the SAP?

Commercial applications perform I/O — a lot of it. When a computer’s CPU is forced to handle this logic, it can spend much of its time processing I/O operations. Some other platforms today devote general purpose processors to handling this I/O role, but that approach can become an expensive means of accessing data.

In contrast, IBM mainframes use the SAP, an asynchronous processor, to handle the preparation (set-up and scheduling) of I/O operations. The SAP is an I/O processor (IOP) running special licensed internal code.

The SAP drives the mainframe’s I/O channel subsystem, which serves a vast collection of more than a thousand high-speed buses. The SAP relieves the operating system and the general purpose CPUs of much of the work required to execute I/O operations. Specifically, the SAP schedules and starts each I/O request; that is, it locates an available channel path to the requested I/O device and starts the I/O operation. The SAP does not handle the actual data movement between central storage (CS) and the channel, however.

The SAP also serves a role in CP sparing. In the unlikely event that a CP fails, the SAP becomes a temporary work area to hold the current instruction while a new CP is reassigned.

How does it work?

For a given I/O operation, the SAP processes the Start-Subchannel (SSCH) instruction, which involves locating a subchannel or logical device in its initial work queue. The SAP processes requests in this queue, based on the I/O priority assigned by your installation’s z/OS Workload Management (WLM) policy, or through the Hardware Management Console (HMC) if the logical partition (LPAR) does not use Channel Subsystem I/O Priority Queuing.
The SAP then attempts to locate an available channel that connects to the control unit, and starts the I/O operation. The SAP uses information in the subchannel to determine which channels and control units it can use to reach the requested I/O device. The channel builds and executes a channel program, which is a list of channel command words (CCWs), controlling the movement of data between device, control unit, and processor storage.

After the I/O completes, the channel signals the completion to the SAP, which in turn signals the completion to a CP through an I/O interrupt. If the I/O was a read operation, the SAP examines a directory to determine whether the requested data is in cache. If so, the SAP can save further processing cycles by simply copying the data from one area of central storage (the cache in the Hardware System Area or HSA) to another area of central storage (the area specified by the requesting program).

**Sounds great — how do I get one?**
You already have one (or more). Unlike the fee-based specialty processors, at least one SAP is included in every mainframe without additional charge. The IBM System z9™ BC server, for example, has one SAP by default, and that SAP assists with all of its I/O operations.

On the current System z9 server, different models come with a fixed number of SAPs as shown in Figure 1, although your installation can increase I/O throughput by adding more SAPs per processor book.

Because the SAP targets only I/O operations, you cannot use this processor for general processing, such as IPLing an operating system. For z/OS, the SAP participates in the IPL process only during the building of the I/O configuration data set (IOCDS), which contains the mainframe’s I/O configuration data.

**Is one SAP enough?**
For very high I/O intensive workloads, your installation might choose to purchase additional SAPs, depending on your processor model. You can assess whether you need additional SAPs by analyzing RMF I/O Queuing Activity Reports or by checking system activity in the processor’s Support Element (SE) console.

To add additional SAPs over the model default, you use z/OS Hardware Configuration Definition (HCD). Here, you can assign the system CP/SAP allocations to optimize the performance of your applications without performing a power-on reset (POR).

Converting CPs to SAPs requires reassignment of the current licensed internal code. However, your installation should ensure the CPs being converted to SAPs can be deconfigured and are not dedicated.

Each mainframe model automatically balances installed channel cards across all available SAPs. The mainframe attempts to assign an equal number of each channel type (ESCON, FICON, and so on) to each available SAP. All channels on a given I/O card have affinity to the same SAP.

**A final SAP tidbit**
For more about IBM System z specialty processors, see [ibm.com/systems/z/specialtyengines/](http://ibm.com/systems/z/specialtyengines/).
I-M? Or I-B-M?

A gr8 Hot Topics quiz 4u

Have you noticed that our everyday communications seem to rely more and more on acronyms, jargon, and abbreviations? As IT professionals, our conversations, meetings, presentations, and e-mail often contain more abbreviations than complete words!

Here at IBM, we love abbreviations—especially acronyms! Most of our products and technologies (not to mention the company itself) are referred to by their respective abbreviations. Flip through this newsletter and you probably won’t find a single page that does not contain an abbreviation of some kind. (If you do, let us know and we will fix it immediately.)

Perhaps the only area where so many abbreviations can be found outside IBM is in real-time text-based communication that uses instant messaging (IM). The need to save on typing when sending instant messages has created a new language of shorthand abbreviations. Ask any teenager!

In the following quiz, try to determine whether the abbreviations are IBM-related or are shorthand abbreviations that might be used for IMs. Don’t be too quick or confident in your answers, however, because some are both IBM and IM abbreviations!

1. Is RAS:
   (a) A z/OS abbreviation for “reliability, availability, and serviceability,” referring to the key traits of the operating system.
   (b) An IM abbreviation for “really am sleepy” meaning “It’s late; let’s stop texting so I can go to bed.”
   (c) Both a and b.

2. Is LOL:
   (a) A z/OS abbreviation for “Log Output Lister,” a utility you can use to view and search the system log.
   (b) An IM abbreviation for “laughing out loud” (more convenient than typing out “HA-HA-HA-HA-HA!”).  
   (c) Both a and b.

3. Is PDS:
   (a) A z/OS abbreviation for “partitioned data set.” Sometimes called a “library,” a partitioned data set is a collection of sequential data sets.
   (b) An IM abbreviation for “please don’t shout.” This abbreviation is used in IM messages in response to someone typing in ALL CAPS (which is considered the written equivalent of shouting).
   (c) Both a and b.

4. Is ICF:
   (a) A z/OS abbreviation for “Integrated Coupling Facility.”
   (b) An IM abbreviation for “I can’t follow,” meaning “I don’t understand what you’re saying.”
   (c) Both a and b.

5. Is VSF:
   (a) A z/OS abbreviation for “Virtual Storage Facilitator.” When working with data sets, VSF refers to an access method for direct or sequential processing of fixed and variable length records.
   (b) An IM abbreviation for “very sad face,” used when ::-( wouldn’t convey enough unhappiness.
   (c) Both a and b.

6. Is IMS:
   (a) A z/OS abbreviation for “Information Management System,” a transaction and hierarchical database management system.
   (b) An IM abbreviation for “I am sorry,” used when you’re not so apologetic to actually spell out the words.
   (c) Both a and b.

7. Is SCLM:
   (a) A z/OS abbreviation the Software Configuration and Library Manager facility, a software configuration management system.
   (b) An IM abbreviation for “stay cool like me.”
   (c) Both a and b.

8. Is RME:
   (a) A z/OS abbreviation for “Remote Management Environment,” an application availability solution that enables you to manage remote copy configuration and storage subsystems and perform failure recovery.
   (b) An IM abbreviation for “rolling my eyes.”
   (c) Both a and b.

9. Is CTC:
   (a) A z/OS abbreviation for “channel to channel,” referring to a type of connection between systems.
   (b) An IM abbreviation for “care to chat?” used to initiate a conversation.
   (c) Both a and b.

10. Is TTYL:
    (a) In UNIX, a generic device driver for a text display (that is, text teletype).
    (b) An IM abbreviation for “talk to you later.”
    (c) Both a and b.

Answers on page 73
Setting up OpenSSH

when you are NOT a z/OS UNIX guru

BY PAUL EVANS, ERIN FARR, AND C.T. WARE

Are you an MVS system programmer who found out that you need to install OpenSSH, which lives in the shadowy land of z/OS UNIX System Services (z/OS UNIX)? Does fear grip you when you think what's waiting for you out there? This article tries to unravel the mystery of getting OpenSSH working and takes the fear out of installing and configuring the daemon and client. We also hope to enlighten you about using sftp through BPXBATCH.

Installing OpenSSH—be brave!

OpenSSH is part of IBM Ported Tools for z/OS. The best place to start is with the familiar SMP/E steps. Because OpenSSH has prerequisites for z/OS, z/OS UNIX, and TCP/IP, you are pretty much stuck with putting OpenSSH in the same SMP/E zone. Also, watch out for the hard-coded paths within the code. You must use the default installation paths. See the Program Directory for IBM Ported Tools for z/OS at: ibm.com/servers/eserver/zseries/zos/unix/pdf/docs/fotza202.pdf

OK, that was the easy part. When you've ensured that OpenSSH is installed, then comes the really fun part: configuring the “Beast.” You will have to familiarize yourself with some of that z/OS UNIX stuff.

Configuring OpenSSH—be serene


The IBM Ported Tools for z/OS User’s Guide keeps getting updated with new information, so be sure you are looking at the latest version.

Second, consider these basics for OpenSSH:

1. Client commands, including:
   • sftp — a secure file transfer utility
   • ssh — a secure login shell
2. Server
   • sshd — the secure shell daemon
3. Utilities, including:
   • ssh-keygen — a key generation utility
   • ssh-keyscan — a utility for gathering public host keys

Set up the server part of the product first, so that you can interactively use the client commands to access the z/OS system and test things. We know you really want to use just BPXBATCH, but to get things going you will probably need to do at least one sftp interactively. This means setting up the sshd server so that you can connect to your z/OS system, then using the sftp command from z/OS UNIX. It becomes clearer how to use BPXBATCH when you've done this interactively.

The server side of the coin—setting up the daemon

Next take a look at “Chapter 4. For system administrators” in the IBM Ported Tools for z/OS User’s Guide, which describes setting up the OpenSSH daemon. Follow the steps in that chapter to help get you up and running. You can tweak your settings after you're done.

To create the daemon, assume that you will use TSO OMVS (the UNIX shell) with READ access to BPX.SUPERUSER (that is, with the UID(0) user ID). For all z/OS UNIX commands, like those that copy sample files, copying and pasting them from the manual should work.

The chapter also includes steps that describe changing the server and client configuration files. Use the defaults as delivered; they are usually good to go. Once everything is working, you can think about getting fancy.

Now, let's look at building keys, which is an area that can be a tad confusing.

Sorting out the keys for authentication

OpenSSH keys are stored in the z/OS UNIX file system. There are two main uses for the keys that OpenSSH needs for authentication:

• To authenticate the user logging on to the system
• To authenticate the system to the user.

The system host keys

Clients out in the world use host keys to ensure that they are communicating with the correct server. Because few clients use protocol version 1 of the keys, you probably want to build protocol version 2 keys. Of these, you can create Digital Signature Algorithm (DSA) keys, Rivest Signature Algorithm (RSA) keys, or both. For now, we’ll create a DSA key.

In the IBM Ported Tools for z/OS User’s Guide, find the table “Generating the host keys for the SSH server;” copy and paste the z/OS UNIX ssh-keygen commands from the table, and then’s that — you’ve created the host keys!

For a DSA key, these same ssh-keygen commands build files in /etc/ssh/ that look like this:

```
ssh_host_dsa_key
ssh_host_dsa_key.pub
```

There is a private part and a public part to each key. The file with a .pub extension is the public part of the key. The client gets the public part of the key and saves that part in the known hosts file, which resides in the z/OS UNIX file system. This allows the client to establish connections with your host.

You can collect host keys from other systems to allow your users to establish...
outbound OpenSSH connections to those systems. You can get the public key of another system using the `ssh-keyscan` command. The private key and passphrase (if used) are placed in your system-wide known hosts file `/etc/ssh/ssh_known_hosts` for any client on your system to use.

Here’s an example of the command:  
```bash
ssh-keyscan -t dsa IP_hostname >> /etc/ssh/ssh_known_hosts
```

IP_hostname is the IP name of the target host (for example, yoursysthere.com.) You can also specify both hostname and IP address to obtain a public key that allows users to specify the name or IP address interchangeably.

The `-t dsa` option asks for the DSA type of public key from the remote host.

The `>>` operator appends the output of the command (the key public) to the `/etc/ssh/ssh_known_hosts` file. Be sure to look at that file to ensure that the correct key is there. This is only the system half of the keys. Now, you need the client keys.

### The client side of the coin

If a user wants to transfer files with `sftp` through BPXBATCH, one way is by using keys for user authentication.

When the keys are for use from BPXBATCH, build them with a null passphrase. That means when `ssh-keygen` prompts for a passphrase, you just press Enter.

To generate authentication keys for OpenSSH, use the `ssh-keygen` command, which runs from TSO OMVS. Just build a protocol version 2 key while logged in under your user ID. The location of your key is based on your `$HOME` directory. The `$HOME` variable that OpenSSH uses comes from the value for the `HOME` statement in the RACF OMVS segment of your user ID. (If you use a null passphrase, know that any unscrupulous person can obtain your private key file to access your accounts as opposed to your private key and passphrase.)

For complete information, see the section “Steps for setting up user authentication” of the IBM Ported Tools for z/OS User’s Guide, SA22-7985.

---

### Authenticating the local user to a remote system

Next, send the local user’s public key to the remote system to “publish” it in the remote user’s authorized keys file. This action allows the local user to be authenticated to the remote system. To transfer it, use your PC clipboard, old fashioned `ftp`, or interactive `sftp` with a password. The `authorized_keys` file resides in the z/OS UNIX subdirectory `.ssh` under the user ID’s `HOME` directory on the remote site. For example, if the remote user’s `HOME=` value is `/u/William`

the authorized keys file is

```bash
/u/William/.ssh/authorized_keys
```

Different implementations of SSH are available on other platforms. These other platforms might have a different location where OpenSSH needs to place the user’s public key.

You’ve reached your goal! Now you can use OpenSSH.

### Running sftp from BPXBATCH

We return to something more familiar to the MVS System Programmer — good old JCL. Figure 1 shows sample JCL that user BILLY on HOST1 runs to perform an `sftp` from BPXBATCH. The STEP1 step is a simple `IEBGENER` that sets up a stream of `sftp` sub-commands that the `sftp` command in STEP2 can read:

```
//STEP1 EXEC PGM=IEBGENER  
//SYSPRINT DD DUMMY  
//SYSIN DD DUMMY  
//SYSUT2 DD PATH="/tmp/&SYSUID.sftp.input",FILDATA=TEXT,PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRMUX  
!cp -tr"/YOUR.DATA.SET.C(HELLO)" hello.c  
binary  
put hello.c /u/William/hello.c  
!rm hello.c  
quit  
!*/  
//STEP2 EXEC PGM=BPXBATCH,  
//PARM=('PGM=/bin/sftp -b /tmp/&SYSUID..sftp.input',  
//       'WILLIAM\&HOST2')  
//STDOUT DD SYSOUT=*  
//STDERR DD SYSOUT=*  
!*
```

OpenSSH does not have direct support for MVS data sets; however, you can call z/OS UNIX commands from within `sftp` to copy MVS data sets to and from the z/OS UNIX file system. Documentation APAR OA10742 describes the use of the exclamation mark to escape to the local shell on HOST1 and perform z/OS UNIX commands there. The `cp` command is a copy command that can move an MVS data set into the z/OS UNIX filesystem where `sftp` can then send the data to the remote site. At the end of the stream of sub-commands is the z/OS UNIX `rm` command that removes the file from the local z/OS UNIX file system (just to be tidy.)

STEP2 runs the z/OS UNIX `sftp` command using BPXBATCH, reading the stream of commands from STEP1. Note that STEP2 also shows how to continue a long `PARM=` line. The second line starts in column 16.

As part of the `sftp` command, specify the user ID to use on the remote system and, of course, the remote system that you are sending data to. In Figure 1, the remote system is HOST2 and the user ID is WILLIAM. The public key for BILLY will be found in the authorized keys file of WILLIAM, authenticating the user to the system.

### The path is clear

We hope this article has guided you through some of the z/OS UNIX mysteries of OpenSSH. But if not, Paul, your IBM guru, is patiently waiting for your Q&A call. ■

Namaste.
move z/OS UNIX directory trees with \texttt{pax}

\section*{How to}

\textbf{BY GARTH GODFREY}

In z/OS UNIX System Services, the \texttt{pax} command is used extensively for moving z/OS UNIX directory trees. And, it’s no wonder—this one command allows you to “wind up” (package) an entire z/OS UNIX directory tree into a single archive file and later “unwind” (extract) those trees onto another system. You have probably found \texttt{pax} to be very useful for doing things like cloning or backing up directories.

Is there a risk, however, that using \texttt{pax} could cause you to lose some file attributes? If you are not careful, the answer is yes!

Here are some quick tips for using \texttt{pax} correctly.

\textbf{To \texttt{pax} or not to \texttt{pax}?}

The \texttt{pax} command allows you to write, read, list, or copy archive files (archive is another name for a tarball). Because it is a z/OS UNIX shell command, you can enter it whenever you are logged into a shell, or you can issue it from a shell script or a BPXBATCH job.

The \texttt{pax} command is the best choice for doing the following types of work:

\begin{itemize}
  \item Moving parts of a directory tree from one z/OS system to another.
  \item Unwinding a tarball, which is an archive created by the \texttt{tar} command, from a UNIX system. (\texttt{tar} is the historical UNIX command; it is superseded by \texttt{pax}.)
  \item Creating a tarball that can be unwound on other systems (z/OS or non-z/OS).
\end{itemize}

When is \texttt{pax} not your best choice? When you need to clone an entire file system from one z/OS system to another. Here, you should use a dump and restore method, such as DFSMSdss, instead.

\section*{Some facts about \texttt{pax}}

The \texttt{pax} command has four main actions—write, read, list, and copy:

\begin{itemize}
  \item \texttt{-w} writes (or creates or winds up) an archive, typically to store an entire directory tree.
  \item \texttt{-r} reads (or extracts or unwinds) an archive.
  \item \texttt{-rw} copies a directory tree from one location to another on the same system.
\end{itemize}

\section*{Is there a risk, however, that using \texttt{pax} could cause you to lose some file attributes?}

\section*{Make a worthy choice}

You can create \texttt{pax} archives in a variety of formats, but you can usually whittle it down to two good choices:

\begin{itemize}
  \item \texttt{USTAR}. The default format; it can be exchanged with other UNIX systems. Use this format for an archive that must be unwindable on a pre-z/OS V1R8 system or on an older level of another, non-z/OS UNIX system (that is, a system that does not support the UNIX03 specification).
  \item \texttt{pax}. Specified by the \texttt{-x pax} option when you create an archive; this is the latest format for exchanging archives with UNIX systems that support the UNIX03 specification. Use this format for archives that you will unwind only on z/OS V1R8 or later systems (or other UNIX systems that support the format).
\end{itemize}

\section*{To envelop and contain celestial spirits...}

On z/OS, you can use an MVS data set to hold a \texttt{pax} archive. You might do this if you are concerned about having enough z/OS UNIX file system space to hold the archive. Alternatively, if you are transferring the archive to another system, you can use an MVS data set as an intermediate location for the archive.

For an MVS data set, use the following syntax to specify the data set name when you create the archive:

\begin{verbatim}
-f "/'posix.godfrey.pax(gbin)'"
\end{verbatim}

Here, I created a new member \texttt{gbin} in an existing partitioned data set (PDS).

Later, when you unwind the archive, use the same syntax to read the archive from the data set member.

\section*{Three incantations you’ll need}

There are many choices when it comes to \texttt{pax} options, but here’s what I recommend for performing the most common tasks.

\section*{Creating an archive}

Suppose I need to create an archive for backing up a directory tree. Assume that the tree will be restored on z/OS systems only, and that all of its attributes should be saved.

The steps for creating this archive depend on whether I plan to restore it on a pre-z/OS V1R8 system or a later system.
For an archive that I might restore on a pre-V1R8 system, I could enter:

```
cd /u/godfrey/bin
pax -wz -o saveext -f /u/godfrey/backup/bin.pax
```

In this example, notice that:
- `/u/godfrey/bin` is the directory for backing up the directory tree.
- The `cd` command sets my working directory to the top of the file hierarchy to be archived. I tell `pax` to start at this directory by specifying `"."` as the pathname. This approach gives me the option to unwind the archive later under a different directory, if need be.
- The archive (named by the `-f` argument) is created in a file that resides outside of the directory tree that I am archiving. Always do this.
- The `-o saveext` option ensures that `pax` uses z/OS specific headers in the archive, to save long names and all file attributes.

On the other hand, if I know that the archive will only be restored on a z/OS V1R8 or later system, I will use a slightly different approach:

```
cd /u/godfrey/bin
pax -wz -x pax -f /u/godfrey/backup/bin.pax
```

Here, I used the `-x pax` option, instead of `-o saveext`. Introduced in z/OS V1R8, the `-x pax` option uses a UNIX standard format to save long names and all file attributes.

### Listing the archive

To list an archive with the extended attributes, I could enter:

```
pax -v -oE -f /u/godfrey/backup/bin.pax
```

### Unwinding the archive

To unwind the archive on a different z/OS system, I must first transfer the archive to the new system. Here, I transfer the bin. pax archive from the `/u/godfrey/backup` directory on the original system to the `/tmp/godfrey` directory on the new system through binary FTP:

```
cd /u/godfrey/bin
pax -r -pe -f /tmp/godfrey/bin.pax
```

In this example:
- `/u/godfrey/bin` is the directory in which the archive is to be unwound.
- `pax` automatically recognizes the archive’s internal format.
- The `-pe` option tells `pax` to preserve everything. This option restores all of the file attributes that were saved in the archive. (You will need the proper authorizations, of course.)

### Three portability issues to heed

If you need to write a shell script that will run on multiple UNIX systems, be careful of three potential portability issues:
- On z/OS V1R8 and later, specify the environment variable `_UNIX03=YES` to ensure that the `pax` command syntax follows the current UNIX standard rules.
- Ensure that your shell script and the text data in the files use the correct character set. By default, most text files on z/OS use the EBCDIC character set, while most other UNIX platforms use ASCII. The z/OS UNIX `pax` command provides options to perform character set conversion on the fly. In the following example, `pax` is used to convert the files in the archive from ASCII to EBCDIC as part of an unwind on z/OS:

```
pax -rf /tmp/import.pax -o from=ISO8859-1, to=IBM-1047
```

- z/OS extended attributes are not supported on other UNIX systems, so avoid using `-o saveext`.

### Don’t be like Macbeth

(He feared *moving trees.*) I hope that these tips will help you move your z/OS UNIX directory trees with confidence and peace of mind.

---

---
Students are learning about enterprise systems (that is to say, mainframe) concepts and gaining practical experience in the classroom with the help and support of college and university educators, the IBM Academic Initiative System z program, and the mainframe community. You can hear, firsthand, what students are saying about mainframes, and learn why they feel that studying enterprise systems sets them apart from other students and gives them a competitive edge in the marketplace.

Perspectives on mainframes: From the mouths of students
Recently, I spoke with students who are taking a variety of enterprise systems classes at Illinois State University, the University of Arkansas, the University of Toledo at Ohio, and Syracuse University. These schools are all participating members of IBM’s worldwide Academic Initiative program. To see a list of participating schools, visit ibm.com/university/systemz.

Here’s what the students had to say.

Troy Crutcher, Illinois State University, Information Systems

Q: What mainframe topics are you studying in school?
Illinois State University is developing an Enterprise Computing Systems program in collaboration with local and regional corporate representatives and the IBM Academic Initiative System z program. One of the classes in this program is called Introduction to Enterprise Computing Systems. This class was my first exposure to mainframe systems.

Q: Is there anything that you learned about mainframes that has surprised you?
Having had the opportunity to work firsthand on mainframe systems, I am amazed at the performance of these machines. Nothing comes close to matching the IBM System z performance and high levels of availability.

Q: What would you share with other students who might be interested in learning about mainframes?
As a result of taking the Introduction to Enterprise Computing Systems class, opportunities opened up for me that I never knew existed. In the spring of 2007, I completed a six-month internship at IBM in Poughkeepsie, N.Y. During this unique rotational co-op assignment, I spent time working in five different areas: z/OS Build and Installation, Solution Support, the Benchmark Center, Integration Test, and Service. My internship at IBM was an incredible experience that left me wanting to learn more.

Q: Do you feel that learning about mainframes will better prepare you (give you an advantage) for a job in IT?
Most students believe that the mainframe system died a long time ago, and that was my perception. After studying enterprise systems at school and completing my IBM internship, I know that nothing could be further from the truth. There is a huge demand for mainframe skills in the industry. After returning to school, I’ve been a mainframe evangelist and have been spreading the word about the demand for students with these skills.

Q: Have you participated in any student mainframe contests or internships? Briefly tell us about your experiences.
In addition to participating in an IBM co-op program, I participated in the IBM Master the Mainframe contest. No experience was necessary to participate, and it was my first live interaction with a mainframe system. It was pretty cool being able to see the various things you could do with one of those machines.
Adam Norwood, University of Arkansas, Information Systems

Q: What mainframe topics are you studying in school?
The University of Arkansas offers a full enterprise systems curriculum. I recently studied CICS development — in particular, how to tie CICS in with the Web and integrate it with different database technologies.

Q: Is there anything that you learned about mainframes that has surprised you?
Before I learned more about mainframes, I thought that the mainframe was a very limited legacy platform. It was definitely surprising to find that the mainframe is incredibly adaptable and can really handle everything you could ever want to throw at it. With [z/OS] UNIX System Services, [Linux® on System z], virtualization, strong security, and Web interfaces all so tightly integrated with z/OS, there’s very little one cannot do these days on a mainframe.

Q: What would you share with other students who might be interested in learning about mainframes?
Definitely do it. It’s one of the best things a student studying in the business or technology fields can do. You are instantly marketable once you pick up mainframe skills, and it’s a great way to become familiar with the many facets of IT in general. Mainframes can do it all.

Q: Do you feel that learning about mainframes will better prepare you (give you an advantage) for a job in IT?
Absolutely. With business IT becoming more and more of a complex tech “ecosystem,” the bonus of more specialized coursework in a strong platform like System z will give me a huge advantage over the average IT job candidate.

Di Lu, Syracuse University, Information Management

Q: What mainframe topics are you studying in school?
Syracuse University introduces enterprise systems concepts in several of its courses: from database management to courses that address management and technical architecture of enterprise systems environments. In my studies, I’ve learned about and gained hands-on experience with supported operating systems, service-oriented architecture, virtualization concepts, security and policy compliance, and application development tools such as IBM Rational Developer for z.

Q: Is there anything that you learned about mainframes that has surprised you?
I always imagined that “Big Iron” would be physically very big. But, in actuality, I was surprised to learn just how small these systems are and how amazing it is that thousands of virtual servers can run within a single box.

Q: What would you share with other students who may be interested in learning about mainframes?
IBM has application tools, such as Rational® Developer for System z (a GUI-type interface for mainframes), that allow you develop, test, and deploy simple and complex applications with the click of a mouse. This modern and simplified approach accelerates the development of Web applications,
COBOL applications, and XML-based interfaces. Interactive workstation-based environments make it easy for students who are new to the mainframe environment to become productive more quickly.

Q: Do you feel that learning about mainframes will better prepare you (give you an advantage) for a job in IT? Manageability of massive server farms, power consumption, and data center real estate are very real concerns for businesses. As companies look to consolidate their environments, [as well as] lower costs and power consumption, they will look toward “green” and environmental solutions such as the IBM mainframe system. Knowledge of the mainframe systems and the advantages they provide will give current college students a competitive advantage in the IT job market.

Q: Have you participated in any student mainframe contests or internships? Briefly tell us about your experiences. Yes, I am currently participating in the 2007 [Master the Mainframe] contest. I’ve found the contest to be challenging, exciting, and incredibly beneficial. Students do not need any background or prior knowledge with mainframes in order to participate. During the contest, students learn practical aspects of systems programming and application development on the mainframe, like how to use ISPF interfaces, JCL, SDSF, COBOL, Parallel Sysplex, and DB2. Learning while having fun, what could be better?

Adam Shea, University of Toledo, Computer Science and Engineering Technology

Q: What mainframe topics are you studying in school? As a student at the University of Toledo in Ohio, my first introduction to mainframe systems was a class I took called Introduction to the New Mainframe: z/OS Basics.

Q: Is there anything that you learned about mainframes that has surprised you? The reliability and security features on the mainframe are light years ahead of anything I have seen. Students are unaware just how prevalent mainframes are in today’s world.

Q: What would you share with other students who may be interested in learning about mainframes? The mainframe is the most advanced commercial computing platform available. If you get the opportunity, take an enterprise systems class. A large majority of global Fortune 1000 companies are mainframe clients and approximately 80% of the world’s corporate data originates or resides on mainframes.

Q: Do you feel that learning about mainframes will better prepare you (give you an advantage) for a job in IT? Definitely. Traditional mainframe knowledge in the COBOL and CICS languages, together with open technologies such as Java running on Linux, are very much in demand.

Q: Have you participated in any student mainframe contests or internships? Briefly tell us about your experiences. Yes, I participated in an IBM internship program in 2006 and a summer co-op in 2007. Participating in applied learning programs like internships and co-ops gave me the practical experience I was looking for. Working side by side with leading mainframe subject matter experts was invaluable. After completing my internship, I took, and passed, the IBM System z Entry Level for z/OS System Programmer worldwide mastery test. By passing this test, I validated to myself and potential future employers that I have what it takes to run these mission-critical mainframe systems.

The mainframe in your future
There you have it from the next generation of mainframe enthusiasts. The word is out about the value of learning mainframe skills for all those budding IT specialists who we think will be developing rewarding careers in the industry. Long live the mainframe!
Web 2.0 has initiated a new age of collaborative and social networking Web sites. As mainstream businesses begin to explore Web 2.0 and its underlying technologies, it is natural for them to question the business value of adopting yet another programming model. What is the advantage of Web 2.0 and its simplified Web services over the well-known Web services? The answer is, it enables non-technical users to solve business problems by using a more cost-effective, quicker-to-deploy Web mashup technology with minimal or no aid from information technology (IT) specialists. Thanks to the openness of this mashup technology in Web 2.0, Web surfers with no programming skills can now easily customize their Web sites with more feeds and widgets from elsewhere on the Web.

IBM Mashup Starter Kit contains two key technologies: Mashup Hub and QEDWiki. IBM Mashup Hub is a mashup tool that creates and stores IMS RESTful (Representational State Transfer) services to enable reuse and collaboration. It runs on the bundled Apache server and offers powerful widgets to merge, transform, filter, annotate, or publish information in new formats, such as RSS, ATOM, and XML. QEDWiki serves as the user interface editor, allowing non-IT service developers to quickly and easily mash information feeds to create a single view

**IMS and Web 2.0**

The IBM IMS Web 2.0 strategy goes far beyond what Web 2.0 can offer, by unleashing enterprise assets that run crucial business processes. Countless everyday activities such as checking order status, looking up inventory, and filling in forms run enterprise IMS transactions behind the scenes. Now it is possible to take advantage of the power of IMS effectively, and often more cheaply, on the Web. A new offering, IMS Info 2.0, uses Web 2.0 to simplify the integration of data and content, thereby shortening the development and deployment cycles. IMS Info 2.0 makes IMS assets easier to use by enabling users to remix and mash up the data. IMS Info 2.0 is part of IBM Mashup Starter Kit ([alphaworks.ibm.com/tech/ibmmstk](http://alphaworks.ibm.com/tech/ibmmstk)), which empowers line-of-business professionals to create mashup Web sites and application solutions with content and feeds from enterprise data sources or anywhere else on the Internet.

---

**Unleash IMS with Info 2.0**

*BY JENNY HUNG AND SHYH-MEI HO*

---

**Figure 1. How IBM Mashup Starter Kit supports mashup Web sites**
of disparate sets of information and then deploy that view as a Web page. Together, this package collects and processes information that ultimately provides data for next-generation applications. What’s more, you can use the IBM Mashup Starter Kit without writing a single line of code!

**How does it work?**
Let’s look at Figure 1. First, a service developer uses Rational Developer for System z to generate the metadata file and IMS Connect XML converters by importing the COBOL copybook of an IMS transaction. The service developer then uses IBM Mashup Hub to create an IMS feed by specifying Enterprise Data Sources as the category, IMS as the data source, and then providing the IMS connection information and the generated metadata file. The service developer must also have the generated IMS Connect XML converters compiled and linked into IMS Connect. Now the IMS feed is ready to be served.

At run time, a Web application (such as a Web browser, QEDWiki widget, or another Web mashup) can invoke the IMS feed (that is, the RESTful service). When the IBM Mashup Hub server receives an HTTP request from the Web application, it looks up the service in the RESTful services registry and invokes the IMS RESTful service adapter. The IMS RESTful service adapter processes the parameters, establishes a connection with IMS Connect, and sends the request as an XML message. The IMS Connect XML Adapter function then invokes the inbound XML converter to transform the XML message into an input byte array that is understood by the IMS transaction.

When the response returns from IMS, IMS Connect uses the XML Adapter function and the outbound XML converter to transform the output byte array into XML and sends it to the IMS RESTful service adapter. The IMS RESTful service adapter then forwards the XML message to the RESTful service, which returns the XML output to the Web application.

Figure 2 depicts a sample mashup composed of IMS data and a Google map. Using IBM Mashup Hub and QEDWiki, the fictional company Tractor Company 2.0 is able to extend its IMS assets (the tractor branch and inventory transaction data) and customize the Web experience for its users by mashing together Google Maps and other Web services. Furthermore, Tractor Company 2.0 can publish the IMS feed to the Web community, allowing any other Web consumer to reuse the information and create mashups, creating new business uses and opportunities.

Thanks to IBM Mashup Starter Kit, it is now easier than ever to make use of IMS assets from the Web. With Info 2.0, IMS is proudly riding the Web 2.0 wave, enabling collaboration among customers and partners.
What do bell-bottoms, low-carb diets and hierarchical databases all have in common? They entered the scene in the 1960s and became immensely popular in the 1970s. Since then, both bell-bottoms and low-carb diets have waxed and waned in popularity, as per the usual trends in fashion and diets.

And what of hierarchical databases? Since its introduction in 1969, IBM Information Management System (IMS) has been hugely successful as a premiere hierarchical database system, and remains one of the most heavily used systems today. Yet, with the current popularity of relational databases and the technologies and tooling surrounding them, one could easily argue that IMS and the hierarchical database have fallen out of fashion (like bell-bottoms).

However, with a new breed of data interchange standards on the scene, the times are (once again) a-changin’ and IMS finds itself in a unique position among the established industry databases. In IMS V9 and IMS V10, we added new functions to help you ride the next wave of database technologies.

**New horizons for IMS**

In the late 1990s, IMS began to invest heavily in new technologies that can help you make the most of your IMS databases. With the arrival of SOA, we have further increased our efforts to adapt IMS to the business needs of today. Among the new technologies on IMS are JDBC, SOAP, and MFS web services.

However, it is another new technology that is making the biggest impact in restoring IMS to the height of fashion. That technology is the extensible markup language or XML.

**Storing XML data? Enter IMS**

XML is powerful. As a general-purpose markup language, it:
- Can be used to represent almost anything
- Is platform independent
- Is easily parsed
- Handles encoding and byte ordering.

In a relatively short time, XML has become the de facto standard for data interchange.

However, what about storing data as XML? As more businesses are beginning to use XML to transfer data, we are seeing a shift towards the use of XML databases. This makes perfect sense — why bother with all the confusion of converting data between XML and your native format when you can just store data as XML in the first place.

The relational database developers’ community has expended great effort to find easy solutions to store XML data. However, XML by nature is hierarchical, and is therefore conceptually difficult to fit into tables and columns. At the same time, a new breed of native XML databases have emerged, based upon (you guessed it) a hierarchical database model.

The sudden resurgence of hierarchical databases has placed IMS in a unique position among established databases in the industry. IMS has been dealing with hierarchical data since the 1960s. Starting in IMS V9, IMS has extended its hierarchical support to allow you to directly store and retrieve XML documents in your existing IMS databases.

IMS V10 further enhances these XML capabilities with support for the XQuery language, allowing users to run XQuery 1.0 standard expressions against unchanged IMS databases.

**IMS as an XML database**

The key to enabling IMS to act as a native XML database lies in establishing the relationship between the hierarchy of the XML data, and the hierarchy of the IMS database. Because of the similarity in their hierarchical natures, we can create a mapping between the IMS metadata and the XML metadata. With the help of our tooling, the DLIModel utility, you can use IMS database definitions (DBDs and PSBs) to generate a default XML mapping automatically. This mapping is captured within the annotations of a generated standard XML schema.

The XML schema plays two roles in the IMS XML DB implementation:
- It describes the XML layout of the IMS database in terms of tags, elements and the hierarchical structure. Simply put, it shows what the IMS records look like as XML documents.
- It validates the XML documents that can be stored into the particular IMS database. If an XML document is valid to the XML schema (it abides by the schema’s defined rules), it can be stored into the IMS database.
The ability to store and retrieve XML documents allows IMS to ride the next wave of database technology. Yet IMS XML DB has another trick up its sleeve, turning what looks like a negative (the ability to support only valid XML documents) into a strong performance advantage.

When you have XML documents valid to a particular XML schema, that schema already describes a lot of the information in an XML document — things like the tag names, the hierarchic structure, and even some of the fixed data values. As a result, these static tags and values can be simply discarded, leaving the need to store only the pertinent information. This pertinent information includes the actual data value in the elements, the attributes, and the number of occurrences along the hierarchical path. All these can be stored in a standard IMS record. See Figure 1.

We call this process decomposing an XML document, and it follows these steps:

1. Validate the XML document against the schema
2. Discard the XML tags
3. Convert XML types to IMS defined types
4. Group all one-to-one values as fields within particular segments
5. Transform all one-to-many values as segment relationships
6. Store these segments in the correct hierarchical order as an IMS record.

As you might guess, the retrieval process of the XML documents is simply the reverse of decomposition. As the IMS data is fetched from the database, the XML schema restores all the previously discarded information to the XML documents.

Removing the redundant information that is already in the XML schema and only storing the remaining data makes IMS XML DB an efficient solution for storing and processing XML data.

Another benefit of decomposing is that it makes the XML data look like traditional IMS data. That is, IMS does not distinguish between the data from an XML document and data inserted using traditional IMS applications. As a result, IMS XML-enabled databases work the same way traditional IMS databases do. The databases inherit all the great functionalities and qualities of IMS, the very things that have made IMS successful in the first place.

**IMS learns a new language**

The XML query language known as XQuery was developed by the XML Query Working Group of the World Wide Web Consortium (W3C). In January 2007, XQuery 1.0 officially became a W3C recommendation.

As of IMS V10, IMS supports the XQuery language. You might ask, "Doesn’t IMS already have a query language of its own? Hasn’t DL/1 been the standard language for accessing IMS data since the beginning of the product?"

Yes, of course. However, with the addition of the IMS XQuery support, we can now leverage this emerging technology to retrieve XML documents from IMS more efficiently. This new implementation comes very naturally because, just as IMS XML DB correlates closely to traditional IMS databases, XQuery is also very similar to DL/1.

XQuery uses XPath expressions to address specific parts of an XML document.

If we look at a sample XPath expression to retrieve a particular book element within a bibliography XML document, we can see that it almost looks like the DL/1 SSA statement retrieving the same information from an IMS database:

```
XPath
/bib/book[publisher = "IBM Redbooks" and year="1991"]
```

The similarity between the two query languages all stems from the fact that they are both querying data within hierarchies. Our implementation of XQuery takes advantage of this relationship. In fact, the IMS XQuery engine can even process embedded SSA statements within XQuery expressions for enhanced functionality.

**The more things change...**

Trends come and go. And some trends, like bell-bottoms and low-carb diets, eventually come back. This phenomenon happens in the technology world as well. The hierarchical database is one example of a technology that faded in popularity, but now, with the advent of the XML database, is once again surging.

IMS Development continues to focus on what our customers have always valued: performance, reliability, manageability, and recoverability. This constant attention to customer needs, despite the rise and fall of the latest trends, has made IMS one of the most reliable, performance oriented hierarchical databases available today, and leaves us in a unique position to take advantage of the rise of XML and SOA.

---

**Figure 1. Decomposed XML storage in IMS**
Over the past few years, energy efficiency in the data center has become one of the top concerns for IT managers. As the cost of power grows significantly, the application of energy efficiency to systems performance becomes an important metric. IBM has long been a champion of energy efficiency and the IBM System z9 with z/OS is an outstanding platform for the integration of power efficiency and performance.

**IT costs going up**

Controlling IT costs is the main reason most businesses give for consolidating servers. Here are several dimensions that make up the total IT cost picture:

- **Hardware costs**: Reducing the number of server, storage devices, peripherals
- **Software costs**: Lowering software licensing and renewal costs, operating servers, and supporting software
- **Support staff**: Reducing the number of human resources needed to administer or manage the environment, as well as ongoing training costs
- **Operational costs**: Reducing costs of floor space, power consumption, cooling servers
- **Hidden costs**: Lowering costs associated with inconsistent or incompatible hardware and supporting software, server maintenance, server failures, data loss, and security exposures.

This article discusses a power efficiency and performance benchmark study on the IBM System z9 to show how adding workload to an existing z9™ server is a frugal energy management strategy. Details of the benchmark configuration are available upon request.

**System z9 Java measures up**

The IBM System z Performance team in Poughkeepsie, New York, and Bangalore, India, conducted a series of System z9 Java measurements in 2007. They measured z9 Enterprise Class (EC) model S08 and z9 EC model S18 servers with one and two processor books respectively. Both of these machines were configured with a single I/O cage. The number of processor cores (CPs in System z terminology) that the team used during measurements varied by a factor of four. The team ran a Java workload on z/OS V1R8 that drove processor utilization above 99%.

The IBM System z9 provides internal capability to monitor actual power consumption and temperature of the server. Power Monitor is an additional function for the System Activity Display on the z9 EC Support Element and Hardware Management Console. It displays watts and BTUs per hour as well as cooling air input temperature.

The team took power measurements using the Power Monitor, taking two power display points on every measurement, one prior to execution and the other while driving the processor above 99%. Table 1 shows a summary of the study’s results.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Processor cores</th>
<th>Throughput</th>
<th>Idle kW</th>
<th>Busy kW</th>
<th>Throughput / Busy kW</th>
<th>Throughput / Marginal kW</th>
<th>Idle BTU / Hour</th>
<th>Busy BTU / Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>2094-704</td>
<td>4</td>
<td>5.0</td>
<td>3.752</td>
<td>3.916</td>
<td>1.3</td>
<td>30</td>
<td>12,802</td>
<td>13,361</td>
</tr>
<tr>
<td>2094-708</td>
<td>8</td>
<td>9.0</td>
<td>3.919</td>
<td>3.987</td>
<td>2.3</td>
<td>133</td>
<td>13,372</td>
<td>13,604</td>
</tr>
<tr>
<td>2094-712</td>
<td>12</td>
<td>12.4</td>
<td>5.678</td>
<td>5.973</td>
<td>2.1</td>
<td>42</td>
<td>19,357</td>
<td>20,380</td>
</tr>
<tr>
<td>2094-716</td>
<td>16</td>
<td>14.9</td>
<td>5.727</td>
<td>5.920</td>
<td>2.5</td>
<td>77</td>
<td>19,541</td>
<td>20,199</td>
</tr>
</tbody>
</table>

Table 1. Summary of power measurements
Observations

The first observation is that running the workload has a very small impact on the amount of power used. In fact, the difference in power displays from low processor utilization to high processor utilization was within the cost of an additional light bulb or two, as shown in Figure 1, “Throughput and kilowatt (kW) usage.”

You can also see this observation in Figure 2, “Throughput per marginal kW transactions” for marginal or incremental kilowatts (kW). In Figure 2, throughput is scaled by the extra power that is used to run the workload over that required by an idling server. This figure shows how little power is actually required to run work once the machine is up and running. This means that it takes very little extra power to consolidate work onto a machine that has the capacity to handle the work.

The next observation is that increasing the workload as we moved from 4 to 8 processor cores on a model S08 or 12 to 16 processor cores on a model S18 made very little difference in terms of the power consumed. Increasing the number of cores within a model does not significantly increase the power. See Figure 1.
Going from an 8-processor core on our model S08 server to a 12-processor core server moved us from a model S08 server to a model S18 server. This change results in approximately a 10% decrease in transactions / busy kW. However, as previously noted, the additional growth potential going to a 16-processor core server requires very little extra power. In these measurements, the 16-processor core server has the best transaction / busy kW ratio. See Figure 3, ”Throughput per busy kW transactions.”

**Energy efficient scalability**

This work demonstrates the scalability of the server while power consumption remains relatively fixed; the marginal or incremental power requirements to run extra work are very small. Based on this analysis, consolidating servers into an existing System z9 clearly proves to be an efficient energy management strategy.

For additional information about how energy management is part of an overall server consolidation strategy, see A Power Benchmarking Study on the IBM System z9: Applying Energy Efficiency Metrics to Performance at the following Web site at: [ibm.com/support/techdocs/atmsastr.nsf/WeblIndex/WP101110](http://ibm.com/support/techdocs/atmsastr.nsf/WeblIndex/WP101110)

Thanks to Rathi Vakkayil, Development Software Testing, IBM Systems and Technology Group, Bangalore, India.
Riddle me this
Solving common riddles to good application performance

BY MARK WISNIEWSKI, ED MEKEEL, AND JUDI BANK

It’s an absolute classic. I mean who could ever forget the Batman TV series. The series ran only three seasons (1966-1968), but the reruns seemed to go on for years. The show had a huge impact on a generation of TV viewers. You can still hear quotes from the series to this day. One of the most memorable of Batman’s adversaries was the Riddler (played by Frank Gorshin). Flanked by his evil henchmen, he would allude to his sinister plot by posing a seemingly unsolvable riddle to Batman and Robin. Here’s one from season three:

“When does a painter use a trigger instead of a brush?”

You couldn’t help but agonize along with the “Dynamic Duo” as they summoned all of their logical thinking to arrive at the correct conclusion — eventually foiling the Riddler’s best-laid plans. Sometimes, writing code that performs well under z/OS can seem to be fraught with riddles too. We hope this article provides you with general performance guidelines to help minimize some of the puzzling aspects of writing code that performs well.

POW! Architectural considerations
Like Batman and Robin making the right decisions in building a plan to capture the Riddler, making the architectural decisions early in the design phase can often be the major influence on performance. Sometimes applications are designed to exploit new trendy frameworks that seem strategic, but that add overhead and are unsuited to the application. Here are a few Bat-pointers for avoiding this problem.

Avoid methods that necessitate calls to multiple lower level routines
- Inline where it makes sense
- Perform some transformations on the client (for example, xml parsing or codepage translation)
- Avoid wrappered data types, which cause cascading constructors.

Be aware of unintended locking problems
- Granular locking results in too many calls to the lock service, but if not granular enough, locks are held longer and can cause contention.

Order methods according to the most probable branch
- Don’t embed exception handling in the main path of the code because this causes excessive hardware cache misses.

Use security calls wisely
- Avoid digital signatures and expensive hashes between components executing in the same server image.
- Have multiple security options that can be selected according to the needs of the user.

Task switches are expensive
- Task switches (context switches) cause processor cache misses and multiple dispatches, which are both costly.
- Try to design the structure so a unit of work completes without traversing many threads. For example, don’t create an architecture that requires multiple execution phases with a task switch for each.

Avoid creating pipelines with a single point of contention
- For example, don’t have a single receiver thread or a single sender thread for applications with multiple worker threads
- Prototyping is a good way to evaluate technology alternatives.

WHAM! Programming practices
Every programmer will find something in the list below that will fit neatly into his or her “utility belt”, to be pulled out when rescuing an important module from the fiendish clutches of potentially poor performance. Armed with the right weapons at coding time, you can avoid falling into traps that could add unnecessary cycles and path length.

Avoid unnecessary file manipulation on every request
- Don’t open, read, and close a configuration file for every transaction — read it into storage
- Provide a dynamic method of updating the file without re-initializing the environment
- Don’t read in the same data over and over — cache it in storage
- Save the output from SQL queries instead of rerunning the query.

Avoid copying data and character-by-character operations
- Don’t read, write, format, transform or send a character at a time
- Use a buffer
- Share immutable strings instead of copying them
- Avoid parsing and data transformation where possible
- Pass data by reference, not by value.

Don’t call expensive services inside a tight loop
- For example, fork(), open(), read(), malloc(), free(), create_thread().

Use Pooling
- For example, pool sockets, connections, threads, handles, and mutexes instead of destroying and recreating them.
Struts forwards add overhead
- Don’t include empty actions that cause additional instructions to be performed unnecessarily.

Complexity drives defects and higher development cost
- Simplify where possible.

Optimize frequently called methods
- Use profiling to find out where the “hot spots” are and make sure the code follows performance guidelines and best practices for your specific source language.

Document your program to make it easy to maintain
- If code is complicated or obscure, this can hurt future optimization efforts. New programmers are often afraid to remove or optimize inefficient statements when they don’t understand them.

Use efficient I/O and print formatting
- In both Java and C/C++, the type of I/O constructs (such as stream I/O) makes a difference in the efficiency of the code.

Avoid non-linear algorithms
- For example, a linear search performed multiple times as the number of elements grow (such as a search through all users done once per user).

Avoid APIs that are too generic
- Generic APIs encourage excessive layering.

Take note of the number of bytes sent and received in a transaction
- Avoid large messages.

Use the highest level of compiler optimization when possible
- Some forms of compiler optimization can make programs harder to debug. This can happen when the compiler rearranges generated assembler statements to make the code more efficient, but the resulting assembler cannot be matched with the originating source statement. Under these circumstances, you may wish to trade off performance for maintainability.

BAM! Performance sensitive thinking
Batman always asked the good citizens of Gotham City to be on the lookout for the Riddler and his henchmen, and to be wary of their devilish schemes. The following is a list of performance-related considerations to keep in mind when writing and designing a new piece of code.

Ask yourself, will this code result in:
- A substantial increase in CPU time (instructions) of an existing function
- A substantial CPU time increase while a lock is being held
- Any increase in I/O
- Any increase in 24 or 31 bit common storage areas
- A substantial increase in memory usage
- Any increase in getmain, freemain, or storage obtain and release frequency
- Opening and closing the same file(s) repeatedly
- Creating a new thread for every transaction instead of using a pool of threads
- Processing the same data repeatedly (e.g. parsing, translating) without saving it
- Calling functions inside a loop when they should be called once outside the loop
- Unnecessary copying of strings or data instead of using pointers
- Forking a new process on each call or transaction
- Creating a new JVM on each call or transaction
- Creating a new security context on each call or transaction
- Using unbuffered writes, especially for trace and log?

Tune in again
Holy bottleneck, Batman! As you can see, there are many things to think about when writing application code to run on z/OS. Failure to keep these things in mind can lead to very bewildering performance issues. We are constantly testing performance scenarios, analyzing our findings, and documenting the results. So “tune in” again (same Bat-time, same Bat-magazine) for more tips on good performance.

Oh, by the way, the answer to the riddle in the opening paragraph (I’m sure everyone already got it) is, when he’s a stickup artist. ■
It just makes sense

Using Rational Developer for System z

BY JOHN DIAMOND AND RONALD TOWNSEND

IBM Rational Developer for System z is an extension to the IBM Rational Application Developer platform that is specifically designed for System z development. For developers who have been using Rational tools, but are new to System z, this product can be an incredible productivity enhancer. It puts development, file management, and job management in the same integrated development environment. For developers who prefer a GUI editor with a point-and-click interface, Rational Developer for System z provides it for System z development. (In case you are wondering, Rational Developer for System z was previously called WebSphere Developer for System z.)

Rational Developer for System z is truly a multifaceted development tool. It contains everything you need to develop Java and Java 2 Platform, Enterprise Edition (J2EE) applications. And, when you open the z/OS perspective, it is an integrated mainframe development environment.

Rational Developer for System z is not just for developers new to System z. A study, “Comparing IBM WebSphere Developer for System z to Traditional IBM Mainframe Development Tools,” conducted by the Branham Group Inc in November 2006, demonstrated that Rational Developer for System z provides significant productivity gains over traditional mainframe tools that perform comparable functions. For details, see www.branhamgroup.com/media/articles/wdz_study.pdf.

This article highlights some of the features of Rational Developer for System z that can help simplify designing and developing enterprise applications.

Developing PL/I, COBOL, and C/C++ programs on your PC
Rational Developer for System z provides a workstation development environment for PL/I, COBOL, and C/C++ programs. You can build, run, and debug these programs on your PC.

To deploy these programs on System z, you drag the source code from a local project folder to a remote partitioned data set (PDS). This process uploads the file to z/OS. You then right-click the source code and select Generate JCL to generate JCL for compiling the job — this is a tremendous productivity benefit for someone new to JCL.

The rest of the process can be just as simple:
- To submit the job, you right-click the generated JCL and select Submit from the pop-up menu.
- To view the output of the job, you navigate to the JES folder and look at the top of the list of jobs under the My Jobs filter.
- When you open a job log, it is downloaded to Rational Developer for System z.

Accessing files on the mainframe
Figure 1 shows an example of the Remote Systems view, which allows you to access to JES jobs, MVS data sets, and z/OS UNIX System Services (z/OS UNIX) files. When you click any of these artifacts, it is loaded into the Rational Developer for System z editor.

Rational Developer for System z also provides a JCL editor complete with color-coding and automatic syntax checking. This editor gives you immediate feedback on errors, which can be helpful if you are just learning JCL.

Figure 2 shows an example of a JCL file being edited in Rational Developer for System z. You can also use the editor to get the results of your job quickly without having to leave the development environment. As shown in Figure 2, the editor includes a tab you can click to load the output from a job.

Writing C programs for z/OS UNIX
Rational Developer for System z provides great features for writing C programs for z/OS UNIX. As shown in Figure 1, you can use the Remote Systems view to manage files in your z/OS UNIX file system. The C editor provides color-coding and content assist to help fill in function and variable names. With a click of the mouse, you can compile the C source file being edited on the remote z/OS system; with another click of the mouse, you can run the remote program in the z/OS UNIX file system. The output is displayed in a Rational Developer for System z window.

For developers who want GUI editors, these capabilities can be easier than using FTP to move the files, and initiating a compile job on the remote system.
Debugging applications
With Debug Tool for z/OS, Rational Developer for System z provides a complete debugging environment for local and remote applications. You can step through high-level source instructions, set break points, and watch variables. Rational Developer for System z supports dbx, which enables you to examine, monitor, and control z/OS UNIX programs written in C, C++, and High Level Assembler.

Enabling CICS applications as business services
The service flow project tools, a component of Enterprise Service Tools, enable business service integration for CICS applications. The tools are designed to help you leverage your existing investment in your core business applications, the quality of service of existing enterprise information systems (EIS), and the CICS runtime environment while enabling the move toward service-oriented architecture (SOA).

The service flow project tools provide graphical modeling workspaces that enable anyone who needs to implement SOA (such as enterprise solution architects, integration developers, and CICS application specialists) to create business services that can be invoked as Web services. The tools compose a sequence of CICS application interactions, and then a generator transforms these interactions into a CICS business service that can be deployed to IBM CICS Transaction Server, Version 3.1. To simplify the process, you can use the new service flow wizard to help generate these business service flows.

Updating BMS map sets
Using the BMS Map Editor, you can create and modify basic mapping support (BMS) map sets, which are used to define CICS screens. You can add and arrange fields and maps within a new or existing map set. The display and preview modes provide robust viewing tools, including a filtering option to show specific map combinations. In source mode, you can directly edit the code in the map set file.

Two wizards provide additional support:
- The map set wizard helps you generate and configure new BMS map set files.
- The export wizard helps you create a Java Server Faces (JSF) Web page for each map.

Using the BMS Editor, you can also change defaults for palette items, generate JCL for .bms files, generate .bms files directly on a remote PDS, and print from the Design canvas. In the Palette view, you can use design-page selection tools, such as click-and-drop and drag-and-drop, to access the BMS map components.

Developing services and flows
Enterprise Service Tools (EST) is an integrated perspective that helps you create various types of Web services. It combines capabilities that were formerly available in the XML Services for the Enterprise plug-in and the Service Flow.
Modeler plug-in. Enterprise Service Tools features an enhanced user interface that increases usability and simplifies the overall development process for services and flows. EST also incorporates z/OS DB2 support through the z/OS Database Application Generator.

Enterprise Service Tools can generate the artifacts that are needed to enable existing IMS COBOL applications to run as Web services in the IMS SOAP Gateway environment. The IMS SOAP Gateway is a lightweight Web service solution that enables IMS applications to interoperate outside of the IMS environment through SOAP components.

An integrated development environment
By integrating these tools in a single development environment, Rational Developer for System z is intended to help improve your productivity and make application development more enjoyable.

Answer key to quiz on page 54

1. (a) RAS refers to “reliability, availability, and serviceability”. There is no IM abbreviation RAS; to end a chat, you could just type “G2G” (Got to go).

2. (b) Avid IMers are laughing out loud if you missed this one. There’s no such thing as a Log Output Lister, but you can use the System Display and Search Facility (SDSF) to view and search the system log (in addition to other cool features).

3. (c) PDS is an abbreviation for both “partitioned data set” and “please don’t shout”.

4. (a) Integrated Coupling Facility. If you don’t understand what someone’s saying when IMing, you could just use a “?”.

5. (b) Very sad face. The description given for the nonexistent “Virtual Storage Facilitator” is actually a description of Virtual Storage Access Method (VSAM).

6. (c) IMS is an abbreviation for both “Information Management System” and “I am sorry”.

7. (a) There is no IM abbreviation for “stay cool like me”, although SC (stay cool) is used.

8. (b) The description given for the nonexistent “Remote Management Environment” is actually a description of the Geographically Dispersed Parallel Sysplex (GDPS).

9. (c) CTC is an abbreviation for both “channel to channel” and “care to chat”.

10. (b) Talk to you later. Perhaps you were thinking of tty, the UNIX command that performs input and output on a character-by-character basis? You’d better have a look at “z/OS UNIX terminology made easier to swallow,” on page 11, for a quick refresher on UNIX terms.
We hear you

Your communications serve z/OS Communications Server

BY MICHELE CARLO

The z/OS Communications Server combines the well-known functions of Systems Network Architecture (SNA) with the dynamics of Internet Protocol (IP) connectivity.

As you might have noticed, it takes a lot of information to describe how it all works together!

We hear that you often have to wade through volumes of information to customize configurations and resolve problems. In particular, we hear one comment all the time: Too much information, but not enough information tailored to a particular task. z/OS Communications Server has a good reference library, but you would prefer that more of this information be presented in a more task-based format.

To begin to address this need, we asked IBM Service personnel for z/OS Communications Server IP to examine problem management records (PMRs) to identify the areas in which the lack of clear topic-based or task-based information was inhibiting your ability to solve problems. We learned that the three areas in which most people have problems are:

• Connecting to the network
• Responding to FTP connection timeouts
• Diagnosing name resolution problems.

True, information about these topics already exists in our documentation, but is spread across several books, and is not always presented in a step-by-step or procedural format. It would be easier if all of the information on such key topics appeared in one place, presented in easy, cookbook-like instructions.

To improve the usability of our information, teams of IP service personnel, programmers, and technical writers met regularly to create short, procedural documents for each of these problem areas, as follows:

• “Adding an interface” describes how to add an interface and configure a device in VTAM® or TCP/IP for dynamic or static routing.
• “Fixing common FTP problems” describes what to do about problems you encounter during connection establishment or data transfer, for both the client and the server.
• “Diagnosing name resolution problems” describes symptoms that your system might experience, how to determine whether you have a name resolution problem, and what steps to take when you think you do.

You can find these documents using the title keywords on the support Web site at ibm.com/software/network/commserver/zos/support. Eventually, we hope to incorporate this information into the z/OS Communications Server product library.

Those of us responsible for z/OS Communications Server customer information understand that having the right information is critical to your ability to accomplish tasks in a minimum amount of time. We plan to provide more task-based information in the future.

IPCS class acts

Would you like to improve your ability to diagnose z/OS or z/OS Language Environment problems? An intermediate workshop on using the Interactive Problem Control System (IPCS) can help.

• Have you ever wished you knew more about diagnosing a z/OS error? Do you want to know how to analyze a program check, debug a loop, or set an effective SLIP? The PK100 z/OS Diagnostic Workshop in Poughkeepsie, NY has everything you ever wanted to know about z/OS debugging but didn’t know whom to ask! In this intermediate level class, you will learn these diagnostic skills and all the useful commands in IPCS.

• Want to learn z/OS Language Environment diagnostic skills? Want to know about condition management under Language Environment? Do you know about FASTLINK and XPLINK? Can you locate your parameters on the Language Environment stack? Are you able to diagnose Language Environment HEAP overlays? Do you know how to read CEEDUMPs? Do you know how to use LEDATA or OMVSDDATA? The PK300 Diagnostic Workshop for z/OS Language Environment Applications class in Poughkeepsie, NY can help you learn these diagnostic skills from z/OS Language Environment support experts.

PK100 and PK300 include many labs that provide hands-on opportunities to apply your new knowledge to real-life scenarios. Both classes require basic familiarity with IPCS, z/OS fundamentals, and z/OS Assembler. See ibm.com/training/us, and then search on PK100 or PK300.
Rumors and hearsay usually don’t impress us here at the z/OS Hot Topics central office, but when we heard recently that a new z/OS Information Center was on the scene, we had to investigate. Evidently, this particular online resource has created quite a stir among customers — and that’s just in its beta form!

According to our anonymous source: “This new Information Center is full of z/OS base features and elements. All the PDFs are in one place! And, best of all, it’s searchable by Google!”

Could it be true?

Being the hard-driven z/OS reporters that we are, we naturally had to see it for ourselves. Being the relentless shutterbugs that we are, we wanted to do more than just tell you about it. And, being the software engineers that we are, we most definitely stopped for doughnuts and coffee along the way.

Stalking our z quarry

To uncover the truth, we went undercover, camera in hand. For you, dear reader, we had just one goal: to steal an early glimpse of the new z/OS Information Center — and escape with our necks and doughnuts intact!

Fortunately, we had very little trouble finding our quarry — the Google thing, remember? Wow, were we impressed! The new z/OS Information Center is a dream machine! Its quantity of z/OS product information is unmatched. Its searchable, easy-to-use interface is truly state of the art.

Unfortunately, before we could take more than a snap, the Information Center creators caught sight of us and gave chase. Apparently, a similar incident with z/OS paparazzi had made them wary of newshounds like us. They tore the camera from our photographer’s neck — but not before she stashed the film!

Here now — for your eyes only — is the world’s first exclusive photo of the new z/OS V1R9 Information Center beta (Figure 1).

It’s a beauty! The most popular links on the home page and key information components are precisely where we would want them to be!

Just before we were force to flee the scene, we had a few seconds to click madly on the Web site component tabs. There, we saw links to our favorite resources, arranged logically, including:

- LookAt for searching messages
- DOC APAR & PTF ++HOLD documentation
- IBM Redbooks and education modules associated with the component.

The quick eye among us even managed to memorize the Web site URL to share with you: publib.boulder.ibm.com/infocenter/zos/v1r9/index.jsp

That’s the scoop — now go see for yourself!

Figure 1. z/OS V1R9 Information Center beta
As the Test or Quality Assurance person at a z/OS installation, do you ever find it a struggle to explain your job to someone who—for one odd reason or another—does not work in the IT Industry? If so, what analogies might you use to explain the complexities of your work?

Without scaring the person silly, that is. Take the process of problem determination and re-creation, for example. How might we describe this job in terms familiar to the young and the uninitiated?

In truth, the job sometimes requires us to play all of the following roles and more:

• A detective, interviewing the wronged parties and rounding up clues
• A mathematician, measuring performance and throughput
• A scientist, injecting errors and random variation into a test configuration to surface errors before going into production.

Just for fun, let’s re-imagine the process of problem re-creation as it might apply to some other engineering role in society: a traffic engineer for a busy city. Along the way, we will re-affirm some tips to help you when you must recreate a problem in a real life z/OS environment (yours).

When re-creation isn’t playtime

Imagine that you are the chief traffic engineer for a major metropolitan city. Your boss, the city mayor, has just telephoned you to say that the city has a terrible problem with intermittent traffic jams at a certain intersection. The public is very upset—particularly the part of the public that votes. You need to find the underlying cause of the traffic jams—quickly!

Because you did not experience the problem firsthand, you might not know where to begin. For an intermittent problem, you might need to recreate the problem first—to understand it thoroughly—before you can resolve it.

Here begins the time-honored process known as problem re-creation. It’s one part Science, one part Art, and a complete challenge to do it right. After all, it is called problem re-creation (accent on the “re”—not recreation as in campgrounds and water skiing).

Tip 1: Listen to your users

Your first idea is to ask the city to experience a “few more jams” so you can gather better diagnostic data. Not surprisingly, this idea is not a big hit with the public (imagine how IBM Customer Support feels asking you to re-create a problem). Therefore, as an alternative (and interim) solution to the traffic jams, you ask to have a police officer stationed at the problem intersection to direct traffic manually. (Interestingly, since that was done, there have been no recurrences of the traffic jam.)

Now to look at some typical causes of problems like this.

Down-level product?

Your city uses the Omega-System Traffic Signal with the popular Three-Color enhancement installed. This system also includes the optional Timer feature, which enables City Planning to ensure that no motorist ever encounters two consecutive green lights on any given route (perish the thought).

Unfortunately, the installed version of the product is down-level by several releases. The traffic light vendor assures you that if your city upgrades to the latest level of the system, the problem will end. However, there is no money in the budget to upgrade the system this year.

User error?

Researching traffic records indicates that traffic jams can result from motorists talking on their cell phones and not paying attention to the roads. Interesting data, but what is the connection to the problem intersection?

You file this tidbit away under Maybe and continue hunting for clues.

Tip 2: Narrow the list of possibilities

Too many variables; you need to narrow the list. You investigate further and discover that the traffic jam rate at the problem intersection is 100 times worse than that of any neighboring intersection. Because the same drivers have to pass through at least one neighboring intersection before reaching the problem intersection, and there are no similar problems at the other intersections, something about that particular intersection is causing the problem.

Tip 3: Test with realistic conditions

To trap the problem, you need to be both scientific and crafty. You cordon off some city streets having a similar layout as the problem intersection (translation: configure a test bed for recreating the problem). Also, you use the same traffic lights (that is, you match the software and hardware product levels that are used in the actual system).

You are methodical, and recruit test motorists driving a realistic variety of cars, including many big sport utility vehicles (SUVs). The City Council does not see the need for this attention to detail, but they approve your plan anyway.

The traffic light system’s Three-Color enhancement has been solid in the field, but your city added the Timer feature as a field replacement and there have been some problems reported for it. Therefore, you decide to run some tests with the Timer feature disabled.
Tip 4: Test with realistic workloads
You run the testing for a week without experiencing any traffic jams. Not a surprise, perhaps — you tested this traffic light system thoroughly before putting it into production several years ago.

Then, you notice something crucial: The Test intersection never seems to have any motorists waiting on the cross streets. To force the problem to reoccur, therefore, you must add more traffic to the test system.

You step up the activity. Another week passes, but still nothing. Did you add enough cars? How would we know for sure?

After some measurements, you discover that the test vehicles are making only right turns, due to some settings the last time the workload was run. You realize that, even with all this activity, there is no conflict, no delay because all of the test cars are turning right even on red! Therefore, in performing this particular test, you were merely spinning your wheels (apologies for the pun).

Tip 5: Analyze your path coverage
To assist you with your testing methodology, the vendor support representative advises you to analyze the path coverage. That is, ensure that your testing is exercising all of the critical paths that are used in the actual workloads.

Tip 6: Practice random acts of varying
You devise a randomizer that determines whether the test vehicles are to go straight or turn left. You omit right turns; that path is not interesting if you want to measure contention over the middle of the intersection.

Tip 7: Use first failure data capture
Finally, failure — which, in this case, means success! The next morning, you see that there was a jam during the overnight run. Then, you realize that you cannot figure anything out about why the Test jam happened or even when it happened. Thus, you will have to recreate the problem again. Here, you are reminded of why there is a flight recorder on an airplane: There is no second chance to get data, there is only first failure data capture (FFDC).

Let’s re-imagine the process of problem re-creation as it might apply to some other engineering role

Measure the real life performance
You decide to measure the actual traffic activity at the problem intersection. In doing so, you discover that an average of 5,000 vehicles per hour cross the problem intersection. Your test, however, uses only 1,000 cars. Good thing, you checked!

You start running 5,000 vehicles. Then 7,500, then 10,000. Still no traffic jams result in the Test intersection.

There must be another factor!

Match the actual configuration
You have a crucial insight: The real streets are two-way, but the Test system, while similar in layout, was configured with one-way streets. You reconfigure the Test streets to match the real intersection, and test again. Still nothing.

Tip 5: Analyze your path coverage
To assist you with your testing methodology, the vendor support representative advises you to analyze the path coverage. That is, ensure that your testing is exercising all of the critical paths that are used in the actual workloads.

Therefore, you install video cameras in the test vehicles to record their view of the signal and intersection with a time stamp and a trace facility in the signals to record every light change with a time stamp.

The next morning, the cars are lined up for miles in every direction. This time, however, you have obtained the diagnostic data you need: It shows a clear pattern of displaying red lights on both cross streets just long enough for the queues to reach critical mass, so that it cannot be relieved by a return to the normal signal pattern.

The traffic light vendor examines your traces and finds a bug in their code. They send you a fix, which you apply to the Test system. You run tests with the fix for 24 hours, with the traces turned off, and sure enough, the intersection is now properly serialized. You send the fix to the city traffic department, and the problem is resolved, this time for real.

Moral of the story
Debugging traffic jams as an allegory for problem re-creation? Absolutely. Here is the truth behind the fiction — in seven, easy-to-remember tips for improving your problem re-creations:

• Listen to your users
• Narrow the list of possibilities
• Test with realistic conditions
• Test with realistic workloads
• Analyze your path coverage
• Practice random acts of varying
• Use first failure data capture.

Let’s re-imagine the process of problem re-creation as it might apply to some other engineering role

The End
You’ve heard about Lightweight Directory Access Protocol (LDAP) before and you’ve probably used it in more than one place. There are LDAP services available on multiple operating systems and tied to various applications to store data. Heck, there’s even a version of LDAP that runs in e-mail clients that retrieve, organize, and store your messages. So with this many LDAP clients and solutions out there, what’s so special about z/OS LDAP?

z/OS LDAP: Vital statistics
Well, z/OS LDAP is more than just another pretty face in the security crowd. z/OS LDAP does more than just the standard LDAP protocol. With support for your Parallel Sysplex environment, LDAP on z/OS is exceptionally suited for handling your security architecture. Let’s see what some of the capabilities are.

Like all LDAP implementations, z/OS LDAP is based on a client/server model that provides client access to an LDAP server. The LDAP server maintains directory information in a central location for storing, updating, retrieving, and exchanging. The advantage of z/OS LDAP is that we provide several different repositories, called backends, to choose from to store the directory data:

- Two general purpose backends, TDBM and LDBM, that can store any directory information
- A security backend, SDBM, that provides access to RACF directory information
- A specialized change log backend, GDBM, that contains information on changes to entries in the other backends.

With this many options to choose from, we’ve got you totally covered.

Figure 1. LDAP server overview
TDBM: The power of DB2 data sharing

The TDBM backend stores its directory information in a DB2 database. This backend is intended for medium to large directories where the scalability of DB2 is required. The DB2 database can be owned by a single LDAP server or can be shared between all LDAP servers running in a Parallel Sysplex group. Talk about power! TDBM uses DB2 locking to synchronize updates across the Parallel Sysplex group and ensure referential integrity of the directory information. This capability also allows the z/OS LDAP server to act as a true multi-master configuration (each server has read/write access to the directory) with workload balancing across the Parallel Sysplex, without the overhead of typical LDAP replication scenarios. There is no need to worry about synchronization of the directories or conflict resolution due to concurrent update operations to multiple replicas. With z/OS LDAP, you simply don’t have to worry about anything.

LDBM: Fast and simple

LDBM uses a z/OS UNIX System Services file system to store its directory entries, so you do not need DB2. When the LDAP server is running, LDBM caches all its directory entries in memory to help with performance. The LDBM backend supports small to medium-sized, general-purpose directories. It is simple to set up because it does not require any additional program products. Like TDBM, LDBM provides data encryption, replication, and Parallel Sysplex support as well as support for all SASL bind mechanisms.

SDBM: The long arm of RACF administration

The SDBM backend provides read/write access to RACF user, group, and connection profiles, using the LDAP protocol. (RACF is a component of the Security Server for z/OS.) Because you can use any LDAP client to communicate with the z/OS LDAP server, SDBM gives you the ability to perform RACF user and group administration from AIX®, Windows NT®, SUN Solaris, z/OS, or wherever; keep in mind though that this is not a back door around RACF security. The LDAP client has to present a RACF user ID and password that RACF authenticates, and then RACF controls what the client can do. If the RACF data is shared across the Parallel Sysplex, you can use z/OS LDAP to manage users, groups, and connections in the Parallel Sysplex. We keep it simple and easy to manage, but also secure. And isn’t that what you really want from LDAP?

GDBM: Ch-ch-ch changes

You can configure the LDAP server to create change log entries in the GDBM backend. Each change log entry contains information about a change to an entry in a TDBM or LDBM backend, to the LDAP server schema, or to a RACF user, group, or connection profile. These change log entries allow you to track the changes that have occurred in your directories. You can choose to have GDBM store its entries in DB2 (the same way TDBM does) or in z/OS UNIX System Services files (the same way LDBM does). Another great thing is that these change log entries are treated like any other LDAP entry, which means that they can be searched and even deleted, all using the LDAP client. (But keep in mind that they cannot be modified! We’re secure, remember?).

Parallel Sysplex support enablement

One unique thing that z/OS LDAP does is to enable multiple instances of an LDAP server to run simultaneously. For instance, an LDAP server running in a Parallel Sysplex environment supports multiple instances of the same server within a cross-system coupling facility (XCF) group. Each server instance runs on a separate system but shares the same schema and backend databases. (Actually, if your server has multiple backends, you can choose to share some of them and keep others private). How cool is that? This provides improved workload management when the TCP/IP Parallel Sysplex distributor is used to route incoming connections to available LDAP servers within the Parallel Sysplex. Each server in the same group must be identical to the other servers in the group; therefore, any server in the group can satisfy a request. To make things easier, authentication and authorization definitions are the same for all servers in the same group so that a client who can authenticate and perform LDAP operations on one server in the group can do the same for any other server in the group.

z/OS LDAP and RACF: Best of friends

The z/OS LDAP server and RACF play nice together. We’ve already discussed how the SDBM backend can be used for remote administration of RACF and the GDBM backend can track changes to RACF profiles. But there is more to it than that. The LDBM and TDBM backends can be set up to use RACF passwords for authentication. And RACF can use LDAP to keep entire directories (including passwords) in sync with the RACF database. LDAP and RACF are true partners in the z/OS security field.

Native authentication: RACF watches out for LDAP

This security partnership extends in other ways too. Like a good friend, RACF works with LDAP toward a common goal, in this case authentication. Normally, a client authenticates to an LDAP server by specifying the name of a directory entry and the value of a password kept in the entry (usually in some encrypted form). In fact, LDBM and TDBM both support this sort of authentication. However, there is already a directory on z/OS that specializes in storing passwords — RACF. Wouldn’t it be great if a user could authenticate to LDBM or TDBM, but use a RACF password instead of a password in LDBM or TDBM? That is exactly what LDBM and TDBM native authentication support does. Instead of a password, the LDBM or TDBM directory entry contains the name of a RACF user. When the client authenticates using the directory entry, the password supplied by the client is verified by RACF against the password for that user. The client is authenticated by RACF with no password at all in LDBM or TDBM! One less password for the client to keep track of, one less place for z/OS to store passwords, and one small step for security. By the way, native authentication can use any security server on z/OS, not just RACF.
Our next event: Synchronized passwords

Finally, as if all this RACF goodness isn’t enough for you, try this on: you can synchronize your RACF passwords across all your registries! That’s right; there’s no more manual synchronization or multiple updates necessary. RACF and the LDAP server, in conjunction with the IBM Tivoli® Directory Integrator (ITDI) product, can now automatically keep all these passwords in sync. Here’s how it works:

1. RACF creates change log entries in the GDBM backend in the LDAP server when a RACF user, group, or user-group connection is changed. In particular, when a user password is changed, the change log entry indicates this (but does NOT include the password to protect its security).

2. ITDI periodically searches the LDAP change log for new entries. When it sees that a RACF password has changed, ITDI issues a search to the SDBM backend in the LDAP server to retrieve a secure envelope containing the password from RACF.

3. ITDI then decrypts the envelope to obtain the RACF password and updates other directories with the new password value.

You get to relax because all this work is done automatically!

LDAP has certainly come a long way since its humble beginnings. However, by leveraging the best that z/OS has to offer and by embracing stringent z/OS security controls and allowing the use of security credentials across your enterprise, it's clear to see that z/OS LDAP is the LDAP you want for your z/OS system.

If you are on z/OS, you should be on z/OS LDAP. Try it, you'll like it!

Acknowledgement

Kudos to Jay Brodfuehrer, z/OS LDAP Development, for contributing to this article.

z/OS C/C++ man pages now on the Web!

by Adam Lickel and Barbara Neumann

In response to popular request, online information about C/C++ functions is now available in the form of man pages at [ibm.com/servers/eserver/zseries/zos/le/manpgs.htm](http://ibm.com/servers/eserver/zseries/zos/le/manpgs.htm). The z/OS V1R9 C/C++ run-time library man pages are in EBCDIC IBM-1047 format.

Man pages provide instant access to online documentation without ever leaving the UNIX shell. Instead of searching a PDF or performing a bookshelf search, you can type `man malloc` and get the exact information you need.

It is common for UNIX systems to provide man pages for the C/C++ functions. It is especially relevant for z/OS given the number of extensions that exploit various z/OS features, such as data sets. The man page information is identical to the online documentation already available in BookManager® and PDF form but is more convenient and comfortable for a z/OS UNIX developer to use.

To install the C/C++ man pages:


2. Select a place to store the files. A good candidate would be `/usr/man/C` as any user in the POSIX locale would automatically pick it up without having to adjust his or her `MANPATH`.


We'd appreciate any feedback you have. You can use the feedback button on the Web site to send in any comments on using these man pages.

- For complete information on C/C++ functions, see z/OS XL C/C++ Run-Time Library Reference, SA22-7821.
- For additional programming tools, see the z/OS UNIX System Services Tools and Toys Web page at: [ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html](http://ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html).
Bob Abrams is an IBM Senior Technical Staff Member (STSM) in z/OS Development, with a focus on reliability, availability, and serviceability (RAS) and problem determination.

Judi Bank works in System z Performance at IBM Poughkeepsie. Before joining IBM in 1980, she worked as a mathematics professor at Bronx Community College and a systems programmer for several midrange operating systems on non-IBM hardware.

Debbie Benjamin-Deckert is a software engineer for VSAM RLS in DFSMS. She has 22 years of experience with VSAM. In z/OS V1R7, Debbie helped to design RLS 64-bit buffering.

John Brooks is a Senior Software Engineer at IBM Poughkeepsie, where he works on the z/OS UNIX logical file system.

Patrick Brown is a Senior Software Engineer in z/OS Communications Server in Research Triangle Park, where he has worked for 17 years. Currently, Patrick designs and develops functions for Enterprise Extender and High Performance Routing (HPR).

Michael Buechele is an Advisory Software Engineer in System z Performance at IBM Poughkeepsie. His 25 years of experience includes IMS, DB2, CICS, porting applications to z/OS UNIX, ISV enablement, and customer benchmarking.

Armin Buechler has been with IBM Boeblingen for 20 years, where he has worked on various z/OS components, including TSO/E, RMF, and WLM.

Yvonne Cardinal is a Staff Software Engineer at IBM Poughkeepsie and an information developer for RACF and TDS/LDAP. She has been with IBM for 34 years.

Michele Carlo has been writing information for IBM mainframe software products for over 20 years. She also manages a variety of z/OS Communications Server projects.

John Diamond is a software engineer at IBM Poughkeepsie, where he helps customers implement System z solutions. His previous projects include IBM Grid Toolbox and WebSphere Commerce Suite. John holds an MBA from NYU and a BA in Computer Science.

Sigrun Elsner is a member of the HCD/HCM Development team at IBM Boeblingen.

Paul Evans has been in the IT industry for 34 years, and has worked in the IBM Support Center for the last 13 years.

Jodi Everdon is an Advisory Software Engineer at IBM Poughkeepsie. She is a managing editor for z/OS Hot Topics, the Information Planner for z/OS DFS zFS and SMB, and the author of z/OS Problem Management.

Nicole M. Fagen is an Advisory Software Engineer at IBM Poughkeepsie, and the team lead for Parallel Sysplex Defect Support. Nicole focuses on XES, XCF and GRS, and is well versed in System Logger, RRS, consoles and IBM Health Checker.

Robert Fairlie started with IBM at the Owego location in 1981. His experience includes z/OS and z/VM® support, FSD procurement development, and the development and testing of SFS, LFS, and DCE/DFS. Robert currently leads the z/OS Java Batch Development team.

Erin Farr is a software engineer in z/OS UNIX Shells and Utilities Development at IBM Poughkeepsie. Her previous projects included porting UNIX applications to z/OS, including OpenSSH.

Mike Fitzpatrick is a Senior Software Engineer in z/OS Communications Server in Research Triangle Park, where he has worked for 12 years. He focuses on the design and performance of Enterprise Extender, IP security, and sysplex functions.

Garth Godfrey is a Senior Software Engineer in z/OS UNIX Development at IBM Poughkeepsie. With 28 years of experience, he leads the Shells, Utilities and OpenSSH team, and is a frequent speaker at SHARE.

Russ Hardgrove has worked in the IT industry for 35 years, and in IBM RACF Level 2 Support since 1999. He is a frequent contributor to the RACF-L discussion list, and has presented at the Vanguard Enterprise Security Expo and the IBM SecureWorld® conference.

Shyh-Mei Ho is the Chief Architect for IMS On Demand SOA solutions. She has received two IBM Outstanding Technical Achievement Awards and holds 30 patents. In 2006, Women of Color Magazine selected her as a Technology All-Star.

Jochen Hoffmann is a software developer at IBM Boeblingen and has been a member of the HCD/HCM Development team since 2002.

Karl-Hans Holder has been with IBM Boeblingen for 25 years. Before joining z/OS WLM Development, he worked in VSE Supervisor Development for 15 years.

Christopher Holtz is a software engineer in IMS Development at IBM Silicon Valley Laboratory, with a focus on Java, XML, SOAP, and emerging SOA technologies. He holds an MSE degree in Computer Science and Engineering from the University of Michigan.

Autumn A. Houseknecht is an Advisory Software Engineer in z/OS Software Defect Support at IBM Poughkeepsie, specializing in z/OS UNIX, NFS, and CIM. She holds a BS in Computer Science and an MBA from the University at Buffalo.

William Huber is a software engineer at IBM Poughkeepsie, where he specializes in Function Verification Test for z/OS UNIX.
Jenny Hung is a software engineer for IMS On Demand. Her expertise includes IMS SOAP Gateway, IMS Business Process Choreography, and IMS Control Center solutions. Jenny holds an MS degree in Computer Science from Stanford University.

Ken Irwin leads the z/OS Java Level 2 Support team.

John H. Kettner works in the System z Competitive Project Office at IBM Somers. He has 36 years of mainframe experience and a BS in Computer Science. John writes and lectures extensively on various mainframe subjects.

Randall Kunkel is a Senior Software Engineer in z/OS Communications Server Service and Development at Research Triangle Park, where he has worked for 18 years. His expertise includes Enterprise Extender and High Performance Routing.

Peggy LaBelle is the lead designer for ICSF. She has a wealth of experience in System z security, including RACF Design and Development.

Amy S. Lander is an Advisory Software Engineer at IBM Poughkeepsie. Amy is a technical writer of System z user information; currently, she develops content for the z/OS Basic Skills Information Center. In her spare time, she tests canine IQs. Woof!

Larry L. Law is a Senior Engineer for VSAM RLS in DFSMS. He has 25 years of experience with VSAM, and helped to design RLS 64-bit buffering in z/OS V1R7.

Jason Lee is a Staff Software Engineer for VSAM RLS and Transactional VSAM at IBM Silicon Valley Lab. He holds a BS degree in Computer Science from University of California, Berkeley.

Douglas Lehr is a Staff Software Engineer at IBM Tucson. He works with VSAM, Catalog, and VSAM RLS, and is currently the team lead for VSAM RLS Level 2.

William Li is a software engineer at the IBM Silicon Valley Laboratory. He is currently a member of the IMS XML database team, and has previously worked in IMS Fast Path Test and IMS DB Adapter Development.

Adam Lickel is a software engineer at IBM Poughkeepsie. Since joining IBM in 2006, he has worked on z/OS Language Environment and z/OS UNIX.

Joel Masser works in DFSMS Test. His 11 years with IBM includes assignments in Test, Performance, Service, and Development. Currently, Joel develops tools for testing and problem determination.

Ed Mekeel works in System z Performance at IBM Poughkeepsie. His 38 years with IBM includes seven years in Automated Operations Software Development.

Terri M. Menendez is an IBM Senior Technical Staff Member in DFSMS Development. Her 24 years with IBM includes helping to develop VSAM RLS. In z/OS V1R7, Terri led the effort to develop RLS 64-bit buffering.

Ken Morgan is a software developer at IBM Poughkeepsie. His 26 years with IBM includes assignments in both z/OS and z/VM. For the past five years, Ken has worked on LDAP, specializing in RACF.

Barbara Neumann, an Advisory Software Engineer, is the lead information developer for z/OS Language Environment. She holds an MBA from the University of Rochester.

Wayne O’Brien is an Advisory Software Engineer at IBM Poughkeepsie. Since joining IBM in 1988, he has developed user information for a variety of products. Wayne is a managing editor for z/OS Hot Topics, and holds an MS degree from RPI of Troy, N.Y.

Jun Ogata is a Staff Software Engineer who has been function testing RACF for more than 15 years. He has also tested other z/OS components, most recently XML System Services. Jun strives to demystify the magic of RACF.

Purvi Patel is an Advisory Software Engineer in the I/O Supervisor component at IBM Poughkeepsie. She joined IBM in 1999 and holds an MS degree in Computer Science from New York Institute of Technology in Westbury, NY.

Bob Petti is the Function Verification Test team lead for ICSF Development. His experience includes testing a variety of z/OS components, such as zFS, LDAP, SSL, and DCE.

Kathy Pfeiffer is an Advisory Software Engineer at IBM Poughkeepsie. She is the curriculum manager for the IBM Academic Initiative System z9 Program.

Roity Prieto is a software engineer in the VSAM RLS Development team at IBM Silicon Valley Laboratory.

David Raften is an Advisory Software Engineer in IBM Poughkeepsie, with more than 25 years of experience. Since 1994, he has provided technical support to Parallel Sysplex customers.

Ahilan Rajadeva works in z/OS UNIX Kernel and File System Development at IBM Poughkeepsie, NY.

John P. Rankin is a Senior Software Engineer in System z Performance at IBM Poughkeepsie.

Victor Sachar is an Advisory Software Engineer at IBM Poughkeepsie. He has been with IBM for 11 years, and leads the Security Information team. Victor owns the libraries for RACF, PKI, EIM, LDAP, and NAS, to name but a few. He enjoys the challenge.

Andrew D. Schmidt is a program manager in IBM System z ISV Enablement, where he helps software vendors enable applications for System z and Linux for System z. Previously, Andrew was a developer and release leader in MVS/JES3 Design and Development.
Juergen Schneider has been with IBM Boeblingen for 15 years. He has worked on various hardware functions, integrated I/O, and microcode I/O instructions. For the past three years, Juergen has worked in z/OS WLM Test and Development.

John C. Shebey III is an Advisory Software Engineer at IBM Poughkeepsie, where he leads the z/OS UNIX Level 2 Support team. John has been with IBM for nine years and is a frequent speaker at SHARE.

Mike Spiegel works in z/OS UNIX Kernel and File System Development at IBM Poughkeepsie. He was one of the original designers of z/OS UNIX. Previously, Mike worked in TSO/E Development.

Elisabeth Stahl is the program manager for IBM Performance Marketing and an Executive IT specialist in the Advanced Client Technology Center at IBM. She specializes in industry standard benchmarking and performance. Read her blog at ibm.com/developerworks/blogs/page/benchmarking.

John A. Staubi is a Senior Software Engineer at IBM Poughkeepsie. He joined IBM in 1991 after receiving an MS degree in Computer Science from Northeastern University in Boston, MA. John is currently the team leader for I/O Supervisor Level 3.

Ronald Townsend is an IBM software engineer with 20 years of experience in mainframe application design. Currently, he works on transforming host applications to fit service-oriented architecture (SOA). Ronald holds a BS degree in Computer Science from Purdue University.

Susan M. Tuczynski is a Senior Program Manager with the System z Solution Developer Enablement team, where she assists ISVs with hardware access, tools, and education. Susan manages the System z for ISV initiative, which helps ISVs port applications to z/OS.

Marna Walle is a Senior Technical Staff Member in the z/OS Build and Installation group at IBM Poughkeepsie. Her current responsibilities include z/OS release migration and new installation technologies. Marna is a frequent speaker at SHARE.

C.T. Ware is a Staff Software Engineer on the z/OS Level 2 Support team for z/OS Language Environment, C Run-Time, and OpenSSH. He is a graduate of Clarkson University.

Mark Wisniewski has worked in System Performance for over 25 years. He is responsible for assessing System z growth constraints and improving performance. Mark also focuses on improving high MIPS scalability for System z.

Mike Young, a member of the z/OS Java Level 2 Support team, is notorious for writing overly long and emotional e-mails. He believes himself to be the perfect candidate for addressing the exciting subject of garbage collection.

Harry M. Yudenfriend is an IBM Distinguished Engineer at IBM Poughkeepsie, where he works in I/O Supervisor Design. He joined IBM in 1980 after receiving an MS degree in Computer Science from Columbia University in New York, NY.
Friends, Romans, Countrymen!
We’ll lend you our JCL

The Reusable JCL Collection

BY AMY S. LANDER

JCL servi sumus ut liberi esse possimus
Translation: We are slaves of JCL so that we may be able to be free
(Cicero, sort of)

Do you think that learning Latin might be easier than learning how to code z/OS job control language (JCL)? Then we have a study aid for you: the Reusable JCL Collection in the z/OS Basic Skills Information Center. This collection is a starter set of reusable samples with instructions that tell you how to modify each sample for your own use.

Parturient montes, nascetur ridiculus mus
Translation: Mountains will be in labor, and an absurd mouse will be born (Horace)
(In other words: All that work and nothing to show for it)

JCL is a set of statements that z/OS professionals use to define work to be done by the operating system. Although JCL has a limited number of statements that you can use, it has hundreds of parameters, keywords, and subparameter lists that need to be coded precisely to avoid data entry or syntax errors.

Back in the 1960s, programmers submitted JCL to the operating system on punched cards with an 80-character limit for each line. User interfaces for coding and submitting JCL have improved over the years, but JCL still has strict syntax rules — carried over from the days of punched cards — that are difficult to teach, learn, and remember.

You could use current IBM product documentation to learn how to code JCL from scratch, but it would take considerable time to do so. Although the documentation is quite thorough, it assumes a level of z/OS experience that today’s new hires and retrainees rarely have. Besides, the sheer volume of JCL information is in itself intimidating. And misplacing a parameter — or forgetting a comma — can make the difference between a job that runs and one that does not.

Facilis per partes in cognitionem totius adducimus
Translation: We are more easily led part by part to an understanding of the whole
(Seneca the Younger)
(In other words: It is easier to understand the whole piece by piece)

Most people use “canned JCL” that a mentor or co-worker has already written, but much of it lacks sufficient supporting documentation. In today’s challenging work environment, your mentor might not have the time to explain what each JCL statement and parameter does, nor the time to tell you exactly what to change to successfully use the JCL for your own purposes.

That’s where the Reusable JCL Collection can play a valuable role in your education: The samples and supporting information help you and your mentor work most efficiently. The collection is canned JCL with a twist: Each sample consists not only of JCL code but also:

• Line-by-line descriptions that explain the function of each statement
• Instructions for modifying each JCL statement for use in your work environment
• A checklist for collecting company-specific information that you need to correctly modify the JCL
• An overview of the tools that you need to create and submit JCL, and to monitor submitted jobs and view their output
• Pointers to related topics in the product documentation, if you want to learn more.

A posse ad esse
Translation: From possibility to actuality
(In other words: We made it happen!)

Working with current z/OS customers, we identified common training tasks for new people and developed the JCL needed to accomplish those tasks, which include:

• Creating a data set
• Copying a data set to tape
• Copying a partitioned data set
• Copying a load module
• Deleting a data set.

So the next time your mentor tells you to write some JCL, look to the Reusable JCL Collection for the instructions and resources you need. The collection is available through the z/OS Basic Skills Information Center:

http://publib.boulder.ibm.com/infocenter/zosinfocenter/v1r7/index.jsp

Factum est!
Translation: It is done!

SYSTEMS PROGRAMMERS

GA22-7501-14