On to V1R10
And a greener data center

In this issue:

- Migration actions
- The dawn of EAV
- Unlocking the power of System z
  - 64-bit virtual
  - DVE
  - Virtual storage constraint relief
- IBM Migration Checker for z/OS
- Improved processing for PSP buckets
- Hot Topics “How Tos”
- AutoIPL, security stuff, and more
We’re moving on — to z/OS V1R10, that is, with great articles on migration and exciting new functions that are sure to get you going. As Marna Walle, System z migration guru featured on this issue’s cover says, there’s a wealth of great new stuff in the latest release of z/OS, but you need to get there first. And that’s what our lead article explains in “The shortest distance between two releases,” which provides hints and tips for a smooth migration. Be sure to check into her article about the IBM Health Checker for z/OS and migration. So despite the high cost of gas these days with this issue of z/OS Hot Topics you can feel assured of a hassle-free journey to the best new features of System z!

In “Fortune favors the prepared system,” Greg Daynes helps you prepare for the transition to System z10 by providing a summary of the System z hardware functions you can exploit on z/OS releases starting with z/OS V1R7.

Speaking of z/OS offerings for V1R10, we give star billing to the extended address volume (EAV) with two articles, one that describes this stellar new way to manage your DASD farm by accommodating hundreds of terabytes on a single volume and a second article that describes the ways to exploit EAV.

And there’s more good storage information on exploiting 64-bit virtual, virtual storage constraint relief, consolidating extents, and dynamic volume expansion.

We know that energy costs are going up, but as Susan Greenlee explains in “A greener data center courtesy of Active Energy Manager,” you can learn how AEM, an IBM Director Extension product, can help you save some “green” by monitoring your data center servers and devices for temperature and power consumption.

This issue continues to help take the mystery out of z/OS security with a second installment that demystifies the “magic” of the RACGLIST class. RACF commands allow you to tailor a new set of clothes for customized fields in user and group profiles (“Custom fields for a custom fit”), while you also learn how to grow your own digital certificates through PKI Services instead of buying them (“Grow your own: using locally generated digital certificates”). “Tales” from TCP/IP hardware cryptography take the fear out of how System z hardware can secure IP communications and improve cryptographic performance, “And the Password is…” looks at new V1R10 support of password phrases for z/OS UNIX Systems Services.

There are so many more stops along the way and so much to take in from GDPS to systems management to System z Server Time Protocol, HiperDispatch, JES2 dynamic exits, DFSMS, IMS, device allocation, sysplex, capacity provisioning, Language Environment, AutoIPL, PSP buckets, I/O — you name it, we’ve got it on our itinerary. And don’t forget the latest news on the Academic Initiative with “Contests, contests everywhere.”

We’re also happy to welcome a new fellow traveler to the editorial staff, Victor Sachar of z/OS User Technologies in IBM Poughkeepsie. Welcome, Vic!

With those skyrocketing gas prices, you might not be able to hit the road this summer, but this issue of Hot Topics will take you where you want to go.

The Editors

newsletter@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**

<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>A look at some new migration actions for z/OS V1R10</td>
<td>MARNA WALLE</td>
</tr>
<tr>
<td>8</td>
<td>A healthy migration with IBM Health Checker for z/OS!</td>
<td>MARNA WALLE</td>
</tr>
<tr>
<td>11</td>
<td>Positioning your z/OS system for IBM System z10</td>
<td>GREG DAYNES</td>
</tr>
<tr>
<td>17</td>
<td>An overview of the extended address volume</td>
<td>JAN REDDING, JAMES CAMMARATA, AND CECILIA CARRANZA LEWIS</td>
</tr>
<tr>
<td>20</td>
<td>Deploying extended address volumes</td>
<td>CECILIA CARRANZA LEWIS, JAMES CAMMARATA, AND WAYNE RHOSTEN</td>
</tr>
<tr>
<td>6</td>
<td>Use the new and improved space management commands in DFSMSdss</td>
<td>JUSTIN EASTMAN AND JASON LUURS</td>
</tr>
<tr>
<td>14</td>
<td>A greener data center courtesy of Active Energy Manager</td>
<td>SUSAN GREENLEE</td>
</tr>
<tr>
<td>23</td>
<td>Sex and the 64-bit virtual environment</td>
<td>DON AULT</td>
</tr>
<tr>
<td>25</td>
<td>Virtual storage constraint relief</td>
<td>ELPIDA TZORTZATOS, CHARLES MARI, AND PAULA SPENS</td>
</tr>
<tr>
<td>28</td>
<td>Dive into Dynamic Volume Expansion</td>
<td>JOHN PAVEZA, MIKE KOESTER, AND CHUNXIAO (CONNIE) HE</td>
</tr>
<tr>
<td>30</td>
<td>Demystifying the magic of RACCLIST</td>
<td>JUN OGATA AND RUSS HARDGROVE</td>
</tr>
<tr>
<td>32</td>
<td>HOW TO personalize your RACF checking with IBM Health Checker for z/OS</td>
<td>MARK NELSON AND GEORGE MARKOUIZOS</td>
</tr>
<tr>
<td>34</td>
<td>Custom fields for a custom fit</td>
<td>LAURIE WARD AND ERIC ROSENFELD</td>
</tr>
<tr>
<td>36</td>
<td>DISPOUT will help you out</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Using locally generated digital certificates</td>
<td>WAI CHOI</td>
</tr>
<tr>
<td>39</td>
<td>Decrypting TCP/IP hardware cryptography</td>
<td>CHRIS MEYER</td>
</tr>
<tr>
<td>43</td>
<td>Added support for password phrase and PassTickets</td>
<td>LISA NOVAL AND SCOTT BENDER</td>
</tr>
<tr>
<td>44</td>
<td>Using AutoIPL to minimize z/OS downtime</td>
<td>WALT OTTO</td>
</tr>
<tr>
<td>46</td>
<td>Contests, contests everywhere</td>
<td>KATHY PFEIFFER AND MIKE TODD</td>
</tr>
<tr>
<td>48</td>
<td>z/OS XML processing relief to the rescue</td>
<td>BILL CAREY</td>
</tr>
<tr>
<td>49</td>
<td>zFavorites</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Clicking away failures with IMS V10 abend search and notification</td>
<td>DARIO D'ANGELO, KIN LAU, AND ALAN SMITH</td>
</tr>
<tr>
<td>52</td>
<td>Serviceability scales up!</td>
<td>BOB ABRAMS</td>
</tr>
<tr>
<td>55</td>
<td>Cut and fold IBM System z10</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Using SMP/E V3R5 to identify and install fixes</td>
<td>KURT QUACKENBUSH</td>
</tr>
<tr>
<td>58</td>
<td>Single point of failure detection</td>
<td>MATTHEW HANK SABINS</td>
</tr>
<tr>
<td>61</td>
<td>z/OS Capacity Provisioning can help</td>
<td>MICHAEL GROETZNER, HORST SINRAM, AND RALF THELEN</td>
</tr>
<tr>
<td>64</td>
<td>Full speed ahead with HiperDispatch</td>
<td>DANIEL ROSA AND DONALD SCHMIDT</td>
</tr>
<tr>
<td>69</td>
<td>Extending GDPS across the enterprise</td>
<td>NOSHIR DHONDY</td>
</tr>
<tr>
<td>72</td>
<td>Server Time Protocol adds Network Time Protocol client support</td>
<td>PAUL HAGEN AND GEORGE NG</td>
</tr>
<tr>
<td>75</td>
<td>Enabling multi-user access for the IODF</td>
<td>FRIEDRICH BEICHTER</td>
</tr>
<tr>
<td>77</td>
<td>Tape devices made unavailable to the system</td>
<td>STEVEN B JONES AND GIRIJA VARANASI</td>
</tr>
<tr>
<td>79</td>
<td>RAS enhancements for global resource serialization</td>
<td>BEN SEGAL, CHRIS BROKER, AND STEVEN PARTLOW</td>
</tr>
<tr>
<td>82</td>
<td>Dynamic log stream duplexing</td>
<td>DOUG ZOBRE, ROBERT MILLER JR., AND NICHOLAS JONES</td>
</tr>
<tr>
<td>84</td>
<td>HOW TO upgrade your coupling facility</td>
<td>PATTY SALONE AND NEIL JOHNSON</td>
</tr>
<tr>
<td>87</td>
<td>Replace the sysplex root file system while it is in use</td>
<td>AHILAN RAJADEVA AND WILLIAM HUBER</td>
</tr>
<tr>
<td>89</td>
<td>A look at JES2 dynamic exits support in z/OS V1R10</td>
<td>TOM WASIK</td>
</tr>
<tr>
<td>91</td>
<td>Using the new pool-count delimiter</td>
<td>JUSTIN MCCOY</td>
</tr>
<tr>
<td>93</td>
<td>Easy verification of CEEPROMxx parmlib member</td>
<td>JUSTIN MCCOY</td>
</tr>
<tr>
<td>95</td>
<td>z/OS Language Environment’s CEEROPT</td>
<td>CHARULATHA DHUVUR, JOHN MONTI, AND BARBARA NEUMANN</td>
</tr>
<tr>
<td>96</td>
<td>Our contributors</td>
<td></td>
</tr>
</tbody>
</table>

**Back cover:**

Adding LookAt to the Mozilla Firefox search bar

DEBBIE QUICK
The shortest distance between two releases

A look at some new migration actions for z/OS V1R10

BY MARNA WALLE

To use the wealth of new functions in z/OS® V1R10, you’ll need to migrate to z/OS V1R10 first. In this article, we’ll describe some new migration actions for z/OS V1R10 and explain how to plan for them now to get you to your destination. Of course, be sure to refer to z/OS V1R10 Migration, GA22-7499-13, for a complete list of all appropriate migration actions that apply to you.

**Action: Reallocate your WLM couple data set**

Because of the growth in workload manager (WLM) policy structures, WLM will probably determine that your WLM couple data set is too small during policy activation on z/OS V1R10. If the WLM data set is too small, policy activation will fail on z/OS V1R10.

**What you can do now**

After installing your z/OS V1R10 system, re-allocate your WLM couple data set as one of your first actions. To ensure the correct calculation, use your z/OS V1R10 system to perform the couple data set allocation (a JCL STEPLIB to that system’s SYS1.MIGLIB is fine).

**Action: Obtain updates to programs that use unsupported subsystem console functions and for those without appropriate serialization for CNZMXURF**

As of z/OS V1R10 and APAR OA18204, the following subsystem console request functions are no longer supported:

- Request for demand select (as of OA18204)
- Request to change routing codes (as of z/OS V1R10)
- Request to release, set master authority, or reset master authority on non-owning systems (as of OA18204).

Also, in z/OS V1R10 users must hold appropriate serialization for CNZMXURF (the macro that locates the console control block, UCME) and the UCME that CNZMXURF returns.

**What you can do now**

Activate the tracking facility and look for violations that start with IEAVG700 and UCME-CMS. Obtain any necessary updates to the programs that are identified as affected.

**Action: Keep your z/OS version root file system in a single data set**

No surprises here — the z/OS version root file system has been growing. For z/OS V1R10, it is almost the size of a 3390-3 DASD volume. (The z/OS V1R10 version root file system is approximately 3100 cylinders, which makes it a tight fit within the 3339 cylinders of a 3390-3.) If you currently restore your z/OS version root to a single 3390-3 volume, decide what to do for z/OS V1R10. Plan to keep the file system in a single data set because it is easier to manage as a single file system.

**What you can do now**

Review the location where you will restore your z/OS V1R10 root file system to see if the volume is large enough. Also, make sure to review all the volumes on the systems where you deploy the root file system. If the volume isn’t large enough, decide if you need a larger volume (perhaps 3390-9) or use an SMS data class to make it a multi-volume data set. While you’re at it, review other file systems to determine if they need to move to larger volumes (or to be distributed across multi-volume data sets).

**Action: Review the various migration actions associated with EAV enhancement**

z/OS V1R10 introduces support for extended address volume (EAV), a volume with more than 65,520 cylinders. With this initial release, EAV supports 262,668 cylinders per volume for SMS and non-SMS managed VSAM data sets (except for catalogs, page data sets, and data sets defined with KEYRANGE or IMBED attributes). You might need to perform certain migration actions depending on the following situations:

- ICKDSF R17 APAR PK56092 provides support for the EAV function and affects both EAV and non-EAV users. With this APAR, a change to the default size of a VTOC index occurs if you take the recommended default for the INDEX parameter of the ICKDSF INIT command. If you use the new default, the system builds the index size with a value that is based on the size of the VTOC. Because the build is automatic, you don’t have to calculate the size for the index.
- Responses to the DEVSERV QDASD and DEVSERV PATHS commands have different column widths and column names.
- A new version of the DEFRAG command is compatible with any EAV; however, this new version of the command drops the
CONSOLIDATE keyword. To replace the CONSOLIDATE keyword, a new DFSMSdss™ CONSOLIDATE command is used in conjunction with the new version of the DEFRAG command. The old version of DEFRAG with the CONSOLIDATE keyword is still available, but you have to specify the keyword VERSION1 to use that level. (Note that the VERSION1 level of DEFRAG is not compatible with EAV.) No error occurs if you do not specify VERSION1 on existing JCL that specifies the CONSOLIDATE keyword on the DEFRAG command, but it will result in running the new DFSMSdss CONSOLIDATE command, then the new DEFRAG command. See the article “Consolidate that clutter” on page 6 for more information.

The track addresses in the output of the IDCAMS LISTDATA PINNED command are in a different format, regardless of the volume size.

What you can do now
Review all these changes and determine if they have an impact.

Action: Accommodate changes in the control area (CA) sizes assigned to newly allocated VSAM data sets.
The CA size selection is done differently on z/OS V1R10 for all new VSAM data sets (not just those for EAV). The CA size is 1, 3, 5, 7, 9 or 15 tracks and is determined by the primary and secondary space allocation amounts. As before, you cannot explicitly specify a CA size. Before z/OS V1R10, other possible CA sizes existed.

What you can do now
Determine if you have dependencies on a specific CA size or a dependency on a CA size that is no longer provided. Prepare to adjust any primary and alternate allocation requests to accommodate the CA size.

Action: Ensure compatible levels of VTAM for high performance routing-capable (HPR-capable) sessions
If you run z/OS V1R10 Communications Server as an HPR-capable interchange node in a mixed subarea and Advanced Peer-to-Peer Networking® (APPN), read on. On z/OS V1R8 and later, additional information on APPN session establishment flows identifies when sessions cross from APPN into subarea (or vice versa) through an interchange node. In z/OS V1R10, this additional information separates interchange node sessions from APPN-only sessions by placing the two sessions on different Real-Time Transport Protocol (RTP) pipes. If any instance of VTAM® in the network or in attached APPN networks is not running at z/OS V1R8 or later, the z/OS V1R10 interchange nodes might incorrectly place interchange node sessions on the wrong RTP pipe. The result can be session setup failures.

What you can do now
Ensure that when you introduce z/OS V1R10 in an HPR-capable interchange node environment that all instances of VTAM and attached APPN networks are running z/OS V1R8 or later.

Action: Accommodate FTP server support for the SECUREIMPLICITZOS statement
In z/OS Communications Server, the SECUREIMPLICITZOS statement in FTP.DATA indicates when the z/OS FTP server must configure the SECUREIMPLICITZOS statement or accept the default of TRUE. Because the values of SECUREIMPLICITZOS must be TRUE for both the z/OS FTP client and server for a successful connection to the protected port of the server, consider the effect on your z/OS FTP clients if you set the z/OS V1R10 FTP server value to FALSE. If connections are affected, specify SECUREIMPLICITZOS TRUE for the z/OS FTP server or accept the default of TRUE.

Action: Fix syntax errors identified during SMB startup
In releases before z/OS V1R10, the system does not validate syntax errors in SMB environment variables. Starting in z/OS V1R10, if the system finds syntax errors in the SMB environment variables (for instance, in /opt/dfslocal/home/dfskern/envar), SMB might not start with the intended parameters. A new command, dfssyntax, in z/OS V1R10 can help you identify syntax errors in environment variable files.
What you can do now
Ensure that the syntax on your environment variable file has the correct syntax. When on z/OS V1R10, use dfssyntax to verify any changes to the environment variable file.

Action: Take advantage of z/OS V1R10 ICEPRMxx enhancement in DFSORT.
In releases before z/OS V1R10, when you specify changes to the installation options for DFSORT, you need to customize the ICEMAC macro and place those changes in an SMP/E USERMOD. As of z/OS V1R10, you can use a started task to activate DFSORT installation options specified in one or more ICEPRMxx members in PARMLIB during or after IPL.

What you can do now
After you have initialized z/OS V1R10, instead of using usermods, specify and activate DFSORT installation options in one or more ICEPRMxx members of parmlib. Using ICEPRMxx members is only supported on z/OS V1R10 and higher; you’ll still need your ICEMAC usermods on lower level systems. DFSORT installation options in ICEPRMxx override the default options from ICEMAC. By using ICEPRMxx on z/OS V1R10 and higher, you can eliminate your ICEMAC usermod (and the task associated with installing it) required at each z/OS migration!

An easy ride
With these migration tips for z/OS V1R10, you should have an easy ride to wherever your destination lies. Here’s to a smooth trip.

Consolidate that clutter
Use the new and improved space management commands in DFSMSdss

BY JUSTIN EASTMAN AND JASON LUURS

Are your volumes highly fragmented?
Are your data sets running out of extents?
You are not alone. With the frequent creation, extension, and deletion of data sets, free space on volumes becomes fragmented, which can limit the availability of your DASD space. In addition, as data sets are forced into multiple extents, these files can perform poorly, thus inhibiting the growth of your business.

Fear not, help is on the way. To help you manage your space more efficiently, we did some housecleaning of our own in DFSMSdss. We streamlined our defragmentation processing, and cleaned up a few aspects of the user interface as well.

In z/OS V1R10, we enhanced DFSMSdss in the following ways:
• We added a new command, CONSOLIDATE, to help you consolidate multiple extent data sets on a particular volume.
• We enhanced an existing command, DEFRAG, to improve its performance with highly fragmented volumes. We also added new keywords to this command to help you defragment volumes more efficiently.

Let’s look briefly at how the new and improved DFSMSdss commands (that is, new CONSOLIDATE, improved DEFRAG) can help you keep “a clean house.”

New CONSOLIDATE command
In previous releases of z/OS, you consolidated the data set extents on a volume by entering the DFSMSdss DEFRAG command with the CONSOLIDATE keyword. This approach worked, but it had some problems, which we will address in a moment.

In z/OS V1R10, we added a new command, CONSOLIDATE, which is designed specifically for consolidating data set extents. We think you are going to love the new CONSOLIDATE command for the following reasons:
• CONSOLIDATE works with data set extents—and that’s it. Previously, you needed to run a whole DEFRAG operation to do this step.
• CONSOLIDATE lets you specify which data sets are to be consolidated. To do so, specify DATASET with one or more of the following sub-keywords: INCLUDE, EXCLUDE, or BY. Previously, you needed to run DEFRAG against the entire volume.
• If a data set cannot be fully consolidated to a single extent, CONSOLIDATE will attempt to consolidate as many extents as possible, depending on the amount of available space on the volume. Previously, with DEFRAG, if no free space extent was large enough to hold the entire data set, the data set was not processed.
With the introduction of the CONSOLIDATE command, the CONSOLIDATE keyword on DEFRAG becomes obsolete. If you continue to use it in z/OS V1R10, DFSMSdss honors your request, but issues an informational message and runs the CONSOLIDATE command before running DEFRAG.

To have consolidation proceed as it did in previous releases, specify the VERSION1 keyword with the CONSOLIDATE keyword.

In z/OS V1R10, we streamlined DEFRAG internal processing to improve its performance with highly fragmented volumes.

In addition, to give you more control over DEFRAG processing, we added these keywords to the DEFRAG command:

- MMOVPCT specifies the percentage of volume tracks that DFSMSdss is to try to assemble as free tracks in a contiguous area.
- MAXTIME limits the amount of time in which a DEFRAG operation can run.

In z/OS V1R10, we streamlined DEFRAG internal processing to improve its performance with highly fragmented volumes.

DEFRAG Version 2.0
Volumes continue to grow in size, as does the time it takes to defragment them. These days, you might notice that the DFSMSdss DEFRAG command is taking longer to complete. It’s because the original DEFRAG command wasn’t designed to accommodate today’s larger volumes.

More time for you
With the new CONSOLIDATE command, you can now consolidate extents on a volume without the overhead of running DEFRAG. And, with the enhanced DEFRAG command, you can defragment larger volumes more quickly.

As you prepare to move to z/OS V1R10, you might want to reevaluate your maintenance workload to see if these enhancements will save time. If you do end up with a little more time on your hands, don’t worry. We won’t tell the boss.

Find out more
For more information about the CONSOLIDATE and DEFRAG commands, see the latest edition of z/OS DFSMSdss Storage Administration Guide, SC35-0423.
Doctor’s orders for a healthy migration

IBM Health Checker for z/OS!

BY MARNA WALLE

If you’re one of more than a thousand people who have downloaded the IBM® migration checker for z/OS tool, you’ll be interested in this! With z/OS V1R10, migration assistance has taken a leap forward through the IBM Health Checker for z/OS infrastructure. Recognizing that our “as is” tool was growing popular and that supported programmatic assistance was best done “close to the component,” IBM Health Checker for z/OS was a natural fit.

Distinguishing checks for migration purposes

First, we had to make sure that the new use of IBM Health Checker for z/OS wouldn’t have any adverse affects on your existing check procedures. With that in mind, we have established a convention that helps you distinguish those checks added for migration purposes from those added for availability and best practices. What can be slightly confusing is that sometimes a best practice means doing a certain migration action, so don’t assume that all checks clearly fall into one or the other category.

Starting with z/OS V1R10, migration checks follow a specific naming convention and ship inactive. That is, you’ll have to activate the checks when you are ready to gather migration-planning information. We did this so that you wouldn’t be bothered with migration information during day-to-day use of your system — you are in the driver’s seat and control when to start planning migration.

Naming conventions

The naming convention for the z/OS migration checks is as follows:

ZOSMIGVvRrr _ component _ program _ name

For Integrated Cryptographic Support Facility (ICSF), the naming convention is as follows:

ICSPMIGNnnn _ component _ program _ name

You can see that we’ve used the characters MIG followed by a release identifier (VvRrr or nnnn). This tells you the check is for migration purposes and in which release that migration action becomes appropriate.

Using this convention, you can activate migration checks to suit your migration path. Of course, looking at all migration actions — even those for releases beyond your migration path — always gives you a glimpse at the tasks you need to perform during migration, whether now or later.

Migration checks do not perform any migration action for you. Instead, the checks alert you that a specific migration action is applicable, and then verifies that you’ve performed the migration action successfully.

Steps to take for migration checks

Let’s look at how you’d typically use migration checks. Here’s an overview of the steps to take with more details on each step that follows:

1. Install the latest available migration checks.
2. Activate the migration checks.
3. Review the migration check output and rerun as needed.
4. Deactivate the migration checks.

When you’ve migrated to the new z/OS release, follow these steps as for your previous z/OS release, but understand that this time you are verifying that the migration actions have been performed correctly.
Install the latest available migration checks

A key to a successful migration lies in knowing what you have to do beforehand, and then verifying that it was done properly. With that in mind, we’ve made the migration checks installable on your current z/OS release in the service stream. As with all checks for IBM Health Checker for z/OS introduced in the service stream, migration checks are identified in the functional PSP bucket HCHECKER. Using the new SMP/E V3R5 REPORT MISSINGFIX with the FIXCAT of IBM.Function.HealthChecker makes checking this functional PSP bucket very easy. Alternately, use this Web site to view functional PSP buckets: www14.software.ibm.com/webapp/set2/psp/srchBrokr.

All checks that are introduced in a release or in service are identified on ibm.com/systems/z/os/zos/hchecker/check_table.html. The best thing to do is to monitor the HCHECKER functional PSP bucket periodically to see if any migration checks have been added or updated.

Installing the migration check PTFs during a preventive service window might be best for you to take advantage of an IPL that follows. Checks are often added during initialization of a function, so it might not be convenient to stop and restart that function.

Activate the migration checks

Because migration checks aren’t active by default, you need to take an overt action to run them. To view which migration checks are available on your system, use your favorite method of looking at checks for IBM Health Checker for z/OS, such as through SDSF, zMC, or other product.

You can use wildcards with the naming convention described above to help with this activation. Although there are different ways you can activate a check, we’ll show examples using the MVS™ MODIFY command. (You might find that activating through SDSF is more intuitive as it issues the MODIFY command under the covers.)

By activating the checks both on your old z/OS release and new z/OS release, you can help with preparation and verification activities.

With z/OS V1R10, migration assistance has taken a leap forward through the IBM Health Checker for z/OS infrastructure.

On your old z/OS release

For example, if you are migrating from z/OS V1R8 to z/OS V1R10, you want to run the checks for z/OS V1R10 as well as the release you are skipping, z/OS V1R9. Here are some of the MODIFY commands that use wildcards to do just that:

```
MODIFY HZSPROC, ACTIVATE, CHECK=(IBM*,*MIG*)
MODIFY HZSPROC, ACTIVATE, CHECK=(IBM*,ZOSMIG*)
MODIFY HZSPROC, ACTIVATE, CHECK=(IBM*.ZOSMIGV1R9)
MODIFY HZSPROC, ACTIVATE, CHECK=(IBM*.ZOSMIGV1R10)
along with
MODIFY HZSPROC, ACTIVATE, CHECK=(IBM*.ICSFMIG*)
```

Remember, the ICSF migration checks use a slightly different naming convention from the rest of the z/OS product, so make sure your wildcard choice includes ICSF.

On your new z/OS release

You still need to activate the migration checks on the releases that you came through as well as those that you came to, although some of them might not be available on this “higher” release. (That’s because, in some cases, you must perform a migration action before the migration to the release and cannot do it after migration to that release. It makes no sense to run on the higher release, so that migration check is not available on the release.) Use the same activation examples above on your new z/OS release.

Review the output of the migration checks and rerun as needed

You’ve run the checks; now see what they’ve found!

- If you’re reviewing output from the migration checks on your old z/OS release, exceptions indicate that you have a migration action to perform.
- If you’re reviewing output from the migration checks on your new z/OS release, exceptions indicate these migration actions are still outstanding (and either weren’t performed or weren’t performed correctly).

You can always run migration checks again to verify a specific migration action.
Deactivate the migration checks

Some migration checks run just one time, while others run more than once. You might want to keep the migration checks active for a day or two in order to have a longer "migration picture" to report. When you are ready to deactivate the migration checks, you can do it in a similar fashion as when you activated them. Here are the same examples that use the MODIFY command from above, but this time the checks are being deactivated:

```plaintext
MODIFY HZSPROC,DEACTIVATE,CHECK=(IBM*,*MIG*)
MODIFY HZSPROC,DEACTIVATE,CHECK=(IBM*,ZOSMIG*)
MODIFY HZSPROC,DEACTIVATE,CHECK=(IBM*.ZOSMIGV1R9)
and
MODIFY HZSPROC,DEACTIVATE,CHECK=(IBM*.ZOSMIGV1R10)
along with
MODIFY HZSPROC,DEACTIVATE,CHECK=(IBM*.ICSMIG*)
```

What migration checks will be available for z/OS V1R10?

We are introducing migration checks starting in z/OS V1R10. We won’t have migration checks for all the migration actions listed in the z/OS V1R10 z/OS Migration, GA22-7499-13; that’s just not programmatically possible.

Migration checks for z/OS V1R10

Here are the checks we are planning as migration checks for z/OS V1R10. Keep in mind that other checks for best practices continue to help with migration actions. z/OS Migration cross-references checks for IBM Health Checker for z/OS to migration actions (whether they are migration checks or best practices checks). Remember to keep an eye out for new ones added in the HCHECKER functional PSP bucket (ideally, by using the FIXCAT IBM.Function.HealthChecker).

- ICSFMIG7731.ICSF.RETAINED.RSAKEY — to detect the existence of retained RSA private keys on a PCICC or PCIXCC/CEX2C cryptographic card
- ICSFMIG7731.ICSF_PKDS_TO_4096BIT — to alert that the public key data set (PKDS) size in a pre-HCR7750 environment is sufficiently allocated to support 4096-bit RSA keys
- ZOSMIGV1R9.RACF_PASSWRD_ENVELOPE — to identify installations that are not running password enveloping and have defined a generic profile that can cause RACF® to start enveloping passwords when running on z/OS V1R9 or later
- ZOSMIGV1R0_CS_BINL — to support removal of boot information negotiation layer (BINL)
- ZOSMIGV1R0_CS_BIND4 — to support removal of Berkeley Internet Name Domain (BIND) DNS 4.9.3
- ZOSMIGV1R0_CS_DHCP — to support removal of dynamic host configuration protocol (DHCP)
- ZOSMIGV1R0_CS_NDB — to support removal of network database (NDB)

What will become of the IBM migration checker for z/OS tool from the Web?

The IBM migration checker for z/OS “as is” tool will remain on the Web download page at ibm.com/systems/z/os/zos/downloads. It contains many migration programs helpful with migrations to z/OS V1R8 and z/OS V1R9 (and perhaps beyond). However, we are not planning on making updates to that tool.

Check into it!

We hope this leap forward will bring you more timely and extensive migration assistance and strengthen your use of the IBM Health Checker for z/OS! Keep on checking! ■
Fortune favors the prepared system

Positioning your z/OS system for IBM System z10

BY GREG DAYNES

As the saying goes, “Fortune favors the prepared mind.”

In z/OS terms, this might well translate to “It pays to keep your system software up to date.”

If your system is z/OS V1R7 or later, you are already well positioned to run on the next-generation IBM mainframe, the IBM System z10 Enterprise Class (System z10 EC™). That’s because those releases of z/OS can run on a System z10 EC server. The level of functionality you can exploit will depend which release you are running. The System z10 platform builds on the software support provided for IBM System z9 and earlier mainframes. As a result, migrating to the System z10 platform from System z9 (or even from System z890 or z990) will mean less software for you to install than if you were to migrate from an older generation of server.

Further, you can migrate directly to the System z10™ platform without upgrading to an intermediate family of servers. You do not, for example, need to install System z9® before moving to the System z10. If you bypass a generation of servers, you must ensure that your system meets all of the software requirements and migration considerations for any generations that you skip.

Read on for a description of which System z10 functions are release dependent, plus the steps you need to take to position your system software for this move.

System z10 functions by release

The System z10 platform will run any supported release of z/OS. However, the System z10 functional capabilities depend on which release of z/OS is in use at your installation. Generally, the more current your system, the more new hardware functions you can use.

In the following sections, we list the new hardware capabilities by z/OS release, starting with z/OS V1R7, the earliest supported release for the System z10 platform. Figure 1 provides a summary.

---

**Figure 1. System z10 EC exploitation: z/OS support summary**

Legend

B – FMID in base product (assumes service identified in z9 EC PSP Bucket is installed)
W – FMIDs shipped in a Web Deliverable
P – PTFs in the System z10 PSP bucket are required
N – Not supported

** Level of decimal floating-point exploitation depends on z/OS release and PTF level

---
If your system is z/OS V1R7 or later, you are already well positioned to run on the next-generation IBM mainframe, the IBM System z10 Enterprise Class.

Running z/OS V1R7 on System z10
This configuration offers these new capabilities:
- OSA-Express3 10 Gigabit Ethernet (10 GbE)
- InfiniBand coupling
- Increased granularity for multiprocessing (MP) factors (65535)
- New z/Architecture® instruction mnemonics
- HiperDispatch (with the installation of the IBM zIIP Support for z/OS and z/OS.e V1R6/R7 Web deliverable)
- Increased capacity provisioning
- OSA-Express3 double port density.

Running z/OS V1R8 on System z10
This configuration offers the capabilities listed for z/OS V1R7, plus the following:
- Support for up to one terabyte of real memory per logical partition (LPAR)
- Support for hardware decimal floating-point instructions: LE, DBX, and CDA RTLE
- Layer 3 VMAC support
- Service Aids support for large dumps.

To have compiler support for the hardware decimal floating-point instructions and the System z10 machine instructions on z/OS V1R8, you need to install the XL C/C++ compiler for z/OS V1R9. Doing so allows you to:
- Use ARCH(7) or ARCH(8) options to exploit decimal floating point
- Use ARCH(8) or TUNE(8) options to use the new instructions and optimize your code for the System z10 platform.

z/OS V1R8 XL C/C++ does not support decimal floating-point math, but it does support a smaller subset of the ARCH(7) function. For more information about using the XL C/C++ compiler on z/OS, see ibm.com/software/awdtools/czos.

Running z/OS V1R9 or V1R10 on System z10
These latest releases of z/OS offer the capabilities mentioned previously, and integrate the XL C/C++ compiler support for hardware decimal floating-point instructions and the new System z10 machine instructions.

These releases add support for the following:
- 64-way CPs in a single LPAR
- Large pages
- Enhanced capacity provisioning
- HiperSockets™ multiple write facility
- OSA-Express3 double port density.

For z/OS V1R9, using the System z10 cryptographic enhancements requires the installation of a cryptographic Web deliverable. For z/OS V1R10, you have this support already.

Which PTFs and FMIDs to install
As with previous generations of servers, IBM provides z/OS software support for the System Z10 platform through a combination of FMIDs and PTFs. Here also, it pays to stay current. Your release of z/OS will determine which software requirements will need to be met.

Some functions — for example, some cryptographic functions and HiperDispatch on z/OS V1R7 — will require the installation of FMIDs. These are provided without charge as downloadable Web deliverables at the following site: ibm.com/eserver/zseries/zos/downloads.

PTFs and FMIDs for z/OS V1R7
To run z/OS V1R7 on the System z10 platform, install the PTFs listed in the System z10 preventive service planning (PSP) bucket: Upgrade 2097DEVICE, Subset 2097/ZOS.

If you are skipping a server generation, review the PSP buckets for the servers you skipped, as follows:
- For System z9, see Upgrade 2094DEVICE, Subset 2094/ZOS/1.
- For System z990 or z890, see Upgrade 2084DEVICE, Subset 2084/ZOS/1.

To use HiperDispatch, you must download and install the IBM zIIP Support for z/OS and z/OS.e V1R6/R7 Web deliverable (FMID JBB772S), and the PTFs listed in the System z10 PSP bucket. Note that a System z9 Integrated Information Processor (zIIP) is not required on a System z10 server.

If you require CP Assist for Cryptographic Function (CPACF) or Crypto Express2 toleration, you must install one of the following Web deliverables:
- Cryptographic Support for z/OS V1R6/R7 and z/OS.e V1R6/R7
- Enhancements to Cryptographic Support for z/OS and z/OS V1R6/R7 (no longer available)
- Cryptographic Support for z/OS V1R7-V1R9 and z/OS.e V1R7-V1R8 Web deliverable.

If your system is z/OS V1R7 or later, you are already well positioned to run on the next-generation IBM mainframe, the IBM System z10 Enterprise Class.
How would you install a Web deliverable that is no longer available? You can use SMP/E to install a previously downloaded Web deliverable. If you never downloaded the Web deliverable, you must download the latest one.

If you require the support for remote key loading for ATMs and point-of-sale devices and implementation of ISO 16609 CBC Mode TDES MAC, you must install one of the following Web deliverables:

- Enhancements to Cryptographic Support for z/OS V1R6/7 and z/OS.e V1R6/7 Web deliverable (no longer available)
- Cryptographic Support for z/OS V1R7-V1R9 and z/OS.e V1R7-V1R8 Web deliverable.

No matter which cryptographic Web deliverable you install, you should also install the PTFs identified in the Web deliverable program directory and the PTFs recommended in the appropriate ICSF PSP bucket.

PTFs and FMIDs for z/OS V1R10

To run z/OS V1R10 on the System z10 platform, you just need the PTFs described in the System z10 PSP bucket.

With z/OS V1R10, you have most of the software updates already. Here, the important thing is to verify that all of the service from the appropriate PSP buckets is installed.

Reviewing PSP buckets made easier

As with any installation or upgrade, it is important to verify that you have installed all of the service from the appropriate PSP buckets. To this end, SMP/E V3.5, which is included in z/OS V1R10, adds function to simplify the verification and installation tasks associated with PSP bucket service.

One aspect of this new support is that you can use it to install fixes for specific models of the System z9 and System z10 servers (such as the System z10 EC server). For more information, see “Making fast work of your PSP buckets: Using SMP/E V3R5 to identify and install fixes,” on page 57.

Winning is the science of being prepared

Aren’t you glad you stayed current on z/OS? If you did, migrating to the System z10 platform will mean less software to install than if you were to migrate from an older generation of server. In addition, your installation is better prepared to make use of the extensive hardware capabilities of the System z10 platform.

If you are not current, now is the time to evaluate the next migration. For the migration actions for System z10 and earlier generations of servers, see z/OS Migration, GA22-7499-13.

Remember, z/OS V1R7 is planned to be withdrawn from service support in September 30, 2008.

If you stayed current on z/OS, migrating to the System z10 platform will mean less software to install than if you were to migrate from an older generation of server.
Are you relying on published “label” energy ratings for each system to determine the energy needs of your data center? If so, you could be overestimating your energy needs and have no way of knowing if your data center is consuming energy efficiently. Although published information for each system exists, those figures are worst-case numbers that don’t take into consideration that actual energy consumption per server varies with configuration, components, workload, and ambient conditions.

Actual mileage may vary
As with the EPA miles per gallon estimates for vehicles where “actual mileage may vary,” energy usage estimates in your data center can also vary. To understand where and how energy is used, you have to measure it. To make this possible, all IBM server families have been enabled to collect and provide real-time information on energy consumption and ambient and exhaust temperature. Enter IBM Systems Director Active Energy Manager™ (AEM), an IBM Director extension product that gathers, monitors, and manages this data. If you think your data center might be energy constrained, AEM can help you plan future energy needs.

AEM V3.1 is a cross platform product that runs not only on System z, but also on any server that supports IBM Director V5.20.2; it is ideal for deployment on System z™, where AEM is designed to take a data center view. Its main purpose is to measure and monitor the power and temperature of the IT equipment and help manage it through techniques that are part of the product. By installing the base IBM Director and AEM on System z, you can use AEM to monitor energy use for IBM System z™, IBM BladeCenter®, IBM Power Systems, and IBM System x™.

Monitoring your servers and devices
But there’s more! You can also monitor devices and non-IBM servers that can connect to certain Intelligent Power Distribution Units, such as low and mid-range storage devices, even your department’s refrigerator! AEM’s monitoring functions capture information for each system, like AC power coming into the system, DC power coming out of the power supply, input or ambient temperature coming into the system as well as the exhaust temperature. AEM can display this information in real time, and save it over time to show energy usage trends.

System z measures up
System z typically runs at high utilization rates, often 80% to 100%. The new System z10 has been designed to reduce energy usage by more than 80% when consolidating from x86 architecture. For example, the variation in energy use between an idle System z and a System z running at more than 99% utilization during Java™ workload benchmarks can be approximately 150 watts.

These measurements were obtained and validated for System z9 models S08 and S18 configured with a single cargo I/O cage varying the number of processor cores by a factor of four for a z/OS V1R8 Java workload. (See the IBM White Paper “A Power Benchmarking Study on the IBM System z9: Applying Energy Efficiency Metrics to Performance” at ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101110.) Energy consumption on System z is changed by adding or removing hardware like a processor book or an I/O card. System z does not provide support for the power capping management functions of AEM as they add complexity without providing significant energy savings.

In contrast, the average utilization on x86 and other architectures is typically 5% to 20%. AEM also provides capability to manage the energy usage at the system level for POWER6™ processor-based systems, BladeCenter, and System x. Power capping allocates a certain amount of energy to a system and is based on either historical trending or on a percentage of the maximum energy draw. With power capping, AEM can help keep the energy use for these systems under the maximum level that you set by throttling back the clock speed of the CPU, lowering the voltage, or both.
Agent-less energy monitoring

AEM is unique in that you don’t have to install any agents on the servers it monitors or manages. Monitored or managed servers and devices are referred to as objects. AEM employs type 0 agent-less technology to communicate with these objects. The first step in setting up the agent-less communication is to use the discovery process of IBM Director. Before AEM can monitor and manage a system or device, IBM Director Server must discover that object. The discovery process establishes communication between objects to be managed and IBM Director with the communication method unique for each server hardware platform.

AEM and the HMC

With System z, which is managed by the hardware management console (HMC), IBM Director discovers and communicates with the HMC through the Simple Network Management Protocol (SNMP). After adding the System z HMC, AEM uses the SNMP protocol to discover the servers managed by this HMC that, in turn, AEM can manage. If AEM discovers a System z10, it is added to AEM’s console window as an object. (AEM cannot monitor older System z servers.)

When communication between the objects and AEM occurs, AEM talks directly to the HMC to obtain the energy data for the System z10. The data is passed up to AEM through the HMC from the Support Element (SE), which in System z10 communicates with the z10™ power subsystem to receive and store sensor information in a snapshot. The System Activity Display (SAD) application on the SE takes the snapshot data, optionally displays it on the SAD, and forwards data to any HMC that communicates with the SE. When AEM samples the data, it communicates with the HMC that uses the upstream SNMP interface to pass the information to AEM.

Sampling data

By default, AEM samples data from each object once a minute. You can set and increase the polling interval for each object, so that data is collected from a given object less frequently. AEM can be monitoring hundreds of objects. The frequency of polling affects the disk space for storing the historical data, the refresh rate for the AEM console, network traffic to gather the data, and the system resources to run AEM. If only basic energy data is required, polling every five...
or ten minutes is probably sufficient for BladeCenter, Power Systems, and System x. However, System z servers contain a cache that holds one hour of input power data so the polling interval can be set as high as 60 minutes without having any gaps in the input power data that AEM collects.

AEM saves the power, temperature, and event information in a private database for a default 31 days. The space for the database depends on the number of objects being monitored and the frequency of polling. Assuming the default of 1-minute polling, the 31-day space is approximately 3.5 GB for every 100 managed objects.

If you plan to manage more than 1000 objects, allocate the database on a logical disk that is striped over multiple drives to allow I/O spreading. Figure 1 shows an AEM console display of energy trends for a system z10 server (named H73) and reports energy and thermal data over approximately two hours.

---

**AEM Console**

AEM is an IBM Director extension that provides the Active Energy Manager task in an IBM Director environment. When you start the Active Energy Manager task, the AEM console opens a window for managing all objects that IBM Director has discovered and that AEM supports. That can be quite a few objects, so you can launch multiple AEM console windows, each with a unique set of devices. The AEM console window contains a menu bar, a tool bar, and a graphical representation of energy consumption and thermal data for selected managed objects, with objects displayed in a tree structure.

**Interpreting the data**

The energy data that AEM reports for System z is equivalent to the data that SAD reports, which provides a single screen for statistics and a graph for processor utilization. The SAD displays on the SE and on the System z HMC. AEM is able to not only capture those same data points from the System z10 but also display and accumulate the data for trend analysis and for exportation. Although the SAD display works on System z servers before z10, it does not provide a view to the energy use by other systems in the data center and does not save information over time. AEM provides a data center view of your energy use and allows you to perform trend analysis and cap energy use of certain servers.

You can use the trending data from AEM in a number of ways, most obviously, verifying actual energy use and assessing energy needs for servers and the implementation of support structures based on facts rather than label power, using that assessment for cooling and uninterrupted power supply (UPS) sizing. Moreover, you can use the trending data from various platforms to optimize energy use (for example, analyzing performance per watt statistics with performance monitoring tools) and perform comparative energy use analysis (for example, analyzing on a system by system basis or extrapolating down to a single watts/logical image statistic).

---

**Conclusion**

Verifying energy efficiency improvements is needed for Energy Efficiency Certificate eligibility. As virtualization technologies become more pervasive in data centers, you can use the system level energy information from AEM to start more granular comparisons including watts per logical server image.

You can monitor energy use with AEM at no charge. Try it by going online to the IBM Director site and downloading the no-charge IBM Director for Linux® on z and the no charge AEM extension at [ibm.com/systems_management/director](http://ibm.com/systems_management/director).
All about EAV
An overview of the extended address volume

BY JAN REDDING, JAMES CAMMARATA, AND CECILIA CARRANZA LEWIS

Editor’s note: EAV gets the movie star treatment in this issue of z/OS Hot Topics. Start here for the basic concepts of the new technology. The authors explain how z/OS V1R10 and the new 3390 Model A device can help meet your growing data storage needs. Then read the companion (and equally cinematically themed) article, “The three phases of EAV” on page 20, for specifics about deploying EAV in your installation.

Are terabytes bursting the seams of your DASD? Are you running short of device numbers or subchannels? Is your DASD farm becoming a DASD jungle?

Then it’s time for your installation to implement the extended address volume, or EAV for short.

The world before EAV
z/OS has long used a disk architecture called count-key-data (CKD), which has been superseded in more recent years by extended CKD (ECKD™). This architecture divides DASD storage into tracks and cylinders. Storage is addressable through a combination of cylinder number and head number.

Space assignments on the disk are maintained in a structure called the volume table of contents or VTOC. (If any of this is news, we recommend that you read up on DASD volume architecture in the book, z/OS DFSMSdfp Advanced Services, SC26-7400.)

Why rehash all this? Before EAV, the largest DASD volume was limited to about 54 GB, or 65,520 cylinders. The range of addressable tracks restricted the growth of DASD volumes, which in turn constrained the system’s limits on the amount of storage you could define.

With EAV technology, these constraints are history.

What is an EAV?
In z/OS V1R10, support for EAV breaks the barrier of 65,520 cylinders. We’ve enabled this expansion by introducing new track and cylinder addresses that can accommodate hundreds of terabytes for a single volume.

In terms of hardware, EAV function is supported on the 3390 Model A, a new storage device that is available on the IBM System Storage™ DS8000™. The 3390 Model A is a variable capacity device that you can configure for one to 268,434,453 cylinders. See Figure 1.

In z/OS V1R10, an EAV is limited to 262,668 cylinders or about 223 GB — four times the capacity of the largest DASD volume previously supported.

Along with the new technology come new terminology, new track addressing, new DSCBs, new macros, and a new callable service.

Track addressing: Big C’s, little c’s, and fat H’s
An EAV has two addressing spaces, the base addressing space and the extended addressing space (EAS). To avoid confusing these with virtual storage, we do not use the term “address space.”

Base addressing space
The base addressing space is the area on an EAV that consists of the first 65,536 cylinders. Tracks in the base

EAV: A volume with more than 65,520 cylinders

3390 Model A: A device configured to have 1 to 268,434,453 cylinders
DS8000

3390-A “EAV”

3390-3 3390-9 3390-9 3390-9
3 GB 9 GB 27 GB 54 GB
Max cyls: 3,339 Max cyls: 10,017 Max cyls: 32,760 Max cyls: 65,520
100s of TBs

Figure 1. Maximum volume sizes
addressing space are addressed with 16-bit cylinder numbers, using the notation **CCCCHHHH**, where each letter is a hex nibble (two characters per byte). It works like this:

- The big C’s (CCCC) represent 16-bits for a cylinder address.
- The fats H’s (HHHH) represent 16-bits for a track address, of which only the low-order 4-bits are used to describe 15 tracks per cylinder.

Until EAV, all DASD volumes were addressed using these 16-bit cylinder addresses. With EAV, we put the fat H’s on a diet.

**Extended addressing space**

The extended addressing space (EAS) is the area on an EAV above the first 65,536 cylinders. A track in this area is addressed with a 28-bit cylinder number, using the notation **CCCCcccH**. This track address format is used by the access method, extent descriptors, channel programs, and in the DS8000, to access a track.

The seven cylinder nibbles contain a 28-bit cylinder number, but it is not a 28-bit binary number. Rather, it works like this:

- The CCCC portion of the address is the low-order 16-bits of the cylinder number.
- The ccc portion of the address is the high-order 12-bits of the cylinder number. (The cylinder nibbles must be rearranged for it to represent a 28-bit binary number.)
- The H represents the trimmed down 4-bit track number.

The 28-bit cylinder address is a non-contiguous number. To read this track address (for example, to make comparisons and perform arithmetic operations), you must first rearrange the seven hex digits of the cylinder number.

The 28-bit cylinder address is compatible with a 16-bit cylinder address. A device with up to 65,520 cylinders will always have its little c’s set to zero.

However, this new addressing format is not compatible with the 16-bit track address. The twelve high-order bits of the cylinder number are in the twelve high-order bits of the two old fat HHHH bytes.

Thus, any existing programs that work with the track address must be changed to handle the new address format. To help you with this change, z/OS V1R10 adds a new macro, TRKADDR, and callable service, IECTRKAD, to perform these manipulations.

A program that understands 28-bit cylinder numbers will work equally well on volumes that always have zeros in their little c’s. Thus, the program will work with all existing non-EAV volumes. Got it? Try the EAV question (sidebar), which demonstrates why your program needs to consider the new address format.

**How can I use EAS?**

On an EAV, an EAS-eligible data set is one that is eligible to have extents in the extended addressing space when it is created or extended by the system. Any data set can reside in the base addressing space on an EAV, but only EAS-eligible data sets can reside in the extended addressing space.

In z/OS V1R10, VSAM data sets are EAS-eligible. This includes all VSAM data types (KSDS, RRDS, ESDS and linear) for both SMS-managed and non-SMS-managed data sets.

The following data sets are not EAS-eligible in z/OS V1R10:

- Catalog data sets (BCS and VVDS)
- VTOC (resides within the first 64K-1 tracks)
- VTOC index
- Non-VSAM data sets
- Page data sets
- VSAM data sets with imbed or key range attributes
- VSAM data sets with incompatible CA sizes (other than 1, 3, 5, 7, 9, and 15 tracks).

**Space management on an EAV**

An EAV has two managed spaces (Figure 2). **Track-managed space** is the area of a volume located in the first 65,520 cylinders. Space is allocated in track or cylinder increments. All volumes today have track-managed space and each data set occupies an integral multiple of tracks.

**Cylinder-managed space** is the area on an EAV located above the first 65,536 cylinders. Cylinder-managed space is allocated in multicylinder units. For an EAV, the multicylinder unit is 21 cylinders. Space requests directed to the cylinder-managed space can be rounded up to the next multicylinder unit. Note that an EAS-eligible data set could have extents in both spaces.

Track-managed space and cylinder-managed space describe how space is managed, while the base addressing space and extended addressing space describe how the disk is addressed using the new track address format.
Before EAV, the largest DASD volume was limited to about 54 GB, or 65,520 cylinders.

Why 21 cylinders for an EAV?
The 21-cylinder multicylinder unit is the smallest unit that can be used to map the largest possible EAV and remain within the VTOC index architecture (with the new index block size of 8192 bytes). It is also a value that divides evenly into the 1 GB storage segments of a DS8000. A segment, which is the allocation unit in the DS8000, is equivalent to 1113 cylinders. Segments are allocated in multiples of 1113 cylinders above 65,520.

Break point value
The break point value (BPV) directs the placement of EAS-eligible data sets on an EAV, as follows:

- Cylinder-managed space is preferred if the requested space is greater than or equal to the BPV.
- Track-managed space is preferred if the requested space is less than the BPV.

You specify the break point value as the BPV keyword in the IGDSMSxx member of parmlib, or through an SMS storage group. By default, this value is 10 cylinders.

New DSCBs in the VTOC
EAS-eligible data sets allocated on an EAV are created with new types of data set control blocks (DSCBs) in the VTOC. The new format DSCBs are called extended attribute DSCBs, or format 8 and 9 DSCBs.

A format 8 DSCB is equivalent to a format 1 DSCB. It describes the first three extents of the data set and contains a chain pointer to a format 9 DSCB.

A format 9 DSCB provides for additional attribute data and a set of pointers to possible format 3 DSCBs. It also contains a chain pointer to the next DSCB for this data set, which is either the next format 9 DSCB, the first format 3 DSCB (if the data set has more than three extents) or zero.

Suppose, for example, that you have an EAS-eligible data set with 123 extents on an EAV. Here, the data set has one format 8 DSCB, one format 9 DSCB, and ten format 3 DSCBs to hold all the extents.

The purpose of these new extended format DSCBs is to indicate that the extent descriptors contained in a format 8 or format 3 DSCB might contain 28-bit cylinder numbers. The new format DSCBs help to prevent programs from accidentally attempting to use the old track-addressing method on a volume that has extents using the new 28-bit cylinder address.

Instead, programs that use system facilities like CVAFDIR, CVAFSEQ, CVAFFILT and OBTAIN must now specify the new EADSCB=OK keyword. This keyword indicates that the caller supports 28-bit cylinder addresses and format 8 and 9 DSCBs.

For more information about changes to these macros, the new keywords and format 8 and 9 DSCBs, see z/OS DFSMSdfp Advanced Services, SC26-7400.

How do I get started?
Now that you have learned the EAV concepts and terminology, see the article “The three phases of EAV” on page 20 for specifics about deploying EAV in your installation.
Extended address volume or EAV gets the spotlight in this issue of z/OS Hot Topics. In “All about EAV” on page 17, we explained the basic concepts of this new storage technology. If you are involved in projecting the year in which your installation will reach addressing constraints, you are probably relieved to know that z/OS and EAV can accommodate your critical data estimated growth rate. In fact, you probably can’t wait to use them.

Without further ado, we now look at an orderly approach (the three phases) for deploying EAV on your system. With some careful planning and these simple steps, your installation will be off and running with additional storage flexibility.

**Phase 1: Planning and system setup**

You begin by determining what data you want to store on your EAV. Any data set can reside on an EAV, but only VSAM data sets (including those accessed by DB2®, IMS™, and zFS) can take advantage of the extended addressing space (EAS). The EAS is allocated in multicylinder units, therefore larger data sets are the best candidates for that space.

Also, you must ensure that any data placed on an EAV will not require access from a lower-level system. An EAV can come online only to a z/OS V1R10 or later system.

Next, determine whether any application changes are required. “Whoa! Did someone say application changes?” Yes, because of some fundamental changes in system structures, some applications might require updates to access data in the EAS of an EAV. Data sets in the EAS are described by extended attribute data set control blocks (format 8 and format 9 DSCBs). Your applications might require updates to prevent them from inadvertently using incompatible extents.

Fortunately, help is available for making this determination. With the tracking facility, you can identify applications that might require updates to work with EAV. The tracking facility reports the job, program name, and offset within the program where a change might be needed. The tracking facility classifies these areas for further investigation as instances, which are described as follows:

- **Error instances.** These identify programs that would fail if actually run on an EAV with a data set that has a format 8 or format 9 DSCB. For example, the use of the OBTAIN macro that does not specify the EADSCB=OK keyword will fail on reading a format 8 or format 9 DSCB. The EADSCB=OK keyword indicates to the system that the calling program can handle extended attribute DSCBs and the 28-bit cylinder numbers that might be present in the extents described in those DSCBs.
- **Warning instances.** These identify programs that might improperly use returned information. For example, the LISTCAT output contains the extent descriptors in their native 28-bit cylinder format. A program that parses the extent descriptors in the LISTCAT output must be changed to handle 28-bit cylinder numbers.
- **Informational instances.** These identify programs that might benefit from using new services. For example, the EXPDATA option on the LSPACE macro returns free space statistics for a volume’s track-managed space.

Figure 1 provides a summary of the error, warning, and informational instances.
Just because a program is listed in the tracker report does not mean there is a problem. The report simply identifies programs to be checked for potential updates. IBM provides an exclusion list, which you can customize, to prevent the tracking facility from flagging instances that were already checked. The tracking facility can also check vendor programs. You can send flagged instances in IBM or vendor programs to IBM for further investigation.

If for some reason you cannot modify an application that is identified as requiring changes, you must ensure that you do not introduce EAV into any storage groups that are used by the application.

For information about how to use the tracking facility and where to send the instances, see z/OS MVS Planning: Operations, SA22-7601. For details on the system function invocations that might be recorded, see z/OS DFSMSdfp Advanced Services, SC26-7400.

After you have determined the data to be placed on EAV and verified that the applications using this data can work with EAV, you can begin your system setup:

- To allow the system to allocate new data sets on an EAV, update the SMS parmlib member, IGDSMSxx, to specify USEEAV=YES.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.

After you have determined the data to be placed on EAV and verified that the applications using this data can work with EAV, you can begin your system setup:

- To allow the system to allocate new data sets on an EAV, update the SMS parmlib member, IGDSMSxx, to specify USEEAV=YES.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.
- If necessary, update the storage group definition to add the HIGH and LOW threshold percentages for the track-managed space of the EAV. These new options allow the system to use different percentages for track-managed space when performing DFSMShsm™ space management and SMS volume selection.

Before expanding a volume, you must withdraw any copy service relationships in which the volume is participating. After you expand the volume, run an ICKDSF reformat job to update the VTOC and index of the new space. Running an ICKDSF reformat job will enable the system to use the expanded space.

For more about the DVE function, see “Dive into Dynamic Volume Expansion” on page 28.
Creating a new volume

You can create a new EAV using the DS8000. This can be done through the graphical user interface (GUI) or command-line interface (CLI) commands. Here’s an example of creating a CKD volume using the `mkckdvol` CLI command:

```
dscli> mkckdvol -extpool p2 -cap 262668 0080
Date/Time: 08 April 2007 17:04:31
IBM DSCLI Version: 5.1.0.204 DS: IBM.2107-7503461
CMUC00021I mkckdvol: CKD Volume 0000 successfully created.
```

After creating an EAV:

- Use Hardware Configuration Definition (HCD) to define the EAV address to z/OS.
- Use ICKDSF to initialize the volume: VTOC, VTOC index, and allocate the VVDS.
- Add the EAV to the appropriate SMS storage groups.

After validating the configuration, enter the SET SMS=XX command to activate the changes to SYS1.PARMLIB. Finally, vary the EAV online.

Remember...

Whether expanding an existing volume or creating a new volume, after the EAV are visible in the storage group, SMS volume selection will begin using them for data set allocations and extends. You are now ready to begin using the additional capacity — what a relief!

Well, almost ready...

Phase 3: Migrating data

For many installations, especially those constrained by device numbers, there remains the very important step of migrating existing data onto an EAV. Over the years, the challenge of migrating data to larger volumes has been the topic of numerous white papers and guides.

Migrating data to EAV is essentially no different from what you have done for other volumes in the past. A typical migration procedure begins with a physical volume level copy from a source volume to a larger target volume. You then continue with data set level copy functions.

When migrating to EAV, be aware that using a physical volume copy operation will keep the VSAM data sets in the track-managed space, while a logical data set copy operation can move the data set into the cylinder managed space. This will allow more space for non-VSAM data sets in the track managed space. Also, a logical copy will tend to result in the allocation of larger extents for the target data set.

All DFSMS™ functions typically used during volume migration procedures will support EAV in V1R10. Other popular IBM migration products, such as Softek Transparent Data Migration Facility (TDMF™) and z/OS Dataset Mobility Facility (zDMF), also support EAV. For non-IBM products, contact the vendor to ask about their plans for supporting EAV.

Businesses like yours continue to search for the “silver bullet” that allows migration to larger volumes without the need for application outages. With careful planning, it is possible to obtain this goal in many cases, and to keep disruptions to a minimum.

Just the beginning

The introduction of EAV in z/OS V1R10 is just the beginning of a great story. As EAV grow over time toward their architectural limit of 100s of TBs per volume, you cannot help but wonder about the vast amount of data that could exist in your installation.

With some careful planning and these simple steps, your installation will be off and running with additional storage flexibility.

Just think, soon there will be little reason to stay awake at night worrying about your ability to contain storage growth.

Businesses like yours continue to search for the “silver bullet” that allows migration to larger volumes without the need for application outages. With careful planning, it is possible to obtain this goal in many cases, and to keep disruptions to a minimum.

Just the beginning

The introduction of EAV in z/OS V1R10 is just the beginning of a great story. As EAV grow over time toward their architectural limit of 100s of TBs per volume, you cannot help but wonder about the vast amount of data that could exist in your installation.

With some careful planning and these simple steps, your installation will be off and running with additional storage flexibility.

Just think, soon there will be little reason to stay awake at night worrying about your ability to contain storage growth.
Sex and the 64-bit virtual environment

BY DONault

Okay, I lied; this article has nothing to do with sex unless you are an introverted programmer (like me) who considers exploiting 64-bit virtual sexy.

When you need a chunk of virtual storage below two gigabytes (“the bar”), you use the GETMAIN macro or the STORAGE OBTAIN macro. The cost to do so is a little expensive, so you might resort to writing your own storage manager to get one large chunk of virtual storage and carve it up among the individual requestors.

For 64-bit virtual, z/OS provides the IARV64 GETSTOR service, which doles out storage in 1 MB increments. Unless your application needs megabyte hunks of storage (say, for a giant buffer or a huge array), you need to write your own storage manager to hand out smaller pieces of storage.

In z/OS V1R10, we have added two new services to help you manage 64-bit virtual storage:

• IARST64, which performs a service similar to the GETMAIN macro or the STORAGE OBTAIN macro. It allows you to request a variable size hunk of virtual storage.
• IARCP64, which performs a service similar to the CPOOL macro. It allows you to create a cell pool of fixed size elements.

Before we get into the details, let’s look at why and when you would use these new services.

Why use 64-bit virtual

For continued growth, all of us — the operating system, IBM products, vendor products, and your applications — require storage. If we are obtaining common storage from CSA or SQA subpools, however, we are consuming a rapidly shrinking resource: virtual storage below the bar. If we are obtaining private storage, the old 2 GB limit minus common storage, is sometimes just not enough.

To permit more growth, we all need to consider moving our older control blocks from below the bar to above the bar. In addition, we should design any new users of common storage to exploit 64-bit common storage. (For details on the IBM efforts to reduce our own common storage footprint, see “Virtual storage constraint relief: Apply directly to your workloads” on page 25.)

Once you choose to take advantage of 64-bit common, your choice of services boils down to one of the following approaches: using the IARST64 service, using the IARCP64 service, or using the IARV64 service and writing your own 64-bit virtual storage manager.

Each address space has two gigabytes, minus the size of common for private storage below the bar. In some applications, either the growth of the application or the growth of common storage is causing the application to run into the 2 GB limit.

When this happens, your choices are to exploit data spaces or 64-bit virtual. Data spaces provide up to 2 GB of virtual storage at a time. With 64-bit virtual, the amount of storage is, practically speaking, limited only by your imagination and your company’s IT budget for real and auxiliary storage.

After you decide to exploit 64-bit virtual, the choices are the same as for 64-bit common. You can use the IARST64 service, the IARCP64 service, or you can write your own 64-bit storage manager to carve up the megabytes you obtain with IARV64.
Attributes of IARST64 and IARCP64

These new services exploit common code internally to manage 1 MB chunks of virtual storage that are obtained with IARV64. These services do not use the subpool convention that exists for GETMAIN or STORAGE OBTAIN.

To use them, you specify the storage attributes you want, which avoids the subpool guessing game. You specify:

- Common or private storage
- Storage key 0-7 for common and 0-15 for private
- Fetch-protected or not fetch-protected
- Pageable, DREF or FIXED.

When requesting private storage, you can specify OWNINGTASK=CURRENT, MOTHER, IPT, JOBSTEP, CMRO, or RCT. This covers most of the options one would need for controlling ownership of private storage. When the owning task ends, the storage is freed.

Supported environments

You can get storage in TCB or SRB mode, any address space control (ASC) mode, cross-memory and for fixed storage, you can even get storage while disabled for interrupts. Details of which locks can be held are provided in the macro prologs.

To permit more growth, we all need to consider moving our older control blocks from below the bar to above the bar.

Because these services are based on 1 MB extents, they have size restrictions. IARST64 supports requests for 1 to 64K bytes. IARCP64 supports cell sizes up to about half a megabyte.

Ownership of storage

When requesting common storage, you can specify an owner in a way similar to how you do it through STORAGE OBTAIN. You can specify OWNER=HOME|PRIMARY|SECONDARY|SYSTEM, or you can specify the address space identifier (ASID).

This approach does not tie-in completely with common storage tracking, but it does allow you to identify abandoned storage through IPCS formatting support. As with other common storage, this storage is not automatically freed when the owning address space ends.

Bells and whistles

With the new services, we attempted to provide as many of the benefits you have now with STORAGE OBTAIN, but without the heavy instruction cost. For example:

- Depending on the cell size, boundaries are automatically forced to quadword, cache line, or page.
- To detect overruns at time of the FREE request, trailers are used when they fit.
- Double freeing of storage is detected and rejected with an abend.

Lusting for more information?

See the macro prologs and the z/OS MVS Assembler Services publications.
Configuring and tuning a z/OS system for optimal performance and reliability often means making a trade-off between the size of the common area and the size of the private area. Here, you might find yourself in the infamous "virtual storage balancing act." As you try to maintain the minimum required private area for a specific application, you must also ensure that the common area remains large enough to maintain the overall health of the system.

In this article, we discuss the steps that IBM is taking to provide virtual storage constraint relief (VSCR) in the z/OS operating system. You can use the same techniques at your installation to provide VSCR at the application level.

Curbing our appetite for common
To help address this platform-wide growth constraint issue, IBM has launched a coordinated effort to reduce the system's use of common storage. To enable future z/OS images to support larger workloads, we plan to move select system control blocks into an "above the bar" common area, so that more of the premium storage below the bar can be maintained for private address space utilization.

There is also a focus on limiting or reducing the use of common storage by z/OS products. Estimates indicate that an average reduction of 500 KB for 24-bit common areas and 40 MB for 31-bit common areas per year is necessary on the z/OS platform. Because all z/OS products use these areas to some degree, all z/OS products have been asked to contribute to the reduction effort. Specifically, each product development area was asked to reduce its net usage of both 24-bit common storage and 31-bit common storage.

Moving our data to higher ground
Data traditionally stored in the common area needs to move elsewhere, for example, to the private area, 64-bit shared, 64-bit common, or to a common area data space (CADS).

Where to move the data depends on the attributes of the storage area. z/OS components typically use the common area for the following reasons:

- Make data accessible to other address spaces
- Fix data in storage for I/O operations
- Reference data when disabled for waits.

Today, no 64-bit services exist to satisfy all of these requirements because:

- 64-bit private is not automatically accessible to other address spaces
- 64-bit shared requires other address spaces to request access. In addition, it does not allow data to be fixed in storage.

What and where to move
Determining which system control blocks to move, and where, requires that we first answer a number of questions to understand why the data resides in the common area, such as:

- How is the data accessed?
- Does the data need fixed storage, for example, for I/O?
- Does the data need disabled reference storage?
- Must the data be splittable?
- Is the data input to services that accept only 31-bit or non-access list entry token (ALET) qualified parameters?
- Must the data remain in 31-bit common storage, for example, lock words?
- Do other services or functions require that some of the data remain in 31-bit common storage? For example, data used as input to services that accept only 31-bit or non-access list entry token (ALET) qualified parameters.
Some control blocks contain parameters that system services require to be in 31-bit common storage (such as lock words). Consider separating the data that can be moved into 64-bit virtual or dataspaces from the data that must reside in 31-bit common storage.

To help you apply VSCR techniques at your installation, we describe the attributes of each storage area in the sections that follow. Figure 1 provides a summary of these considerations.

Considerations for private area
Private area storage has the following attributes:
- Other address spaces can access this storage using an ALET on the caller’s primary address space access list (PASN-AL) or dispatchable unit access list (DU-AL).
- Can be fixed; can be disabled reference (DREF).
- Owned by a task or address space.

Considerations for 64-bit shared
64-bit shared storage has the following attributes:
- Allows sharing above the bar across multiple address spaces (except when allocated through the IARV64 GETSHARED service). Each address space must explicitly obtain access through the IARV64 SHAREMEMOBJ service.
- Cannot be fixed or DREF.
- Owned by the system; must be explicitly freed.

Considerations for 64-bit common
64-bit common storage has the following attributes:
- Allows sharing above the bar across all address spaces without requiring explicit requests.
- Only authorized callers; must be running with PSW key 0-7.
- Can be fixed; can be DREF.
- Owned by the system; must be explicitly freed by the requestor.

To help you apply VSCR techniques at your installation, we describe the attributes of each storage area...

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Can be accessed by one address space</th>
<th>Can be accessed by a subset of address spaces</th>
<th>Can be accessed by all address spaces</th>
<th>Supports disabled reference (DREF) access</th>
<th>Supports fixed storage</th>
<th>Owned by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>Ideal for this purpose.</td>
<td>Possible, but not ideal.</td>
<td>No</td>
<td>Yes for 31-bit. No for 24-bit.</td>
<td>Yes</td>
<td>Task or address space.</td>
</tr>
<tr>
<td>64-Bit Shared</td>
<td>Possible, but not ideal.</td>
<td>Ideal for this purpose.</td>
<td>Possible, but not ideal.</td>
<td>No</td>
<td>No</td>
<td>System, but must be freed by caller.</td>
</tr>
<tr>
<td>64-Bit Common</td>
<td>Possible, but not ideal.</td>
<td>Yes, but overlays are possible.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>System, but must be freed by caller.</td>
</tr>
<tr>
<td>CADS</td>
<td>Possible, but not ideal.</td>
<td>Yes, but limited to 2GB per CADS.</td>
<td>Yes, but limited to 2GB per CADS.</td>
<td>Yes</td>
<td>Yes</td>
<td>Application storage manager.</td>
</tr>
</tbody>
</table>

Figure 1. Finding a new home for system data
To enable future z/OS images to support larger workloads, we plan to move select system control blocks into an “above the bar” common area.

Considerations for CADS
The CADS has the following attributes:
• Accessible by any address space in access register (AR) mode using an ALET on PASN-AL.
• Limited to 2 GB of virtual storage.
• Limited to 250 instances, or less, depending on the value of the MAXCADS parameter of the IEASYSxx member. The maximum number of ALETs on a PASN-AL is 510. A larger MAXCAD value means fewer entries are available for other data spaces. Changing the MAXCAD setting requires an IPL.
• Can be fixed; can be DREF.
• Owning address space must be non-swappable.
• Managed by a storage manager that is supplied by the application.

Above the bar with 64-bit common services
In z/OS V1R10, a new option REQUEST=GETCOMMON was added to the existing IARV64 service. This service manages 64-bit common storage in much the same way as the system manages the existing 31-bit common storage.

Observe the following considerations for using the IARV64 REQUEST=GETCOMMON service:
• Storage is managed as memory objects that a program creates. A memory object is a contiguous range of virtual addresses allocated by programs in 1 MB multiples on a 1 MB boundary.
• 64-bit common memory objects are visible to all address spaces at the same address, and are accessible to all address spaces.
• The requestor must explicitly free 64-bit common memory objects; they are not freed by the system when the requestor or creator goes away.

Other enhancements for 64-bit common services
For diagnostic purposes, you can associate an owner with the memory object at the time of its creation. You can assign ownership to the system, the home address space, the primary address space, or any other address space in the system. For storage tracking purposes, all the same data reported by CSA tracker is retained for 64-bit common.

In z/OS V1R10, the new IEASYSxx HVCOMMON keyword is used to specify the size of 64-bit common storage in multiples of 2 GB. The default size for 64-bit common is 64 GB, with a minimum size of 2 GB and a maximum size of 1 TB (one terabyte).

SVC dump services was enhanced to dump 64-bit common storage. Specify the LIST64 keyword to allow the dumping of 64-bit common storage. In addition, when you create 64-bit common memory objects, you can:

• Specify the SVCDUMP=LIKECSA attribute to have these objects dumped for dump requests that specify SDATA=CSA.
• Specify the SVCDUMP=LIKESQA attribute to have these objects dumped for dump requests that specify SDATA=SQA.

The system monitors the 64-bit common area and issues appropriate messages when the amount of allocated storage exceeds certain thresholds. This processing is very similar to system monitoring of the 64-bit shared area.

Lastly, Resource Management Facility (RMF™) was enhanced to display 64-bit common storage usage statistics. SMF records have new fields that report these statistics. The statistics include information such as:
• Amount of 64-bit common storage allocated
• Number of real frames backing 64-bit common storage
• Number of auxiliary slots used for 64-bit common storage.

Find out more
For information about using 64-bit common storage, see "Using the 64-bit address space" in z/OS MVS Programming: Extended Addressability Guide, SA22-7614.

For information about using the IARV64 macro, see the topic on using a 64-bit address space in z/OS MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDITINFO-IXGWRITE), SA22-7610.
Are you looking for a simpler method to expand your z/OS DASD volumes? Does creating a separate target volume consume precious device numbers? Do you need to expand volumes quickly, avoiding the time-consuming copy step? If so, then we have a solution for you.

With the introduction of Dynamic Volume Expansion (DVE), increasing the size of volumes just became easier. No longer do you need to allocate a separate larger volume and copy the original volume to the larger volume. Nor do you need to use data migration software to allow applications continued access while the volume is being copied to the larger volume. DVE provides the ability to grow the capacity of a volume while the volume remains online to host applications.

Let’s get started
Assuming that you have installed the appropriate DS8000 LIC and software PTFs (see the end of this article for details), you’re all set to expand a volume with DVE. You can do this using the IBM DS8000 command line interface or graphical user interface. We’ll do it using the command line interface’s CHCKDVOL (Change CKD Volume) command. Because the volume capacity we will request would exceed EAV capacities, we’ll first change the volume model. The following example changes the DS8000 volume with identifier 1A34 to a 3390 model A:

```
CHCKDVOL –dev [machine-serial] –datatype 3390-A 1A34
```

Next, we use the CHCKDVOL command to expand the capacity of a volume. The following example expands the DS8000 volume with identifier 1A34 to 262,668 cylinders:

```
CHCKDVOL –dev [machine-serial] –cap 262668 1A34
```

This causes the DS8000 to allocate additional space on the disk arrays to volume 1A34. After the volume resize occurs, a state change interrupt is presented to all attached hosts systems. On z/OS systems, message IEA019I displays on the console to indicate that it’s necessary to update this volume’s VTOC and INDEX using ICKDSF.

```
IEA019I D018,TMP121, VOLUME CAPACITY CHANGE, OLD=00065520, NEW=00262668
```

You can then run ICKDSF to update the VTOC and VTOC INDEX.

```
//EXAMPLE JOB
// EXEC PGM=ICKDSF
//VOLDD DD DISP=SHR,UNIT=3390,VOL=SER=TMP121
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
//REFORMAT DNAME(VOLDD) VERIFY(TM121) REFVTOC /*
```

The event notification facility (ENF) broadcasts the volume capacity change across the sysplex. After updating the volume’s VTOC and VTOC INDEX, ICKDSF issues an ENF64. Two listeners of the ENF64 are DFSMShsm and DFSMSdip™.

When notified of a volume expansion, every DFSMShsm host in the sysplex updates its records with the new capacity and fragment index for the expanded volume. This is indicated with message ARCO602I.

```
ARC0602I: THE CAPACITY OF VOLUME volser WAS SUCCESSFULLY UPDATED TO nn CYLINDERS
```
DFSMShsm performs functions such as backup, recovery, migration, and recall based upon the new volume capacity. DFSMSdfp also listens for the ENF64 and uses it to update information about the location and size of the VTOC and VTOC INDEX on a volume. The newly acquired space can be used when new data sets are allocated or existing data sets are extended.

Free space information for system-managed storage is obtained by SMS (through the Common VTOC Access Facility) whenever the amount of free space changes for a volume. SMS volume-selection algorithms use this information to select an appropriate device on which to place a new data set.

Software vendors can also listen for ENF64 by using system macro IECENF64.

**Volume model**
A 3390-3 can be expanded up to 3339 cylinders. If the expanded size is greater than 3339 cylinders, the volume model will be converted to a 3390-9 during the resize. A 3390-9 can be expanded to 65,520 (X’FFF0’) cylinders.

With z/OS V1R10, a volume can be expanded to sizes larger than 65,520 cylinders. Volumes of this size are known as extended address volumes (EAV), and the volume model is 3390-A. A 3390-A can be expanded up to 262,668 cylinders (X’4020C’).

Remember that the volume can remain online during the entire volume expansion procedure. However, any copy services relationships (FlashCopy®, PPRC, or XRC) must be quiesced prior to volume expansion. The hardware will fail an attempt to expand a volume with active copy services relationships.

**Take the plunge!**
As long as your system meets the following hardware and software requirements, there’s nothing easier than diving into DVE. So go ahead, and see how easy it is to increase the size of volumes. For more information on the CHKDVOL command, see IBM System Storage DS8000 Command-Line Interface User’s Guide, SC26-7916.

**Hardware requirements**
- M/T2107 Release 3.

**Software requirements**
- z/OS V1R7 or later
- z/OS V1R10 (for EAV support)
- PTF for APAR OA20043 (for SMF 22 record and ENF64 support)
- PTF for APAR OA21113/OA21216 (for device support)
- PTF for APAR PK57851 (for ICKDSF)
- PTF for APAR OA21574 (for DFSMShsm).

**With the introduction of Dynamic Volume Expansion, increasing the size of volumes just became easier.**
A security administrator and a wizard were back in their favorite bar one day. “Thanks for all your help understanding RACLIST”, the security administrator said to the wizard (see “Rediscover the magic of RACLIST” in z/OS Hot Topics Newsletter Issue 18, February 2008, GA22-7501-14), “but there’s another RACLIST mystery that you might be able to explain for me. What is the benefit of setting up a RACGLIST class for a RACLISTed class? This just seems like an unnecessary step, and fills up my RACF database with extra junk.”

“Well,” said the wizard, “RACGLIST was introduced to address a challenge brought about by sysplex technology, so let’s start with a look there.”

The challenge of sysplex technology

The introduction of sysplex technology brought with it a whole new set of challenges for RACLISTing classes. The largest of these was trying to keep the RACLISTed information synchronized on all systems in a sysplex.

When one issues the RACLIST or RACLIST REFRESH command on each system in a sysplex, in the time between issuing the command and the RACLIST processing completing on each system, there is a possibility that changes made to the RACF database can result in different systems ending up with different profile information in their respective dataspaces. For example, a change which is not captured in the RACLIST on the first system is captured in the RACLIST on one of the later systems. So now the systems have slightly different data, which in turn generates slightly different security decisions.

“To illustrate the problem of maintaining the same RACLISTed information on multiple systems of a sysplex,” said the wizard, “consider these two photographers.” The wizard then waved his hands, and (with a flash of light and a puff of smoke) two photographers suddenly appeared.

The photographers each snapped their respective cameras, and instantly two fully developed prints magically appeared in the wizard’s hands. “Now, see here,” said the wizard, showing the security administrator the pictures. “These two pictures are very similar (I’m incredibly handsome in each and you have some lettuce between your teeth), but there are slight differences (your eyes are shut in this one, for example) because they were taken at slightly different moments from slightly different perspectives. In the case of two photographs, these little differences are acceptable. In the case of information RACLISTed on two systems of a sysplex, these little differences would lead to inconsistent security decisions and potentially big problems. Even small differences would cause the two systems to act differently.”

Here comes sysplex communication

When sysplex communication is enabled, certain RACF commands will be propagated between systems in the sysplex. Two of these commands happen to be the SETROPTS RACLIST(<class>) and the SETROPTS RACLIST(<class>) REFRESH commands. This does not remove the window of opportunity during which different systems can obtain slightly different data, but it greatly reduces that window by having the initial system in the sysplex propagate that command across each system in the sysplex. In the end, we still have the same problem, just a smaller window in which the problem can be created.

“When RACF creates the dataspace with the RACLISTed information,” said the wizard, “it is important to
understand that it does not simply copy the RACF profiles into the dataspace. Actual processing occurs while copying the profile data to help improve the performance of security decision making. This processing may be the real mystery of RACLIST processing,” continued the wizard, “and so I’m not going to demystify it. I need to keep some secrets from you,” the wizard winked, “or else I’ll be out of a job.”

The thing to keep in mind here is that some additional processing occurs (beyond just a simple copy) each time a RACLIST is done. So even with sysplex communication on, it’s as if you’re issuing the RACLIST command on each system, with each system doing (more or less) the same copying and processing of the class profiles being RACLISTed. Not only does this repetitive processing increase the window of opportunity for things to get out of sync.

“So isn’t there a way to avoid doing that repetitive processing on each system, and ensure data consistency?” asked the security administrator.

“Yes there is,” said the wizard smiling. “This brings us to RACGLIST.”

So what’s with RACLIST?
The RACLIST class was introduced to ensure that all systems would get a single view of RACLISTed data when sysplex communication is enabled. Creating a class profile in the RACLIST class is like telling the photographer to take a picture, and save the image to share with his friends.

With RACLIST, you first create a profile in the RACLIST class for a class whose RACLISTed information needs to be consistent across the sysplex. To activate RACLIST processing, all you need to do is turn the RACLIST class on, and create a profile for the class to be RACLISTed in the RACLIST class. For example, to have RACLIST processing set up for the SECLABEL class, enter the following commands:

\[
\text{SETROPTS CLASSACT(RACLIST)}
\text{RDEFINE RACLIST SECLABEL}
\]

Once RACLIST processing is set up, the next system to issue the SETROPTS RACLIST(<class>) or the SETROPTS RACLIST(<class>) REFRESH command to RACLIST the class populates the MVS dataspace as usual, but also copies the information in the dataspace back into the RACF database in the RACLIST class. This system also acts as a coordinator, instructing the peer systems in the sysplex to RACLIST the class through sysplex communication. When a peer system is told to RACLIST the class through sysplex communication, the dataspace created is populated using the information from the RACLIST class. It’s as if a single photographer took a picture, and then made that picture available to different people. The result is that everyone has an identical picture. In the case of a sysplex, this means that each system can make security decisions based on identical information.

And what’s more, repetitive processing is eliminated because only one system, the coordinator, performs the additional processing to improve the performance of security decision making.

The same advantage of RACLIST is realized when you IPL one system in the sysplex. With RACLIST set up, the data saved in the RACLIST class populates the dataspace at IPL time.

Some things to keep in mind
You should be aware of several things. Because a copy of the RACLISTed data will be stored in the RACLIST class, this will require a larger RACF database to basically store a second copy of all the profiles in the RACLISTed class.

It is also important to have sysplex communication turned on for RACF, otherwise the RACLIST command will not be propagated to every system in the sysplex. As a result, the only way to refresh the RACLISTed information would be to issue the RACLIST command on each system in the sysplex, which destroys the single system image you were trying to create within the sysplex.

Remember that when you have sysplex communication turned on for RACF, the RACLIST operation will be propagated to the peer systems in the sysplex. Therefore, you do not need to issue the SETROPTS RACLIST command on multiple systems. If you do, each of those invocations will propagate the RACLIST operation to the peer systems. As a result, each system ends up doing the RACLISTing for the same class multiple times.

“The final thing you need to keep in mind,” said the wizard, “is that it’s getting late. Let’s turn our attention to other matters.” And with that, the wizard waved his hands and conjured two large margaritas out of thin air (much to the annoyance of the bartender, who preferred it when customers actually bought their drinks). Raising his glass to the security administrator, he offered a toast, “To RACLIST!”
H ot Topics collectors out there (and we know there are many of you) remember that in “Check Please”, in z/OS Hot Topics Newsletter Issue 13, August 2005, GA22-7501-09, we described the new RACF checks introduced with z/OS V1R7 as part of IBM Health Checker for z/OS. These checks have proven to be popular with z/OS clients and have been extended with new resources in each subsequent z/OS release. With z/OS V1R10, RACF is introducing an exciting new type of check: the installation-defined RACF check.

Unlike other RACF checks where IBM decides the checking, in an installation-defined RACF check, you specify which resource to check. Do you have sensitive production data sets that you want to ensure do not have excessive access allowed to the "general users" of your system? No problem! An installation-defined check allows you to do that and more.

Check your steps
There are four steps in defining an installation-defined RACF check:

1. Decide which resources you want to check and the maximum access to allow a “general user”. The resources that you can check are data sets or resources in either IBM-defined or installation-defined general resource classes.

2. Define a RACF profile in the RACFHC class. Add your list of resources to this profile and specify the following information:
   - The resource name. You can check these types of resources:
     - Sequential data sets.
     - Partitioned data sets.
     - Libraries.
     - VSAM data sets.
     - RACF general resources. For member/grouping resources, the member name must be specified and the class must be RACLISTed.
     - The class of the resource (such as DATASET or FACILITY).
     - The maximum “general user” access. Optionally, for data sets, you can also specify the volume on which the data set resides.

   You can specify “pre-canned” resource lists, such as IRR_PARMLIB to examine all of the data sets defined in the current active parmlib. A full list of the available resource lists is in IBM Health Checker for z/OS User’s Guide, SA22-7994.

3. Define your installation-defined RACF check to the IBM Health Checker for z/OS. Here you associate the RACF profile created in step 2 to the specified resource names with your check. You’ll assign the severity, interval, check owner, and check name.

4. Run your check!

It’s really that simple! Let’s get into the details.

How installation-defined checks work
Each installation-defined RACF check can process a set of resources. The resources don’t have to be from the same class and do not have to have the same “maximum allowed general access.” You can have multiple installation-defined RACF checks defined, which is useful if you want to group your resources by application and allow unique access to the check output or if you want to assign different severities or intervals.

To define the resources for a specific check, you define a profile in the RACFHC general resource class that contains the list of resources for your check to examine. Figure 1 shows an example.
The next step is to register your check with the IBM Health Checker for z/OS by creating a parmlib entry for your check. (Figure 2.) Here you define all of the characteristics for your check and where you associate your check with the RACFHC profile that you created in Figure 1.

Place this check registration statement in the parmlib concatenation in a member that begins with HZSPRM, such as HZSPRMMN. When you've defined your check in parmlib, you need to tell IBM Health Checker for z/OS to process the parmlib entry. You can do an MVS MODIFY command similar to the example in Figure 3.

That's it! When your check runs it will examine the RACF protection for the resources that you've specified and raise an exception if any resource allows a general user more access than you specified in your check! Figure 4 shows typical output from an installation-defined check.

So for hassle-free checking use the RACF installation-defined check with IBM Health Checker for z/OS and give your checks some clout. For complete information, see IBM Health Checker for z/OS User's Guide, SA22-7994.

```
ADD CHECK(USER01,MY_INSTALLATION_HEALTH_CHECK)
  CHECKROUTINE(IRRHCR00)
  MESSAGETABLE(IRRHCM00)
  ENTRYCODE(100)
  PARM('USER(USER01)RESOURCELIST(MY_RESOURCE_LIST)')
  DATE(20070425)
  REASON('My sensitive resources')
  GLOBAL
  ACTIVE
  SEVERITY(HIGH)
  INTERVAL(08:00)
```

```
RDEFINE RACFHC MY_RESOURCE_LIST
  ADDMEM(DATASET/PROD.VALUABLE.DATA/ZDR17B/NONE
         DATASET/SEC.FILING.FORMS/NONE
         DATASET/PUBLIC.REPORTS/REGVOL/READ
         MYCLASS/SEC.APPROVAL//NONE)
```

```
Resource List from MY_RESOURCE_LIST

S Resource Name _ Class Vol UACC Warn ID* User
- ----------------        ------   ------  ----   ----  ---- ------
E PROD.VALUABLE.DATA DATASET ZDR17B Read No ****
  SEC.FILING.FORMS DATASET REGVOL
  PUBLIC.REPORTS DATASET REGVOL
  SEC.APPROVAL MYCLASS

* High Severity Exception *

IRRH237E The MY_RESOURCE_CHECK check has found one or more potential errors in the security controls on the installation-specified Resources on this system.

Explanation: The RACF installation resource check has found one or more potential errors with the protection mechanisms on.

System Action: The check continues processing. There is no effect on the system.

Operator Response: Report this problem to the system security
```

```
Figure 1. A RACFHC profile

Figure 2. Registering your check

Figure 3: Refreshing parmlib

Figure 4. Check output
Custom fields for a custom fit
Defining your own fields in USER and GROUP profiles

BY LAURIE WARD AND ERIC ROSENFELD

The RACF database is a repository for security-related information about users, groups, and resources. As with most repositories, each installation has different requirements for the data stored in the entries. Back in the 70s, when leisure suits were in all their glory, RACF gave you the ability to store customized data in RACF profiles. The interface to this data was through low-level programming interfaces that required knowledge of RACF database structure and assembler programming to set or look at the values.

Now it’s time for RACF to don a new suit. With z/OS V1R10, RACF provides the ability for an installation to define and manage fields in user and group profiles with RACF commands. Gone are the days of having to understand the structure of the RACF database in order to associate your own customized data with user and group profiles. You can create your own fields in the RACF database, and tailor the names and attributes of those fields to suit your purposes, using command interfaces.

The fabric for the garment
Before you define a custom field, you need to decide on some attributes for the field. First, choose whether your field will be associated with user or group RACF profiles. Then you’ll need a keyword name, which must be 1-8 characters long, for your field. What type of data will you put in your field? Do you need a character, numeric, hexadecimal, or a flag field, which has a YES or NO value?

Next, choose a maximum field length. Character fields can be up to 1100 characters long. You can also choose a heading that is displayed by the list commands and help text to allow the commands that set the field values to give additional information. For character fields, you get to choose restrictions for which characters are allowed in the first and remaining positions in your field. In addition, you can choose to preserve characters in mixed case, or use the default of upper case. For numeric fields, you can have minimum and maximum values for the field.

Figure 1 shows the attributes you can customize for each of the four types of custom fields.

If these choices for field attributes are not specific enough for your field, you can write a field validation exit to apply additional rules to the custom field data. For example, maybe you want to enforce a naming convention in the EMAIL field for user profiles to ensure that the character ‘@’ appears somewhere in the field. You can write an exit, and use the MVS dynamic exit facility to activate it.
You only need to activate the CFIELD class once, because the CFIELD class will remain active unless it is deactivated by the SETROPTS command.

Finally, you can use the IRRDPI00 command to activate the custom field. You may not be familiar with the IRRDPI00 command because most installations only run this command with a started procedure during system initialization. It can also be run as a TSO command like this:

```
ALLOCATE FILE(SYSUT1)
DATASET('SYS1.SAMPLIB(IRRDPSDS)')
SHR IRRDPI00 UPDATE
```

You must issue the IRRDPI00 command each time you add, update, or delete a CFIELD profile, because the IRRDPI00 command has been updated in z/OS V1R10 to read the CFIELD profiles from the RACF database and extract the custom field definitions from the CFDEF segments. The custom field definitions are then placed into the RACF dynamic parse tables with all the other RACF keywords. Your keyword, the keyword-name portion of the CFIELD profile name, is then ready to be used in the CSDATA segment on the appropriate commands.

To store your installation data, use the ADDUSER, ALTUSER, ADDGROUP, or ALTGROUP command. To display this data, use the LISTUSER or LISTGRP command. The ISPF panels are another way to display and set custom field data. They will display all of the available custom fields for a user or a group on one CSDATA panel, and then you can add, change, or delete the data in the fields.

Stitching the pieces together

Once you have identified the attributes for your custom field, setting it up can be done in a snap. You can use the RDEFINE command to create a profile in the CFIELD class for your new keyword. The name of the profile contains the type of profile in which the field will reside (USER or GROUP), and the keyword name. The format looks like this:

```
profile-type.CSDATA.keyword-name
```

For example, USER.CSDATA.EMAIL or GROUP.CSDATA.LOCATION.

The CFDEF segment on the CFIELD profile will define all the attributes of your new field. Here is the syntax of the CFDEF segment and keywords for the RDEFINE command:

```
RDEFINE CFIELD profile-type.CSDATA.keyword-name
CFDEF ( TYPE( CHAR | FLAG | HEX | NUM) MAXLENGTH(nnn) FIRST( ALPHA | ALPHANUM | ANY | NONATABC | NONATNUM | NUMERIC) OTHER( ALPHA | ALPHANUM | ANY | NONATABC | NONATNUM | NUMERIC) HELP('help-text') MINVALUE(nnn) MAXVALUE(nnn) MIXED ( YES | NO ) LISTHEAD('list-heading-text') )
```

Many of the fields have defaults based on TYPE, so you rarely need to specify every attribute when you define a custom field. There are also ISPF panels available to guide you in defining custom fields.

Once you have created a profile for your keyword, you must activate the CFIELD class with the SETROPTS command:

```
SETROPTS CLASSACT(CFIELD)
```

Figure 1. Field attributes for each custom field type
**A visit to the dressing room**

Let’s button up and take a look at how this works with an example. Suppose you want to have a field in user profiles to store an e-mail address. Figure 2 shows how it might look.

**A finely tailored suit**

We started from the roots of RACF with a skill-intensive method of keeping customized information for users in the database and moved forward to a more administrator-friendly, command-based approach to storing installation-defined data for users and groups. RACF indeed has upgraded its wardrobe with this new support.

When you are ready to give a custom fit to your RACF user and group profiles, see the following Security Server RACF publications for more information:

- The RDEFINE, RALTER, RLIST commands (for information on the CFDEF segment and keywords) and the ADDUSER, ADDGROUP, ALTUSER, ALTGROUP, LISTUSER, LISTGRP commands (for information on the CSDATA segment and keywords) in the z/OS Security Server RACF Command Language Reference, SA22-7687
- The IRRDP100 command details and the custom field validation exit IRRVAF01 in z/OS Security Server RACF System Programmer’s Guide, SA22-7681.

**1. Define the keyword.**

```
RDEFINE CFIELD USER.CSDATA EMAIL
  CFDEF (TYPE(CHAR) MAXLENGTH(50)
    FIRST(ANY) OTHER(ANY) MIXED(YES)
    LISTHEAD(‘USER’S E-MAIL ADDRESS =’)
    HELP(‘E-MAIL ADDRESS’))
```

**2. Activate the CFIELD class.**

```
SETROPTS CLASSACT(CFIELD)
```

**3. Use IRRDP100 command to check and activate the fields.**

```
ALLOCATE FILE(SYSUT1) DATASET(‘SYS1.SAMPLIB(IRRDPDSDS)’) SHR
IRRDPI00 CHECK
IRRDPI00 UPDATE
IRRDPI00 LIST (USER CSDATA)
```

**4. Use the new custom field keyword.**

```
ALTUSER ANDREW CSDATA (EMAIL(‘Andy@anywhere.org’))
```

**5. List the results.**

```
LISTUSER ANDREW CSDATA NORACF
USER=ANDREW
CSDATA INFORMATION
USER’S E-MAIL ADDRESS = Andy@anywhere.org
```

![Figure 2. Custom field example](image)

**DISPOUT will help you out**

Are you looking to parse XML data from within a program? Having problems getting started? Or perhaps you’re looking for some code that processes the output generated by the z/OS XML System Services parser?

Well, have you checked the z/OS XML System Services Web site lately? Have you seen that there is now a free downloadable REXX™ program that displays the XML output buffer in a more easily-readable form? It’s called DISPOUT, and you can get it from [ibm.com/servers/eserver/zseries/zos/xml/download](http://ibm.com/servers/eserver/zseries/zos/xml/download).

DISPOUT reads in the output buffer data from a dataset, processing and displaying each token one at a time. The DISPOUT download package also includes a sample JCL that executes DISPOUT. By submitting this job to run the DISPOUT program, you’ll be able to display the generated output using the System Display and Search Facility (SDSF). You may find this handy, because the output from DISPOUT can be lengthy.

DISPOUT may also inspire you to create your own programs that process XML documents using the z/OS XML System Services parser. Enjoy!
Over the years, the stories and myths have filtered down from administrator to administrator, but the question remains: When it comes to obtaining digital certificates, what makes the most sense for your business?

Before buying digital certificates, read this article. You might find that z/OS Cryptographic Services PKI Services can help you streamline your process and save money.

Two schools of thought

In a client-server environment, digital certificates are used to verify the identity of the client and server, and protect the information flowing in the network. We covered the basic concepts of digital certificate technology in the article “Security alert: Do you want to proceed?” in z/OS Hot Topics Newsletter Issue 14, February 2006, GA22-7501-10.

The decision of where to obtain digital certificates is often based on either of two beliefs, one being that it is better to create your own certificates, and the other that it is better to have certificates created for you.

Needing digital certificates for secure communications, many businesses have largely been conditioned to rely on an established third-party provider, called a commercial certificate authority or CA — even if the certificates are to be used only for communications between internal servers. This practice is based on the assumption that a commercial certificate is just more secure.

In contrast, other businesses take a different approach to obtaining digital certificates. Seeing no difference in functionality between a commercial certificate and a locally generated certificate, and seeking to avoid the expense of purchasing from a commercial certificate authority, these businesses use certificate generation tools to create their own certificates for their applications.

The truth about digital certificates

In reality, the digital certificate format is platform independent. Certificates are created under a common standard set forth by the Internet Engineering Task Force (IETF) – RFC3280.

Although it is true that a “homemade” certificate is functionally equivalent to a commercial certificate and can provide the same amount of security, weaknesses can exist in the management of these locally generated certificates.

You will find that most businesses generate a self-signed certificate if they are not buying from a commercial CA. Their goal is simply to supply a certificate to make the application work. The recipients are instructed to accept the certificates blindly, which can result in many of these certificates floating around.

One problem with this approach is that it creates a false sense of trust. Even though your business chooses to be secure, the trust process of interrogating and installing a large number of self-signed certificates can be very time consuming.

The trust of a certificate is based on the trust of its issuer. If you choose to trust the issuer’s certificate after ensuring that it comes from a trusted source, you install the certificate in your key store. What is the issuer’s certificate in a self-signed scenario? It is the self-signed certificate itself. Thus, trusting a hundred self-signed certificates means investigating and installing a hundred certificates!

When it comes to obtaining digital certificates, what makes the most sense for your business?
Security that’s close to home
Wouldn’t it be nice if your business could:
• Avoid purchasing digital certificates?
• Control the issuance of digital certificates?

With z/OS PKI Services, you can do both. PKI Services provides a certificate authority for the z/OS environment and enables you to issue and administer digital certificates, so that you do not have to purchase them from an external certificate authority. By setting up z/OS PKI Services as an internal CA, your installation can issue all the certificates it needs for internal use, that is, certificates for its networks, servers, and e-mail.

With certificates created through z/OS PKI Services:
• You can have certificates distributed right in the requestor’s browser
• Your administrator can query which certificates are issued
• The requestor can be notified when the certificate is going to expire.

Better yet, you can set-up the ‘autorenew’ feature to have the renewed certificate mailed to the requestor before the old one expires. This approach has the convenience of a self-signed certificate. It provides better management, yet it carries no additional charge because it is included as a base feature in z/OS!

Certificates issued by an internal CA can be used by an external site, too. It is not true that a server certificate for external use has to be issued by a commercial CA. While it might be more convenient to buy a commercial certificate for an external site, this is not the only valid way of providing security for communications.

The convenience of a commercial certificate is that the visitors to the external site do not need to install its CA certificate. All the well-known CA certificates come pre-installed in the browser. To use a certificate issued by an internal CA, you just need to distribute your CA certificate to the site visitors and have them install it. This is a one-time effort. After the internal CA certificate is installed, the certificate works the same way a commercial certificate does.

Find out more
If you are concerned about the recurring costs of digital certificates, creating your own through PKI Services can be a less costly alternative.

To find out more about PKI Services, see z/OS Cryptographic Services PKI Services Guide and Reference, SA22-7693, or go to ibm.com/servers/eserver/zseries/zos/pki.
Tales from the crypto

Decrypting TCP/IP hardware cryptography

BY CHRIS MEYER

With new regulatory standards mandating secure network communications, more and more z/OS installations are turning to network security protocols to provide endpoint authentication and data encryption to meet their compliance goals. However, the associated cryptographic operations carry a significant processing cost. As a result, the same installations are looking to the rich set of hardware facilities in IBM System z to ease the burden as much as possible. But when you combine varied facilities like CP-assisted cryptographic instructions, cryptographic coprocessors and accelerators, and even zIIP specialty processors with complex protocols, things can get confusing in a hurry.

z/OS supports two industry standard protocols for securing IP network communications, as follows:

- Secure Sockets Layer (SSL), which the Internet Engineering Task Force (IETF) has standardized as Transport Layer Security (TLS). We use the terms SSL and TLS interchangeably in this article. You can either code your applications to use TLS directly, or you can let z/OS Communications Server take care of it for you through its Application Transparent TLS (AT-TLS) support.

- IP Security (IPsec), with which the Internet Key Exchange (IKE) protocol is closely associated.

In this article, we hope to clear up some of the crypto confusion by explaining how one or more of the hardware facilities can assist the different types of cryptographic operations that each protocol performs.

In addition to these cryptographic facilities, a System z10 or System z9 Integrated Information Processor (zIIP) specialty engine is used to direct some CPU-intensive TCP/IP cryptographic operations away from general CPs.

On the software side, there are three key components:

- The System SSL component of the Cryptographic Services base element of z/OS implements the SSL and TLS protocols for z/OS and also provides a number of cryptographic services to other components, including IKED.

- The z/OS Communications Server implements the IPsec protocol and AT-TLS services in the TCP/IP stack and also implements the IKE daemon (IKED).

- System SSL, IPsec, and IKED all use the z/OS Cryptographic Services Integrated Cryptographic Service Facility (ICSF) if it is available. ICSF provides callable services for a wide variety of cryptographic operations and is also the only way that cryptographic coprocessors and accelerators are accessed (CPACF can be accessed directly or through ICSF).

The existence of CPACF and the coprocessors or accelerators is checked at runtime. If they’re not available, TCP/IP and System SSL use their own software implementations of the algorithms (if available).

SSL/TLS

TLS establishes an end-to-end encrypted session over TCP between two applications. When a TLS session is established, an SSL handshake takes place between the endpoints. After the SSL...
handshake is complete, the secure session is established and any data that flows over the connection is protected according to the chosen cipher suite. Figure 1 shows the basics of TLS session establishment.

In most environments, applications must be explicitly coded to use TLS protection. On z/OS, this means using the System SSL APIs. System SSL provides APIs to establish a secure connection (which performs the SSL handshake) as well as APIs to protect to the data that flows over the TLS session. The handshake, as well as the subsequent data protection, is performed within the context of the TLS-enabled application.

Unlike most environments, z/OS Communications Server provides a way to apply TLS protection to TCP sessions without requiring changes to the application program. This capability is appropriately called Application Transparent TLS. With AT-TLS, you create policies to describe the traffic to protect along with the type of TLS protection to apply. When those TCP sessions are established, the TCP/IP stack performs the TLS operations on behalf of the application according to the policies. Although application changes are not needed, APIs are available for applications that require TLS awareness or control. Under the covers, AT-TLS is largely a wrapper for System SSL. This means that the cryptographic characteristics of AT-TLS are identical to that of System SSL itself. The key difference to remember

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPACF only</th>
<th>CPACF + CEX2C/CEX2A</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA signature generation</td>
<td>In software</td>
<td>In coprocessor mode only. Otherwise in software (accelerator does not support this operation).</td>
</tr>
<tr>
<td>RSA signature verification</td>
<td>In software</td>
<td>In coprocessor/accelerator</td>
</tr>
<tr>
<td>PKA encrypt/decrypt for handshake</td>
<td>In software</td>
<td>In coprocessor/accelerator</td>
</tr>
<tr>
<td>SHA-1 digest generation</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>SHA-224 digest generation</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>SHA-256 digest generation</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>SHA-384 digest generation</td>
<td>In software on z9™, CPACF in z10 EC</td>
<td></td>
</tr>
<tr>
<td>SHA-512 digest generation</td>
<td>In software on z9, CPACF in z10 EC</td>
<td></td>
</tr>
<tr>
<td>DES (56-bit) encrypt/decrypt</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>3DES (168-bit) encrypt/decrypt</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>AES-128 encrypt/decrypt</td>
<td>CPACF</td>
<td></td>
</tr>
<tr>
<td>AES-256 encrypt/decrypt</td>
<td>In software on z9, CPACF in z10 EC</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. System SSL (and AT-TLS) hardware crypto usage

Figure 1. Establishing an SSL/TLS session
IKE peers negotiate an IKE (“phase 1”) tunnel (one bidirectional SA) over an unprotected UDP socket.

IKE peers negotiate an IPSec (“phase 2”) tunnel (two unidirectional SAs) under protection of the IKE tunnel.

Data flows through IPSec tunnel using the Authentication Header (AH) and/or Encapsulating Security Payload (ESP) protocol.

**Figure 2. Creating an IPsec security association**

IKE peers negotiate an IKE (“phase 1”) tunnel (one bidirectional SA) over an unprotected UDP socket.

IKE peers negotiate an IPSec (“phase 2”) tunnel (two unidirectional SAs) under protection of the IKE tunnel.

Data flows through IPSec tunnel using the Authentication Header (AH) and/or Encapsulating Security Payload (ESP) protocol.

**Table 2. IKE hardware crypto usage**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPACF only</th>
<th>CPACF + CEX2C/CEX2A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffie-Hellman operations</td>
<td>In software via SystemSSL</td>
<td>In software via SystemSSL</td>
</tr>
<tr>
<td>RSA signature generation (clear key only)</td>
<td>In software</td>
<td>In Coprocessor if available, otherwise, in software</td>
</tr>
<tr>
<td>RSA signature verification</td>
<td>In software</td>
<td>In Coprocessor/Accelerator</td>
</tr>
<tr>
<td>DES (56-bit)</td>
<td>In software</td>
<td>In software</td>
</tr>
<tr>
<td>3DES (168-bit)</td>
<td>In software</td>
<td>In software</td>
</tr>
<tr>
<td>AES-128</td>
<td>In CPACF via ICSF, otherwise not supported</td>
<td>In CPACF via ICSF, otherwise not supported</td>
</tr>
<tr>
<td>SHA-1</td>
<td>In software</td>
<td>In software</td>
</tr>
<tr>
<td>MD5</td>
<td>In software</td>
<td>In software</td>
</tr>
</tbody>
</table>

IKE daemon

Table 2 shows the z/OS IKE daemon (IKED) hardware usage. IKED only uses CPACF (if available) for AES encryption/decryption. Unlike System SSL, IKED accesses CPACF through calls to ICSF. This means that IKED relies completely on ICSF for AES support. If CPACF or ICSF are not available, IKED does not allow AES encryption. All other symmetric encryption, decryption, and hashing operations are performed in IKED software algorithms.
IKED calls System SSL for RSA digital signature and Diffie-Hellman operations. As a result, RSA digital signature operations are offloaded as described above for System SSL. System SSL always performs Diffie-Hellman operations in software.

Unlike System SSL, adding extra coprocessors or accelerators will not affect the performance of IKED.

IPsec (AH and ESP)
As shown in Table 3, the TCP/IP stack, like IKED, calls ICSF to use CPACF for DES, 3DES and AES encryption, decryption and SHA hashing. If either ICSF or CPACF are unavailable, the stack uses its own software algorithms for DES, 3DES and SHA (software-only AES is not provided). The stack always performs its own MD5 hashing in software and it never uses a CryptoExpress2.

The TCP/IP stack also makes use of zIIP specialty engines. Starting with V1R8 (PTF UK27062 and UK27063), most IPsec-related SRB-based processing in the stack, including AH and ESP cryptographic operations, can be offloaded to zIIP engines to reduce general processor utilization. Because AH does not support data encryption, the biggest benefit of the zIIP offload is realized for ESP security associations. When the stack code executes on a zIIP, it can use CPACF in exactly the same manner as described previously.

Summary
z/OS provides a strong set of industry standard protocols for protecting TCP/IP network communications in System SSL, AT-TLS and IPsec with its related IKE daemon. In addition, the System z platform provides a rich set of hardware features that can be used to improve cryptographic performance as well as to reduce cryptographic CPU utilization. The result is a winning combination for protecting your mainframe communications and complying with today’s growing list of industry and government standards for data privacy.

Learn more!
IBM publications provide more information about the following topics:
- System SSL: See z/OS Cryptographic Services System Secure Sockets Layer Programming, SC24-5901.
- ICSF: See z/OS Cryptographic Services Integrated Cryptographic Services Facility Overview, SA22-7519.

Acknowledgements
The author would like to thank the following people for their help and expertise in preparing this article:
- Alyson Comer, z/OS System SSL development at IBM Endicott, NY
- Scott Moonen, z/OS TCP/IP development at IBM Raleigh, NC
- Mike Fitzpatrick, z/OS TCP/IP performance design at IBM Raleigh, NC.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPACF (regardless of Coprocessor/Accelerator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES (56-bit)</td>
<td>In CPACF via ICSF</td>
</tr>
<tr>
<td>3DES (168-bit)</td>
<td>In CPACF via ICSF</td>
</tr>
<tr>
<td>AES-128 (only via ICSF)</td>
<td>In CPACF via ICSF</td>
</tr>
<tr>
<td>SHA-1</td>
<td>In CPACF via ICSF</td>
</tr>
<tr>
<td>MD5</td>
<td>In software</td>
</tr>
</tbody>
</table>

Table 3. IPsec (AH, ESP) hardware crypto usage
Open Sesame was the secret password phrase overheard by Ali Baba that gave him access to the 40 thieves’ treasure. The problem of protecting treasures has been around for centuries. Today, more than ever, we need to keep data safe and ensure proper authorization. Fortunately, z/OS UNIX System Services has a set of functions that provide user ID authentication and verification, meeting your need of providing increased system security and giving you more flexibility with your security choices.

Prior to z/OS V1R10, these services allowed specifying a password or a PassTicket for user ID validation. Although a PassTicket was allowed on these services, the PassTicket value wasn’t very usable because there was no easy, consistent way to associate an application identifier (APPLID) with the PassTicket. A C/C++ application would have to update the z/OS UNIX BPXYTHLI control block (an assembler API) with a specific APPLID value prior to issuing the service; otherwise, they were limited to PassTickets generated with either an APPLID value of “OMVSAPPL” or the system-defined default.

For z/OS V1R10, password phrases are now supported on __passwrd(), __login(), and pthread_security_np() calls. Three new services allow an installation to specify an APPLID value in conjunction with a PassTicket. The services are equivalent to the existing services listed above, with the added feature that an APPLID can be specified.

There’s more! Your installation now has the option of using password phrases in addition to passwords and PassTickets. Furthermore, you can use PassTickets generated with a specific APPLID with all z/OS UNIX security related services. This allows a more granular authentication mechanism for various applications running on a single z/OS image.

Why use a password phrase?
Having a longer password inhibits hackers from guessing at or writing algorithms to determine the password. In addition, for most users, it is a lot easier to remember a phrase, then a password. For example, a password phrase like “z/OS Hot Topics is great” is a lot easier to remember than having to create a password subject to very stringent rules.

Why use a PassTicket?
PassTickets are useful for single-use sign-on, where access to a system is limited or temporary. PassTickets are also great for accessing systems across a network, using FTP, or from batch jobs. This allows access to the host systems without the hazard of a password or password phrase appearing in clear text. PassTickets are inherently more secure than a password or password phrase for these cases. Neither PassTickets nor password phrases are a replacement of passwords. However, they do allow an installation the flexibility of managing systems in a more secure way.

How do passwords, password phrases and PassTickets differ?
Passwords are always 1-8 characters. They are subject to the security product rules for content and z/OS UNIX can uppercase by prior to calling the security product. Password phrases are always greater than eight characters. The security product can set additional limitations. For example, RACF allows password phrases from 14-100 characters, but RACF also provides an exit that can change the limit to 9-100. z/OS UNIX will never modify or uppercase password phrases prior to calling the security product.

The security product might set further restrictions. For example, RACF does not allow the user ID to be contained within the password phrase.

PassTickets are always eight characters and are created using a cryptographic key, a time stamp, a user ID, and an APPLID value. They have a short life span (just 10 minutes, plus or minus its creation time) and can only be used to authenticate once.

Neither PassTickets nor password phrases are replacements for passwords. They do however allow an installation the flexibility to manage their systems in a more secure way.
Don’t keep me down!

Using AutoIPL to minimize z/OS downtime

BY WALT OTTO

Maybe you’ve been there — it’s 2:00 in the morning, and z/OS has just gone into a disabled wait. Rather than just re-IPL, you want to be a good citizen and take a stand-alone dump, so IBM can have a shot at finding the problem. But all the phones are ringing, and you need to do something fast.

The conversation goes something like this (usually getting faster and louder!)

“Hurry up and take the dump, so we can get back up!”
“I don’t remember how.”
“Well, look in the book!”
“Where’s the book?”
“What do you mean, ‘Where’s the book?’ What did you do with it?”
“I thought you had it!”

So either you thrash around and finally figure out how to take the dump, or you just re-IPL and think, “Maybe next time.”

Relax and automate!

But now in z/OS V1R10 there’s a way to automate the taking of the stand-alone dump and the re-IPL of z/OS: AutoIPL.

Establishing an AutoIPL policy

Coding an AUTOIPL statement in DIAGxx (with some info other than NONE), and then causing that DIAGxx member to be read, establishes an AutoIPL policy. As of this writing, that policy won’t apply to any restartable wait states, but it will apply to all but a few non-restartable wait states. Those non-restartable wait states are in the hard-coded wait state action table (WSAT), with flags indicating whether a stand-alone dump or a re-IPL is appropriate for each. (For example, wait states associated with Basic HyperSwap™ should not result in a re-IPL of z/OS.) Keep in mind that the WSAT is in the read-only nucleus, and it is not an intended interface.

After you establish an AutoIPL policy, you can display it by issuing D DIAG. The system displays the SADMP and z/OS devices and load parameters.

If it’s not what you intended, you can edit the DIAGxx member and issue SET DIAG=xx again to replace the previous policy.

Program-directed IPL facility

The AutoIPL function is made possible by the program-directed IPL facility, and, of course, it requires that that facility be installed on your machine. All IBM System z10 machines and later IBM System z9 machines have the facility (either standard or as an option), while older machines do not. (If you try to establish an AutoIPL policy on a machine that doesn’t have the facility, you will get an environment error message.)

The processing works like this — when some z/OS component encounters a serious problem and decides that it’s time to load a disabled wait state, it invokes the Loadwait component to make it happen. If there is an AutoIPL policy in place, and if the WSAT entries don’t negate it, Loadwait loads a wait state on n-1 CPUs, but instead of loading a wait state on the last CPU, it sets up to load either SADMP or z/OS. If it’s to be SADMP, and if a re-IPL of z/OS is also indicated, Loadwait places the z/OS device and load parameter in a storage location where SADMP can find them and sets a flag to tell SADMP to IPL z/OS at the conclusion of the dump. Then Loadwait invokes the program-directed IPL firmware function to initiate the load. If there is no active AutoIPL policy, Loadwait simply loads the wait state as it has always done.

You can also invoke AutoIPL manually, using the VARY XCF command. For instance, after you have established a policy, enter the following command:

VARY XCF,RIMSKY,OFFLINE,REIPL

Getting started

You start by coding an AUTOIPL statement in a DIAGxx parmlib member, specifying the device on which the stand-alone dump program (SADMP) resides and the load parameter that z/OS will use to load SADMP, and the device and load parameter that SADMP or z/OS will use to re-IPL z/OS. Then either point to that DIAGxx member so that the system can read it during IPL, or cause the system to read it immediately by issuing a SET DIAG=xx command. Of course, you still need to take care to generate the SADMP properly, ensure that there is enough output space, and so forth, but you can do that at your convenience, before the phones start ringing.

The syntax of the AUTOIPL statement is pretty straightforward:

AUTOIPL SADMP(sadmp info) MVS(mvs info)

Usually, info is a device and a load parameter, but it can be NONE, in which case the system will not perform that action. And you can specify MVS (LAST) to indicate to use the current z/OS device and load parameter. If you don’t code an AUTOIPL statement, neither action will take place.

Don’t keep me down!
In this example, XCF varies the system named RIMSKY out of the sysplex, and requests a wait state that causes Loadwait to initiate a re-IPL of z/OS. Similarly, the following command varies KORSAKOV out of the sysplex and initiates a SADMP, to be followed by a re-IPL of z/OS:

VARY XCF,KORSAKOV,OFFLINE,SADMP,REIPL

Several XCF messages have been enhanced to reflect the new command options.

Still another way to invoke AutoIPL, after you have established a policy, is through the SADMP load parameter. The seventh byte of the load parameter contains flag bits, and if SADMP finds the second bit on it performs an IPL of z/OS at the conclusion of the dump. (This is meant to hasten a re-IPL of z/OS when a SADMP is taken manually.)

New messages
SADMP has two new messages:
• AMD114I tells the operator that SADMP was loaded through AutoIPL.
• AMD113I tells the operator that SADMP is about to IPL z/OS. It also indicates whether the IPL of z/OS was requested by z/OS or by the operator (through the new load parameter flag).

A few notes:
• AutoIPL is not appropriate in a Geographically Dispersed Parallel Sysplex™ (GDPS®) environment because GDPS could cause the system to be reset while a SADMP or re-IPL is in progress.
• Do not define either the SADMP or the z/OS load device as a secondary device in a Metro Mirror pair.

Calm yourself with AutoIPL
So root around in your desk and find the SADMP book, get a cup of coffee, and set up an AutoIPL policy. Then, the next time a wait state happens, you won’t be having that frantic 2:00 AM conversation! Here’s where to find more information:
• For general information about AutoIPL as well as the current entries in the wait state action table, see z/OS MVS Planning: Operations, SA22-7601.
• For the syntax of the AUTOIPL statement, see z/OS MVS Initialization and Tuning Reference, SA22-7592.
• For the full syntax of the VARY XCF command, see z/OS MVS System Commands, SA22-7627.
• For the new wait state reason codes that XCF uses to invoke AutoIPL, see z/OS MVS System Codes, SA22-7626.
• For updated SADMP messages, virtual storage management (VSM) messages, and XCF messages, see z/OS MVS System Messages, Volume 1, SA22-7631, Volume 9, SA22-7639, and Volume 10, SA22-7640.
• For the new option within the SADMP load parameter, see z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589.
Contests, contests everywhere

BY KATHY PFEIFFER AND MIKE TODD

Sponsored by the IBM Academic Initiative program for System z, the twelve-person team that developed the first-ever IBM Student Mainframe Contest in the fall semester of 2005 knew that it had a good plan to draw students to the mainframe platform. But it didn’t know what kind of reaction to expect once the contest went live. The reaction: enthusiasm — and not just in the U.S. and Canada, where the contest drew its first 750 participants in 2005, but in the twelve countries and over 1000 schools worldwide that have participated in IBM Student Mainframe Contests in the three years since.

“The results were amazing,” reported José Vilela of the IBM Academic Initiative for Brazil and Latin America, after his team ran Brazil’s first mainframe contest in the spring of 2007. “We had more than two thousand enrollments in nine days. After the contest ended, IBM hired thirty-six contestants to work in several entry-level technical mainframe positions.”

Professors have also been impressed with the skills and job opportunities that their students have picked up through the contests. Henrique Ribeiro, a computer science professor from Universidade Bandeirante de São Paulo, said, “After my students finished the Brazil Mainframe Contest, six of them were hired for all sorts of Brazilian companies to work with mainframe technology. Other students keep searching for information on the platform to learn more about it. It continues to be rewarding for them long after the contest has ended.”

While IBM teams in various geographies adapt the contest to suit the tastes of local students and employers, the basic premise remains the same: offering a hands-on competition that challenges students to complete a series of increasingly difficult technical tasks on a mainframe system. Students log into that system remotely from their homes, dorm rooms, and computer labs. As the challenges get more difficult, the prizes get bigger. The contests are all advertised as “No Experience Necessary,” so the earlier tasks arm students with the knowledge they’ll need to complete the harder tasks later in the contest.

As of this writing, IBM teams have run Student Mainframe Contests in the United Kingdom, Belgium, the Netherlands, Luxembourg, Sweden, Norway, Denmark, Finland, Thailand, Brazil, the United States, and Canada. All told, 8180 students worldwide have competed, with many of them landing jobs as a result.

“My participation and successful completion of Part One of the IBM Student Mainframe Contest helped to get me a job at a large bank working as a Junior Systems Support Analyst,” said Elizabeth Bell, a student at Georgian College in Ontario, Canada. “This position has never been offered to a student before, and I never would have been able to secure it without all that I have learned so far in the contest.”
Spreading the news
As the good idea keeps spreading, the list of countries offering mainframe contests continues to expand. “We are running a mainframe contest in Australia as part of the ongoing vitality hiring program of the Australia Development Lab,” said Troy Astle of Software Group. “As well as providing an educational and entertaining experience for students, the contest will expose young software engineers to exciting System z career opportunities.”

The contest continues to evolve as each country refines the formula. In the fall semester of 2007, the contest opened to high school students in the U.S. and Canada for the first time. The result: 57 high schools were represented among the 325 schools with registered contestants. With 1750 students competing, the overall winner of the contest was Sushen Patel, a junior from Highland Park High School in Dallas, Texas.

Everyone’s a winner
When the first contest launched in 2005, the goal of the IBM Academic Initiative System z contest team was to provide a fun, enjoyable learning experience for students who had no prior knowledge or experience with an IBM mainframe. Not only have students embraced the contest as “cool,” as evidenced by the many repeat contestants, they’ve come to realize that mainframes process much of the world’s most strategic information and applications and are very much alive and well. For future information about IBM Academic Initiative student contests, visit [ibm.com/university/students].
In IBM Announcement 107-190, dated April 18, 2007, IBM made a statement of direction (SOD) regarding future enhanced support for XML on the z/OS platform. The SOD focused on the ability to redirect XML parsing performed by z/OS XML System Services (z/OS XML) to specialty engines, as well as leveraging of z/OS XML primitives by other higher-level XML processors, and improved capabilities in z/OS XML. Specifically, the SOD indicated that:

1. IBM intends to enable the z/OS XML to take advantage of the System z Application Assist Processor (zAAP).
2. IBM intends to enable the z/OS XML to take full advantage of the System z9 Integrated Information Processor (zIIP). (In other words, 100% zIIP offload!)
3. IBM intends to extend and expand on the use of z/OS XML System Services (enabled for zAAP specialty processors) as the basis for future enhancements:
   - IBM intends to add validating parsing to the z/OS XML component.
   - IBM intends to enhance the XML Toolkit for z/OS so that eligible workloads can exploit the z/OS XML component.

Most of this support is already available.

The first item, the ability to redirect parsing performed by z/OS XML to a zAAP processor when invoked in task control block (TCB) mode (often referred to as task mode), was included in z/OS V1R9 and has been rolled back to z/OS V1R7.

The second item, the ability to redirect z/OS XML to a zIIP when invoked in service request block (SRB) mode, has been announced in the z/OS V1R10 preview to be available as part of z/OS V1R10. We say this is a 100% zIIP offload to distinguish this support from redirection of z/OS XML from an application that is itself redirected to a zIIP (for example, certain DB2 workloads that are defined as zIIP-eligible). Prior to this support, if z/OS XML was being used by DB2 workload that was zIIP-eligible, the z/OS XML processing was redirected at the same percentage as DB2 itself; the z/OS XML processing just “came along for the ride” because DB2 was invoking its services. With this new support, any use of the z/OS XML parser from an enclave SRB invoker is fully redirected, regardless of whether the invoker itself is zIIP-eligible. This support is being rolled back to z/OS V1R8.

The final item in the SOD described both a significant functional improvement in z/OS XML in regard to schema validation support, as well as exploitation of it by the Apache Xerces-based C++ XML parser included in the XML Toolkit for z/OS. As indicated in the z/OS V1R10 preview announcement, z/OS XML is adding the ability to perform validating parsing in addition to its current support for non-validating parsing. This means that you will be able to have z/OS XML check whether a given XML document adheres to a given World Wide Web Consortium (W3C) schema definition when it is parsing the document. Like the existing non-validating parser, processing performed by this validating parser can be directed to a zAAP or zIIP processor (depending on whether the invoking application is running in task or enclave SRB mode). Historically, validating parsing has been an expensive proposition, to the point where many customers decided that the cost was just too great for the benefit derived. Instead, they often rely on pre-production testing, and perhaps performing validation within their application code, rather than relying on an XML parser to do the validation. With this new support, validating parsing has become a more viable option.

The other aspect of this final item concerns the XML Toolkit for z/OS. Through the support provided for XML Toolkit for z/OS V1R9 by PTF UA40707 and PTF UA40708, the Apache Xerces-based XML4C parser can use z/OS XML.
as its underlying parsing technology for non-validating parsing. This means that, with very minor changes, many applications can continue to use the current SAX2 and DOM interface styles provided by the XML Toolkit parser, but actually have that XML4C parser use z/OS XML as part of its processing. This support has the benefit of allowing the work performed by z/OS XML to be redirected to a zAAP engine if one is available, and is designed to provide a significant raw performance improvement in the overall operation.

This additional support is provided by a new set of z/OS-specific C++ classes that are similarly named and match certain existing classes in the parser. By making these minor name changes and rebuilding the application, you can enable the application to be partially executed on a zAAP specialty engine. Note that although most of the function in the XML4C parser is accommodated in this new support, there are some restrictions (such as lack of support for entity resolvers and the SAX2XMLFilter class). The XML Toolkit for z/OS User’s Guide, SA22-7932, describes these restrictions in detail.

Additionally, thanks to support in Enterprise COBOL V4.1 (5655-S71) for a new XMLPARSE compiler option, the COBOL built-in XML parser can use z/OS XML for XML processing. This lets the COBOL XML parser take advantage of zAAP specialty processors, and has enabled other functional improvements (such as support for parsing XML documents that use XML namespaces). For more information, refer to Software Announcement 207-339, dated December 11, 2007.

As you can see, IBM is making good on its promise of improved processing of XML data on z/OS!

Good sources of information about the support described here are available from the following Web sites:

- XML Toolkit for z/OS: [ibm.com/servers/eserver/zseries/software/xml](ibm.com/servers/eserver/zseries/software/xml)
- z/OS XML System Services: [ibm.com/servers/eserver/zseries/zos/xml/](ibm.com/servers/eserver/zseries/zos/xml/)
- Enterprise COBOL for z/OS: [ibm.com/software/awdtools/cobol/zos/](ibm.com/software/awdtools/cobol/zos/)

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-4331-13) are separately orderable.
A good way to get bad news

Clicking away failures with IMS V10 abend search and notification

BY DARIO D’ANGELO, KIN LAU, AND ALAN SMITH

IMS Version 10 introduces a new function that helps you understand IMS abends more quickly and makes problem source identification easier. The abend search and notification function (ASN) captures relevant, symptom-specific information at the time of an IMS system failure. It then automatically constructs an e-mail containing this information and sends this notification to a set of recipients defined at installation time.

Specifically, ASN provides the following advantages:

• Automates the immediate notification of system programmers and other designated recipients in the event of a system failure
• Provides direct access to the abend description and debugging procedure (when available)
• Facilitates problem source identification with hyperlinks pointing to documentation in online references and databases of known problems with the same or similar symptoms and, potentially, even the solution of the problem experienced
• Provides access to necessary resources (such as manuals, diagnostic procedures, and similar items) to reduce the “know-how” needed to solve a problem.

What is the goal?

If the abend has been previously reported, the fix is just a single click away. This concept aligns with the IBM direction to make all serviceability-related information available to customers seamlessly, anytime, and anywhere.

Beyond its notification capability and references to information such as abend description, abend diagnosis procedures, and previously reported problems, e-mail that is generated by ASN (as shown in Figure 1) also contains hyperlinks to the following types of information:

• APARs, problem management records (PMRs), and technotes matching the symptoms of the problem experienced
• The most recent preventive service planning (PSP) bucket
• A hyperlink that can be used to order fixes (PTFs) online
• Information for contacting technical support.

Further, to easily identify the environment where the failure occurred, the following information is captured: IMS ID, IMSplex name (if available), time of ASN invocation, abend code, and associated return code (when meaningful), module name involved in the abend (if available), and the latest maintenance level associated with the failing module.

Why IMS ASN?

If we consider that over 85 percent of all PMRs are known or non-defect oriented problems (that is, the problem resolution did not require a software fix), the reason for implementing this function becomes clear.

Although ASN is not a “silver bullet” for all IMS failures, it can provide tremendous benefit by helping you quickly identify and reliably diagnose the causes of the experienced failure and, therefore, the solution to the failure.

Figure 1. Example of an IMS ASN e-mail notification
The importance of efficiently recognizing and solving an IMS problem is obvious when you consider the amount of critical data that IMS manages. Here are some numbers that can help to quantify the importance of this issue:

- For one IMS customer, over $3 trillion in transactions are processed \( \text{per day} \).
- Over 50 billion transactions run through IMS \( \text{per day} \).
- IMS serves around 200 million users \( \text{per day} \).
- IMS manages over 15 billion GB of production data.
- For one IMS customer, nearly 100 million transactions are processed on a single IMS system \( \text{per day} \).

In these high-workload environments, the cost of low reliability and excessive downtime to wait for problem resolution is often intolerable. Building on the known value of IMS as a high-availability system, the ASN function further reduces your downtime by providing automatic notification, direct references to research materials, and quick discovery of previously reported system failures.

**What else?**

To take full advantage of IMS ASN functions, we have merged and restructured the message and codes and diagnostic manuals to provide not only a generic understanding of why the failure occurred, but also to provide instructions that can be used to perform some basic problem source identification.

Additionally, you can customize the IMS ASN e-mails to fit the needs and requirements of any environment. Using an ISPF-driven interface, you can modify every aspect of the e-mail content, from changing the words and text to adding more online references as hyperlinks, from specifying any number of recipients to making the graphics look and feel the way you prefer. However, customization is not required, because IBM provides an initial sample of e-mail text that already contains hyperlinks to various online references.

If you consider e-mail to be too slow a means of notification, IMS ASN includes the option to send messages to portable devices using short message service (SMS). Due to concerns about the screen size of mobile devices, the SMS message does not contain hyperlinks to the various online resources.

---

In addition to the abend-driven initiation of e-mails, IMS ASN also provides an ISPF interface for sending e-mails on demand. From the ISPF panel, you can:

- Enter the criteria for selecting the hyperlinks to online references
- Specify a different set of recipients to receive the e-mail notification.

**Conclusion**

With ever-increasing system workloads, the availability and reliability of the system become more critical. And as systems become increasingly complex, the need to find new ways of reducing the amount of time necessary to identify and solve a problem is also vital.

IMS ASN introduces an autonomic mechanism that instantaneously notifies you of critical failures and helps you to understand the abend cause. More importantly, it shortens system downtime in case of failure by providing a quick and reliable way to find known and available solutions.
It’s a well-known fact that z/OS delivers designs with very high RAS characteristics different from other platforms. They are intended to help z/OS systems and applications stay available for weeks and months at a time, ensure reliable operation while detecting and recovering from system errors, and capture diagnostic data when necessary.

The “S” in RAS includes all of the serviceability characteristics that make it possible to determine the cause of a problem that affects the operation of z/OS or a major subsystem (like DB2, IMS, or CICS®). As part of z/OS first-failure data capture, storage dumps represent the status of system, subsystem, and application software, triggered automatically by a program’s recovery, or upon request through operator commands.

The z/OS flight recorder
A key component of z/OS diagnostic information is its invaluable “flight recorder” traces, which are maintained by the system and included in storage dumps. In particular, the system trace captures the flow of the system through low-level supervisor instructions, while the master trace maintains a brief snapshot of system messages that occurred shortly before a system dump. Furthermore, tools are available to format and analyze the resulting z/OS logs and dumps to help a user understand functions that were running, and the sequence of events that led to the point of error.

System trace enhancements allow the system to maintain larger system traces, which allows the resulting trace to be more useful for diagnosing large configuration problems. In z/OS V1R9, the system trace buffer size can increase to permit the capture of additional data by faster System z engines. z/OS has the ability to allocate more than 999 KB system trace buffers per CPU. The actual size that you choose depends on the amount of real memory that you wish to dedicate to the system trace and the number of CPUs supporting the z/OS image, although we recommend that you try modest increases over earlier releases.

In z/OS V1R10, the system trace buffers move above the bar, allowing the increased size buffers to no longer impact fixed storage below 2 GB. In addition, the default trace buffer size per processor increases from 256 KB to 1 MB byte. The TRACE command extends to allow the specification of larger buffer sizes:

```
TRACE ST,nnnK or mmmM or nG
```
Managing large stand-alone dumps

While most z/OS dumps are non-disruptive in nature (that is, they usually don't affect the operation of the system), the stand-alone dump that the installation takes when an entire z/OS LPAR is no longer operational is very disruptive, and results in a very large dump data set. The size of a stand-alone dump grows directly with increased amounts of real storage required for system component and application growth.

Taking a stand-alone dump is not a common event, yet planning for the dumps (which can involve several DASD volumes), initiating the dumps when needed, and post-processing them continue to be a challenging exercise. Releases earlier than V1R10 supported the ability to define a multiple-volume dump data set that simulates "striping," writing blocks of data in I/O priority order to each of the volumes defined in the data set. This greatly improved the elapsed time of stand-alone dump, but it also increased the amount of time needed to initialize and navigate the dump, using IPCS.

In z/OS V1R9, stand-alone dump captures the page frame table space in ascending virtual storage order. It uses the table as a pre-built dump index that relates each absolute storage page, as managed by RSM, to its virtual storage identity. This allows IPCS to map each of the associations in the dump directory and improves the time required to handle typical IPCS dump analysis requests for virtual storage, improving the navigation through a large stand-alone dump.

In z/OS V1R10, additional enhancements improve the management of large stand-alone dumps. Initiating a stand-alone dump is easier through the VARY XCF,OFFLINE,SADMP,REIPL command. This command is described, along with the AutoIPL and auto-stand-alone dump features introduced in z/OS V1R10. See "Don't keep me down: Using AutoIPL to minimize z/OS downtime," on page 44.

Subset dumps with COPYDUMP

IPCS COPYDUMP is the recommended tool for transcribing dumps and generating subset dumps. When invoking COPYDUMP, the user specifies a list of ASIDs to extract from the dump and write to an output data set, which results in a (smaller) subset dump that can be sent to IBM (or ISV) Service in lieu of the full stand-alone dump. IBM provides instructions to extract and send the first 20 ASIDs to create the subset dump. However, numeric ASIDs are not as convenient to specify as job names. For example, the system can restart several important address spaces as different ASIDs from when the system was IPLed.

COPYDUMP now allows you to specify by name the set of address spaces making up the subset dump. The following COPYDUMP syntax results from this change:

COPYDUMP ODS('OUTPUT DATASET NAME')
JOBLIST(j1,j2,j3,...)
IDS('INPUT DATASET NAME')
NOCONFIRM

IPCS supports a panel that allows you to enter COPYDUMP information more easily. You can specify the following lists:
- Input data sets
- Job names to include in the subset dump
- Output data set name.
You can also allow the panel to default the list of address spaces that represent core z/OS address spaces such as PCAUTH, RASP, TRACE, DUMPSRV, and others.

To summarize, you perform the following end-to-end sequence to process a stand-alone dump,

1. Take the dump (through the hardware console or the VARY XCF command).
2. Use COPYDUMP to extract a subset dump and write it to a striped data set.
3. Compress the data set. As of z/OS V1R9, TRSMAIN is available in z/OS as the AMATERSE utility (with rollback to V1R7 through an APAR).
4. Send the data set package to IBM (or your local ISV service) using an FTP program.
5. The IBM Service organization obtains the compressed data set and attempts to initialize the dump.

In order to help shorten this path, you can extract the dump directory and send it with the dump; the receiver of the dump can associate the directory with the dump. Use COPYDDIR EXPORT to copy the dump directory to a data set that is sent with the dump. IBM Service can then use COPYDDIR IMPORT to associate the directory with the dump. Following this procedure will help IBM access the data in your dump sooner!

**Support for transaction dumps greater than 2 GB**

Transaction Dump (IEATDUMP) was introduced in OS/390® as a programmable way to request a dump of an application address space, written to an automatically allocated data set. You can specify a number of parameters such as a dump title, address range, and a data set name pattern used to create the data set that is to contain the dump (similar to what you would specify with the DUMPDS NAME= parameter.) The dump is captured to a data space and then written to a new data set, using the name pattern. However, only a single 2 GB data space is used, and IEATDUMP does not support writing to multiple data spaces. A new parameter &DS (dump section) is supported to represent a section of the dump. By specifying a value like ‘..T&DS.’, IEATDUMP will write the first dump with T001 in the name, the second dump will have T002 in the name, and so forth until the entire dump is written.

**Detecting hung SVC dump processing**

It can be very frustrating when a component or subsystem fails, resulting in an SVC dump, yet only to have the dump be rejected because another dump is being processed at the same time. The cause is an internal serialization used to prevent a concurrent SDUMP from running. As a result, if the DUMPSRV address space is hung, because of an I/O or allocation problem, the system rejects all dumps with an indication that a dump is already in progress. But you know that there is no SDUMP in progress! The typical recovery action is to recycle the DUMPSRV address space, which allows future dump requests to be honored but results in losing any captured dumps that have not yet been written to data sets.

In z/OS V1R10, SDUMP hang situations are detected and the DUMPSRV address space is dumped as a result, using IEATDUMP. This allows z/OS Service to determine what is causing the hang. This dump writes to an automatically-allocated data set using the default SDUMP data set name pattern, along with the &DS option to accommodate a dump that is greater than 2 GB.

**The future**

It’s clear to see that IBM is positioning z/OS Serviceability to scale up with the increasing size and speed of System z systems. Future z/OS releases will continue to enhance non-disruptive dump processing, as well as improve formatting and analysis of SVC Dumps, improve stand-alone dump capture and transmission, and make improvements to z/OS diagnostic data itself.
Cut up this page!

Although there are currently no local or international laws against cutting up an issue of z/OS Hot Topics, we usually discourage our readers from doing this. (In fact, we strongly recommend that you store issues of the newsletter in acid-free plastic bags for long term preservation and protection. While actually handling the newsletter, we recommend you wear gloves to protect the issue from finger oils.) However, our desire for you to obsessively protect and preserve the newsletter does not apply to this page. Please cut it up!

We want you to cut up this page, because it contains your own personal z10 server. While this particular one won’t have the power or capacity of an actually z10 (it’s just paper), it will sit easily on the corner of your desk. So forthwith, here are your instructions:

1. Simply cut out the shape along the thick dashed white lines, and fold where indicated by the dotted lines. This will form the rectangular shape of the server.

2. Prevent the server from unfolding, by taping tabs A and B to the inside of the folded shape. A free strip of tape is provided on page 11 (Just joking, get your own tape! But do check out the cool article “Fortune favors the prepared system.”)

3. Display your personal z10 on your desk, the dashboard of your car, the coffee table in your house, or anywhere else your friends can see it and be suitably impressed. If your friends get jealous (and this is only natural), tell them they can get their own by downloading this issue from the z/OS Hot Topics Web site at:

   ibm.com/servers/eserver/zseries/zos/bkserv/hot_topics.htm

Why not snap a digital photo of you and your z10 (perhaps by a famous landmark, natural wonder, or adorable baby), and send it to us at newsletr@us.ibm.com? If we really like it, we may post it on the Hot Topics Web site.
Making fast work of your PSP buckets

Using SMP/E V3R5 to identify and install fixes

BY KURT QUACKENBUSH

Have you ever installed a software product and tried to install all service recommended by the product’s preventive service planning (PSP) bucket? Have you ever prepared to install new hardware and wanted an easy way to identify and install required software service? Did you ever want an easy way to install coexistence service or cross-product dependencies for a new z/OS release? If the answer to any of these questions is “yes,” keep reading to learn how z/OS V1R10 SMP/E (separately orderable as SMP/E V3R5) can help simplify those tasks.

Current methods

You can use several methods to identify recommended software fixes when installing a new software release or a new hardware device, or when enabling new hardware or software functions. However, if you currently study PSP buckets at all, you probably either manually review the appropriate PSP bucket for the release, hardware, or function, or you use the Enhanced PSP Tool (EPSPT) to programmatically compare your current inventory to a known list of fixes.

But in the end, neither of these methods is very satisfying because they are manual, error prone, and not integrated into well-known tools and procedures. What’s more, they assume you can determine which PSP bucket upgrade and subset to use!

A better solution has three approaches:

1. Reformulate the information in PSP buckets to group and associate required and recommended PTFs into easily recognized categories of fixes.
2. Deliver this new information in a form understood by SMP/E using existing service acquisition offerings and procedures.
3. Extend SMP/E to exploit this new information for easier installation of groups of fixes and to identify missing fixes.

This error HOLDDATA model is perfect for the PSP bucket information! IBM will create a new form of HOLDDATA, called FIXCAT (fix category), to group the recommended and required PTFs identified in PSP buckets into specific categories of fixes. FIXCAT HOLDDATA will identify APARs, and each APAR will be resolved (fixed) by subsequent PTFs. Here is an example of a FIXCAT ++HOLD statement:

```
++HOLD(HBB7740) FMID(HBB7740)    /* Held FMID   */
REASON(AA20418)                /* The APAR     */
FIXCAT /* Associates an APAR to a fix category */
CATEGORY(IBM.Device.z10-EC-2097) /* A fix category */
RESOLVER(UA39342)              /* The fixing PTF */
```

PSP buckets reformulated — new HOLDDATA

Today, service processes for the IBM z/OS platform produce error HOLDDATA to identify errors in software products and PTFs. Each error is uniquely identified by an APAR, and each APAR is resolved (fixed) by specific PTFs. Error HOLDDATA is delivered with all corrective service, preventive service, and product orders, and is used by SMP/E to ensure that critical PTFs to fix identified APARs are installed.

In this example, the fix category value IBM.Device.z10-EC-2097 indicates that APAR AA20418 identifies required support for the IBM System z10 EC server (device type 2097), and PTF UA39342 fixes the APAR and provides the required support on z/OS V1R9. The category value IBM.Device.z10-EC-2097 is easier to identify than a PSP bucket upgrade and subset.
Here are other sample fix categories to be used by IBM:

IBM.Device.z9-BC-2096
IBM.Device.z9-BC-2096.zAAP
IBM.Device.z9-BC-2096.zIIP
IBM.Function.HealthChecker
IBM.Coexistence.z/OS.V1R10
IBM.ProductInstall.RequiredService

More values will be added over time, but this list should give you a sense of what to expect. The goal is to use fix category names that are recognizable and self-describing.

Delivering the new HOLDDATA

Just like error HOLDDATA, this new FIXCAT HOLDDATA will be delivered by IBM with all corrective service, preventive service, and product orders such as SMP/E Internet Service Retrieval (the RECEIVE ORDER command), ShopzSeries, ServiceLink, TechSupport, and the Customized Offerings (such as CBPDO, ServerPac, and SystemPac®). It will also be available independently on an IBM Internet server (http://service.software.ibm.com/holddata/390holddata.htm). The FIXCAT HOLDDATA will accompany the error HOLDDATA in the SMPHOLD file, and will be processed by SMP/E V3R5 along with the error HOLDDATA.

Important toleration action

SMP/E releases prior to SMP/E V3R5 cannot process the new FIXCAT HOLDDATA. If you are on either of those releases, you must install toleration APAR IO07480 (PTF UO00700 for SMP/E V3R3 or UO00701 for SMP/E V3R4) to allow those releases to silently ignore the FIXCAT HOLDDATA.

Easier installation and verification with SMP/E V3R5

To simplify installation of groups of fixes, and to identify missing fixes, we have updated SMP/E V3R5 in several ways to exploit the new FIXCAT HOLDDATA. First, when a FIXCAT ++HOLD statement is received, the resolving PTF identified on the ++HOLD statement is assigned a source ID that matches the fix category on that statement. Using our previous example FIXCAT ++HOLD statement, PTF U3A39342 is assigned a source ID of IBM.Device.z10-EC-2097. Your first thought might be, “I thought source ID values were short and cryptic.” Well, they don’t have to be any longer. SMP/E V3R5 will support source ID values of up to 64 mixed-case characters.

Your next thought might be, “Gee, that source ID could be useful on the APPLY command.” It sure can be. Suppose you want to install all of the PTFs that provide support for the System z10 server. First, you must obtain current PTFs and all applicable FIXCAT HOLDDATA to ensure that those useful source IDs are assigned. (Using the SMP/E RECEIVE ORDER command is a great method to get the very latest PTFs and HOLDDATA.) Then you could do something like this:

APPLY CHECK GROUP EXTEND BYPASS(HOLDSYS) SOURCEID(IBM.Device.z10-EC-2097).

If you don’t know the exact source ID value to use, but do know the marketed name or the device type for the System z10 server, you can use wildcards to specify a matching source ID value. For example, you can specify *Device*”z10” or IBM. Device.*2097* or some variation.

Using another example fix category, if you want to install all of the z/OS V1R10 coexistence PTFs required on z/OS V1R8, after you obtain current PTFs and the latest HOLDDATA you could do something like this:

SET BDY(ZOSR8T).
APPLY CHECK GROUP EXTEND BYPASS(HOLDSYS) SOURCEID(IBM.Coexistence.z/OS.V1R10).

Using the source IDs created from the FIXCAT HOLDDATA is a great way to install groups of PTFs for specific device support, software functions, or release coexistence support. However,
Suppose six months from now you want to see whether any additional PTFs are required for the System z10 server or for z/OS coexistence. To do that, use the new REPORT MISSINFIX command, like this:

SET BDY(GLOBAL).
REPORT MISSINFIX ZONES(ZOSR8T)
FIXCAT(IBM.Category.z/OS.V1R10),
IBM.Coexistence.z/OS.V1R10).

The REPORT MISSINFIX command checks your global zone for FIXCAT HOLDDATA that matches the specified FIXCAT values, compares the APARs identified in that FIXCAT HOLDDATA with the PTFs installed in the specified zones, and produces a report to identify any APARs that are not resolved. In other words, it reports which PTFs (fixes) are missing for the specified fix categories.

Figure 1 shows a small sample report produced by the REPORT MISSINFIX command:

<table>
<thead>
<tr>
<th>FIX CATEGORY</th>
<th>FMID</th>
<th>CLASS</th>
<th>APAR</th>
<th>SYMMD</th>
<th>NAME</th>
<th>STATUS</th>
<th>RECEIVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM.Coexistence.z/OS.V1R10</td>
<td>HBB7740</td>
<td>AA12345</td>
<td>HBB7740</td>
<td>UA23456</td>
<td>GOOD</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23456</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23719</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23947</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HLE7740</td>
<td>UA54343</td>
<td>UK33146</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td>IBM.Device.z10-EC-2097</td>
<td>HBB7740</td>
<td>AA12345</td>
<td>HBB7740</td>
<td>UA23456</td>
<td>GOOD</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23456</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23719</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AA23947</td>
<td>HBB7740</td>
<td>U34567</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HLE7740</td>
<td>UA54343</td>
<td>UK33146</td>
<td>GOOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HLE7740</td>
<td>UA54343</td>
<td>UK31548</td>
<td>GOOD</td>
<td>NO</td>
</tr>
</tbody>
</table>

Figure 1. Example of a report produced by REPORT MISSINFIX

Similar to when you install a new hardware device, when you install a new software release (a new FMID), it is important to also install all the recommended PTFs that are identified for that FMID in the associated PSP bucket. Installing the recommended PTFs protects you from experiencing known and fixed problems, and SMP/E V3R5 makes this task easier.

Once again you must first obtain all applicable FIXCAT HOLDDATA, and then you can simply apply the recommended PTFs along with the new FMID. You don't even need to know which PSP bucket upgrade and subset to read! Just specify a fix category value of IBM.ProductInstall.RequiredService on the APPLY command. This one common fix category identifies recommended PTFs for all releases of all software products!

The FIXCAT operand on the APPLY command tells SMP/E that all FIXCAT HOLDDATA with a matching category value must be resolved. Further, the GROUPEXTEND operand tells SMP/E to automatically find PTFs that resolve those holds, thus ensuring that all recommended PTFs are installed.

An important point to note: If you don’t specify the FIXCAT operand on the APPLY command, the FIXCAT HOLDDATA in the global zone does not affect the APPLY command. So, do not be afraid to receive the HOLDDATA, and do not be worried that your APPLY tasks will be affected without your knowledge.
Fix Category Explorer

You might now wonder “How can I see all the great fix category values that I can use?” Well, using the new SMP/E Fix Category Explorer dialog, you can view all the fix category values from all the FIXCAT HOLDDATA in your global zone in a hierarchical manner. (See Figure 2.) You can view high-level parent values, and drill down to view lower levels and values. The Fix Category Explorer is integrated into the existing SMP/E dialog options, and can be used, for example, to select values for the FIXCAT operand on the APPLY, ACCEPT, and REPORT MISSINGFIX commands in the Command Generation dialog.

---

### Fix Category Explorer

- **IBM.*
- **IBM.Coexistence.*
  - **IBM.Coexistence.z/OS.*
  - IBM.Coexistence.z/OS.V1R9
  - IBM.Coexistence.z/OS.V1R10
- **IBM.Device.*
  - IBM.Device.ESS-Shark-2105
  - IBM.Device.FICONDirector-2062
  - IBM.Device.HighPerformanceTape-3591
  - IBM.Device.RAMACElectronicArray-9397
  - IBM.Device.a/390Multiprise3000-7060
  - IBM.Device.VirtualTapeServerVTS-3495
  - IBM.Device.z10-EC-2097
  - **IBM.Device.z10-EC-2097.*
  - IBM.Device.z10-EC-2097.Capacity Provisioning
  - IBM.Device.z10-EC-2097.Decimal Floating Point
  - IBM.Device.z10-EC-2097.MIDAW
  - IBM.Devic..z10-EC-2097.ServerTimeProtocol
  - IBM.Device.z10-EC-2097.zAAP
  - IBM.Device.z10-EC-2097.zIIP
  - IBM.Device.z800-2066
  - IBM.Device.z890-2086
  +IBM.Device.z890-2086.*
  - IBM.Device.z9-BC-2096
  +IBM.Device.z9-BC-2096.*
  - IBM.Device.z9-EC-2094
  +IBM.Device.z9-EC-2094.*
  - IBM.Device.z900-2064
  +IBM.Device.z900-2064.*
  - IBM.Device.z990-2084
  +IBM.Device.z990-2084.*
  +IBM.Function.*
  +IBM.ProductInstall.*

********* Bottom of data *********

---

### Conclusion

For more detailed information about FIXCAT HOLDDATA and these great changes in SMP/E V3R5, see IBM Education Assistant SMP/E 3.5 – Simplifying PSP buckets at [http://publibz.boulder.ibm.com/zoslib/pdf/IEA-PDF-SMP35-SimplifyingPSPBuckets.pdf](http://publibz.boulder.ibm.com/zoslib/pdf/IEA-PDF-SMP35-SimplifyingPSPBuckets.pdf) and SMP/E V3R5.0 Commands, SA22-7771.
As a software developer specializing in I/O, 90% of my job is doing my best to make sure your devices are available by way of software. However, there is very little I can do for devices that have lost connectivity with the host. Devices that have paths that share a common piece of hardware are vulnerable, if that piece of hardware fails. Even if a primary and alternate do exist, common components can exist in those devices. Luckily, there’s a new service in z/OS V1R10 called the IOSSPOF service and a new tool called the IOSSPOFD tool to determine which devices are safe and which ones might be at risk.

**IOSSPOF service**

To set up an I/O system that has no single points of failure you must know the hardware paths to devices that are shared. Having one CHPID online is an obvious single point of failure; however, it is not always obvious that even numbered enclosures on a 2107 control unit share a power supply. The IOSSPOF service checks for 16 different types of single points of failure. The checks vary from simply verifying that two data sets are on different devices to more complicated situations such as checking that all the CHPIDs configured to the device don’t share a Self-Timed-Interface. For the complete list and documentation, see the IOSPF messages in z/OS MVS System Messages, Volume 9 (IGF-IWM), SA22-7639.

The service is primarily a macro interface referred to as IOSSPOF. It reports problems through a control block interface that is mapped in IOSDSPOF, by issuing write-to-operator (WTO) messages on behalf of the caller, or by issuing IBM Health Checker for z/OS messages for the caller. The IOSSPOF macro can be invoked by a caller in any key or any mode to allow developers to come up with interesting ways of using this information. For information on IOSSPOF, see MVS Programming Authorized Assembler Services Reference (ENFREQ-IXGWRITE), SA22-7610.
We hope that others will pick up on the potential of writing checks to verify the availability of devices that house important data sets. Figure 1 is an example of the XCF_CDS_SPOF check using the IOSSPOF service to write a report that shows if the primary and alternate couple data sets share hardware components. Loss of primary and alternate couple data sets will mean a multi-system outage. System data sets aren’t the only data sets that are important to your installation. With a little development skill, anybody can write their own SPOF report and by using remote checks, you can integrate SPOF reports into your software product.

Quick tip: Using HZSADDCHECK UPDATE with the INTERVAL keyword creates a check that monitors the device for single points of failure.

**IOSSPOFD tool**

On top of the single point of failure service is the IOSSPOFD tool, available on the z/OS Tools and Toys Web site at [ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html](http://ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html). IOSSPOFD acts as a JCL interface to the single point of failure service. It allows you to quickly check for full availability of your data sets.

Figure 2 shows the JCL to generate a report of single points of failure. This tool also allows wildcards in describing its input. DEVLIST(410-420,980,981) checks multiple devices in one line. See the IOSSPOFD documentation for more information. You can put it to good use by verifying new hardware installations, confirming paging data sets for availability, and checking if that path you varied offline could open up a potential problem. Go ahead, save your eyes from the strain of staring at HCD connection diagrams and give automatic SPOF detection a try… I can’t think of a better way to prevent hardware errors before they happen.

```bash
//IOSSPOFD JOB MSGLEVEL=(1,1)
//STEP EXEC PGM=IOSSPOFD,TIME=WOLIMIT
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSIN DD *
//DSNLST(SYS1.NUCLEUS, SYS1.LINKLIB, DB2.DATABASE)
/* CHECK IMPORTANT DATASETS FOR SPOFS */
/*
```

Figure 2. Example of input to IOSSPOFD checking important data sets
Don’t miss the goal

z/OS Capacity Provisioning can help

BY MICHAEL GROETZNER, HORST SINRAM, AND RALF THELEN

Do you sometimes have nightmares of customers flooding your systems with orders that your servers do not have the capacity to process? Or of important end-of-quarter jobs not getting enough cycles because of other important work?

Thanks to the IBM System z10 Enterprise Class and Capacity Provisioning in z/OS Release 9, the solution is simpler than ever before. Capacity Provisioning gives you the flexibility to activate temporary capacity on a given schedule or based on workload performance. If you choose the workload-based model, you can define a policy to activate capacity only within specified periods when your important work misses its goals because of a capacity bottleneck.

You say that you like the idea of a workload-based model, but you still want the control of making that final decision to activate additional capacity? No problem. Capacity Provisioning, along with your policy, can provide suggestions as to what type of processor (zIIP, zAAP, or general purpose) can help get you back on track, yet no action will be taken without your confirmation.

Just-in-time capacity on IBM System z10

The new Capacity on Demand architecture of the IBM System z10 supports several types of temporary capacity offerings, such as Capacity Back-Up (CBU) and On/Off Capacity on Demand (On/Off CoD). You can install and activate these offerings in a z10 at the same time. Alternatively, you can activate portions of each offering, up to its defined maximum of resources, as long as there are still unused physical engines available. This flexibility raises two questions for the user:

1. How do you find out which portions of an offering can be activated, given other concurrently active offerings and the remaining unused physical engines in the system?
2. How do you determine which of the activation choices best fits your current need for additional capacity, while considering both cost and performance?

You can select the entry that best meets your capacity requirements and your budget. It’s your choice whether to add CPs or change the sub-capacity setting, or do both.

To help with selecting the desired amount of zIIP and zAAP engines, the panels provide the maximum number allowable for each type, and the number of currently unused physical engines that are available for activation.

The System z Application Programming Interfaces (APIs) provide the same information and send notifications to registered users whenever the resource state changes. For example, every time one of the offerings in the system is activated or deactivated.

Let’s see how Capacity Provisioning uses these System z10 hardware features to adjust the machine’s physical processing capacity.

How Capacity Provisioning works

At the heart of Capacity Provisioning is the Capacity Provisioning Manager (CPM), a server address space running on z/OS. Besides listening to commands, CPM can manage one or more System z10 servers through the SE or the HMC.

For workload-based provisioning, Capacity Provisioning requires information about the work running in the z/OS systems. CPM can observe many z/OS systems on the managed servers; these systems do not need to be part of the same sysplex. In each system, workload manager (WLM) manages the workload. Performance information about the workloads, such as their
In management of temporary capacity, CPM offers four processing modes that provide different automation levels for the components interact.

Capacity Provisioning policy
The analysis, confirmation, and autonomic modes require a capacity provisioning policy. A capacity provisioning policy specifies:

- Which workloads can trigger activation of temporary capacity (workload condition)
- How much additional capacity is allowed to be activated (provisioning scope)
- When additional capacity can be activated (time condition)

According to the combination of these aspects, schedule- or workload-based provisioning is accomplished.

Schedule-based provisioning
If you know that a certain amount of additional capacity is necessary at a given time interval, you can define a policy that provides for a scheduled activation and deactivation of resources. Here, you would specify a policy with time conditions representing the time intervals and a provisioning scope describing the resources. When you activate the policy, CPM activates the specified resources when a specified interval starts and deactivates the specified resources when the interval ends.

Workload-based provisioning
To have additional capacity activated only when a critical workload is lacking capacity, define a policy that includes workload conditions and provisioning scope. These are described as follows:

- **Workload conditions** identify one or more WLM service class periods running critical workload. You specify a performance index (PI) at which the workload running in the service class periods is considered to be suffering, and a PI at which the workload is considered to be doing well.
- **Provisioning scope** defines the maximum additional capacity that can be activated when such a service class period suffers.

In analysis mode, CPM informs you when additional capacity is needed according to the policy, but CPM does not activate it. This processing mode is useful to verify your policies. You can also use the provided information to change the capacity of your CPC. If you do not yet have a System z10 server or an On/Off CoD record but want to benefit from Capacity Provisioning, you can also run in this mode.

In confirmation mode, each action of CPM that changes the capacity of a CPC has to be approved before it is performed. Therefore, messages display on the MVS console where you can respond with accept or reject. Confirmation mode is useful when you are starting to exploit Capacity Provisioning to verify your policy or to explicitly control and authorize proposed capacity changes.

In autonomic mode, a policy managed by CPM automatically drives the management of capacity on your CPCs. Actions to change the capacity of the CPCs are performed as soon as sufficient demand is detected.

performance indexes and delays, are provided by WLM, and retrieved through Resource Measurement Facility (RMF) and Common Information Model (CIM) servers.

You define a Capacity Provisioning policy on a workstation using the Capacity Provisioning Control Center (CPCC) graphical user interface (GUI). After installing the policy in the repository data set on z/OS, you can activate it for use by CPM. Figure 1 shows how these components interact.

Capacity Provisioning processing modes
CPM offers four processing modes that provide different automation levels for the management of temporary capacity:

- **In manual mode**, CPM represents an easy to use command interface to the hardware. For example, the MVS console command **ACTIVATE RESOURCE CPC=cpcname ZAAP=1** to the CPM would activate one zAAP processor for the specified CPC. CPM commands are supported in all modes.
- **In analysis mode**, CPM informs you when additional capacity is needed according to the policy, but CPM does not activate it. This processing mode is useful to verify your policies. You can also use the provided information to change the capacity of your CPC. If you do not yet have a System z10 server or an On/Off CoD record but want to benefit from Capacity Provisioning, you can also run in this mode.
- **In confirmation mode**, each action of CPM that changes the capacity of a CPC has to be approved before it is performed. Therefore, messages display on the MVS console where you can respond with accept or reject. Confirmation mode is useful when you are starting to exploit Capacity Provisioning to verify your policy or to explicitly control and authorize proposed capacity changes.
- **In autonomic mode**, a policy managed by CPM automatically drives the management of capacity on your CPCs. Actions to change the capacity of the CPCs are performed as soon as sufficient demand is detected.

Figure 1. Components of z/OS Capacity Provisioning
Additional capacity is activated if a specified service class period suffers and a significant reason is insufficient capacity. CPM analyzes which type of capacity (general purpose capacity, zAAPs, or zIIPs) is suitable to help the workload (or to help the workload in the service class period).

Further, the list of allowed activation choices is retrieved from the System z10 server and the appropriate amount of capacity is determined and activated up to the limit specified by the provisioning scope. Previously activated additional capacity is deactivated if the service class periods fall below the specified performance index threshold.

If provisioning should only be allowed at particular time intervals, time conditions can be specified to indicate when workload conditions are to be considered by CPM.

In Figure 2, the first period of the service class is defined so that it is considered suffering if its PI is in excess of 1.8 for more than 10 minutes. Resources can be deactivated if the PI is below 1.3 for more than 20 minutes.

**Getting started**

Get Capacity Provisioning with z/OS V1R9 (and APAR OA20824) or with z/OS V1R10. We recommend that you begin with the overview presentations and then move on to the instructions for installation, configuration, and exploitation in IBM MVS Capacity Provisioning User’s Guide, SC33-8299. All the information is available from the z/OS Capacity Provisioning Web site at [ibm.com/servers/eserver/zseries/zos/wlm/cp](http://ibm.com/servers/eserver/zseries/zos/wlm/cp).

IBM Redbooks IBM System z10 Enterprise Class Capacity on Demand, SG24-7504 contains information about On/Off Capacity on Demand as well as practical hints and tips for using Capacity Provisioning at [redbooks.ibm.com/redpieces/abstracts/sg247504.html](http://redbooks.ibm.com/redpieces/abstracts/sg247504.html).

**Special thanks**

This article was written in part by Christine Axnix, Charles Haight, and Stefan Wirag. Their expertise contributed greatly to this article. Thank you all for your outstanding assistance.

---

**Figure 2:** A sample workload condition
Full speed ahead with HiperDispatch

BY DANIEL ROSA AND DONALD SCHMIDT

The new IBM System z10 mainframe supports a new function called HiperDispatch (HD), which increases the capacity of the system from zero to ten percent. HiperDispatch is a collaborative effort between hardware, hypervisor, and z/OS to increase the probability of getting a cache hit. A cache hit reduces the number of machine cycles the hardware needs to process a given instruction. As the percentage of cache hits increase throughout the system, the effective capacity of the machine increases because fewer machine cycles are needed to process a given set of instructions.

Cache overview
Each processor has its own level 1 (L1) cache. A collection of processors shares a common level 2 (L2) cache. A group of processors that share the same L2 cache is called a book at the hardware level. A processor can also access a remote book’s L2 cache. The larger the distance between the data and the processor, the longer it takes a given instruction to complete. The fastest place to find a piece of data is in the L1 cache, because it costs the fewest machine cycles to access that data. As one would expect, L2 cache for the same book is second fastest, followed by L2 cache for a remote book, followed by memory. Because it takes more machine cycles to access the L2 cache for a remote book, HiperDispatch is focused on optimizing the L2 cache for the same book to reduce the number of times that remote L2 cache is accessed. HiperDispatch also increases the number of L1 cache hits.

HIPERDISPATCH=NO
To understand how HiperDispatch improves the probability of getting a same-book L2 cache hit, let us examine how work was dispatched historically. We can imitate this behavior on a System z10 server by setting the HIPERDISPATCH=NO parmlib option. A hypothetical example in Figure 1 shows a System z10 server with eight physical processors, where four processors are in book 1 and the other four processors are in book 2. The System z10 mainframe is running a green LPAR and a blue LPAR, where each LPAR has eight logical processors. When HIPERDISPATCH=NO is specified, an LPAR’s share is distributed across all active logical processors. Processor Resource/Systems Manager™ (PR/SM™) dispatches a logical processor on a physical processor, and z/OS chooses a thread to run on that logical processor. With HIPERDISPATCH=NO, PR/SM can dispatch any logical processor on any physical processor. In addition, z/OS can dispatch any thread on any given logical processor. Effectively between PR/SM and z/OS, HIPERDISPATCH=NO dispatches threads randomly on physical processors and prevents cache optimization.

Figure 1. HIPERDISPATCH=NO versus HIPERDISPATCH=YES
HIPERDISPATCH=YES
With HIPERDISPATCH=YES, threads are intelligently dispatched on physical processors. Work is generally restricted to run on a subset of the logical processors defined to an LPAR based upon the weights of all LPARs across the central processor complex (CPC). In each LPAR, the hypervisor designates each logical processor to be a vertical high, a vertical medium, or a vertical low. A vertical high gets 100 percent of a physical processor. There is a one-to-one mapping between a vertical high and a physical processor. A vertical medium gets 1 to 99 percent of a physical processor. Multiple vertical mediums can be mapped to the same physical processor. A vertical low is a discretionary processor that normally does not dispatch work.

To return to our example in Figure 1, logical processors 1-4 for the blue LPAR are vertical highs and have been assigned to physical processors 1-4. For the green LPAR, logical processors 1-4 are also vertical highs and have been assigned to physical processors 8-5. With HIPERDISPATCH=YES, each LPAR still receives the same amount of physical processor time as with HIPERDISPATCH=NO. With HIPERDISPATCH=YES, however, each LPAR normally uses fewer logical processors. With HIPERDISPATCH=YES, PR/SM rediscatches a given logical processor on the same physical processor, and z/OS will dispatch a thread amongst a relatively small static collection of processors. Because PR/SM dispatches logical processors 1-4 for the blue LPAR on physical processors 1-4, and z/OS dispatches threads A-H on logical processors 1-4 for the blue LPAR, more L2 cache hits occur. As a result, fewer machine cycles are needed to process a set of instructions. The same is true for the green LPAR.

But what happened to the other four logical processors in each LPAR? For the blue LPAR, logical processors 5-8 float on physical processors 5-8; for the green LPAR, logical processors 5-8 float on physical processors 4-1. Normally, the vertical lows for an LPAR are in a parked state. z/OS does not dispatch any work on a parked logical processor, so PR/SM does not dispatch that parked logical processor. Why? If the blue LPAR dispatched its logical processors 5-8, it would cause remote L2 cache accesses because the data for the blue LPAR is in book 1. Cache for the green LPAR would also be adversely affected because the data for the green LPAR would be removed to add data for the blue LPAR. So, vertical lows are normally off limits for a given LPAR.

Vertical lows
Vertical lows can be used to dispatch work for a given LPAR when that LPAR is consuming its full weight and there is another LPAR on the system that is not consuming its full weight. When at least one LPAR in the CPC is not consuming its full weight, there is white space in the system. When an LPAR has the right to expand into its vertical lows, a vertical low is unparked and that logical processor starts dispatching work. In Figure 2, the blue LPAR was consuming its full weight and had additional work to perform. Because the green LPAR was not consuming its full weight, the blue LPAR expanded into the white space created by the green LPAR by unparking logical processor 5. Logical processor 5 for the blue LPAR will be parked when the blue LPAR is no longer trying to consume more than its full weight or when the green LPAR consumes its full weight and the white space in the system disappears.

Affinity nodes
Workload Manager (WLM) divides all processors of the same processor type into groups called affinity nodes. Most affinity nodes are composed of processors entirely within the same book to optimize the same book L2 cache. Each affinity node contains only processors of the same type, and has its own work unit queue (WUQ). The processors that are assigned to a given affinity node consume work from the affinity node’s WUQ. When a WUQ goes from empty to non-empty, one processor within that affinity node is signaled to dispatch work. As work continues to be added to that affinity node, other processors assigned to that affinity node (called buddies) are also signaled to consume work from that WUQ.

Help processing
On a regular interval, WLM’s balancer runs on z/OS to distribute work evenly across all affinity nodes by processor type and priority. Between these WLM intervals, supervisor detects address space utilization spikes in real-time. These spikes occur when a processor within an affinity node has not entered a wait in a while, or when an affinity WUQ’s queue depth is too long for the number of processors dispatching work from

---

**Figure 2. HIPERDISPATCH =YES and vertical lows**

---

**Key:**
- **PP** = Physical processor
- **LP** = Logical processor
- **A-H** = Threads
it. When this occurs, the affinity node needs help. The supervisor searches for a processor in another affinity node that is a good candidate helper to assist the overworked affinity node. Ideally, a good candidate helper is a processor that is within the same book and that spends a good amount of its time waiting. If Supervisor is unable to find a processor with a lot of wait time, a helper processor running lower priority work can be chosen. Any processor that is chosen to help dispatches the highest priority unit of work between its own affinity WUQ and the affinity WUQ being helped until a preset threshold is reached. When the threshold is reached, the processor stops helping until it is chosen to help again.

Capacity boost
The capacity boost provided by HiperDispatch depends on the configuration of the System z10 mainframe and the workload running. A System z10 server with all physical processors in one book will see minimal benefit from HiperDispatch because the same book L2 cache is already optimized. Even in this case, IBM still recommends enabling HiperDispatch, because HiperDispatch has shown some capacity benefits even on small N-ways. As the number of physical processors increases on a System z10 server, HiperDispatch provides a larger capacity benefit. The workload characteristics also factor into how much capacity increases with HiperDispatch. In general, workloads that frequently access memory or have a high task dispatch rate will see the most benefit with HiperDispatch. Batch workloads that are compute intensive usually see lower capacity improvements with HiperDispatch.

Enabling HiperDispatch
The HiperDispatch keyword in IEAOPTxx in SYS1.PARMLIB controls whether HiperDispatch is enabled or disabled, as described in MVS Initialization and Tuning Reference, SA22-7592. HIPERDISPATCH=NO is the default, so customers migrating to a System z10 server can just test the new hardware before enabling the HiperDispatch function. After testing the new hardware, the system programmer should migrate all LPARs to HIPERDISPATCH=YES to improve the capacity of the System z10 server. A new Sup_HiperDispatch Health Check will be registered on all LPARs to check whether HiperDispatch is active, as described in IBM Health Checker for z/OS User’s Guide, SA22-7994. For information about planning for HiperDispatch, see the IBM white paper entitled Planning Considerations for HiperDispatch Mode, which you can find by searching for “HiperDispatch” on the Techdocs Web page:

ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs

Even though the benefits of HiperDispatch have been included in the IBM Large System Performance Reference (LSPR) pricing of the System z10 server, enabling HiperDispatch can benefit all kinds of workloads. So enable HiperDispatch on your System z10 mainframe, and find out how fast full speed ahead is.
Get positioned for recovery

Extending GDPS across the enterprise

BY NOSHIR DHONDY

Celebrating its tenth anniversary in 2008, Geographically Dispersed Parallel Sysplex (GDPS) is evolving! Originally, GDPS was introduced as a continuous availability (CA)/Disaster Recovery (DR) solution for customers that had a multi-site sysplex and used a synchronous remote copy technology to replicate their data. Now, GDPS can also provide the following:

- Act as a business continuity and resiliency solution for enterprises that might not have a sysplex and are using asynchronous remote copy technologies at unlimited distances.
- Manage the data across heterogeneous platforms with a function called open logical unit number (LUN) management.
- Provide data management and coordinated recovery for applications that are running across z/OS and Linux on System z with a function called multiplatform resiliency for System z (also known as xDR).

Let’s explore the distributed cluster management (DCM) function now available with GDPS/PPRC V3.5 that uses IBM Metro Mirror (formerly peer to peer remote copy) technology and GDPS/XRC V3.5 that uses IBM System Storage z/OS Global Mirror technology (formerly extended remote copy or XRC).

Data management and recovery across multiple platforms

Most enterprises have a heterogeneous IT environment where the applications and data reside on a variety of hardware and software platforms like IBM System z, IBM System p™, UNIX®, Windows®, and Linux. Many examples exist of a multi-tier architecture where an application server might be running on a distributed (non-System z) platform, while the database server is running on z/OS. If an outage impacts both platforms but only z/OS data and systems are recovered and not the distributed server platform, from the end user’s perspective, the application is unavailable.

With a multi-tiered architecture, you need to provide a coordinated near-CA/DR solution for both the System z and the distributed platforms. Availability can be greatly enhanced if a single point of control exists to manage the data across all the platforms, and if the DR solution can coordinate the recovery across platforms.

Distributed cluster management

Distributed cluster management or DCM is a new GDPS capability that allows the management and coordination of planned and unplanned outages across the following:

- Distributed servers in a cluster using clustering solutions
- System z workloads for which GDPS is responsible.

DCM functions

A GDPS DCM agent running in each distributed cluster is designed to provide advisory and coordination functions between GDPS and one or more distributed server clusters. The advisory functions can provide the capability of continuous heartbeat and status gathering to alert the support staff about events that can prevent recovery at the time of an outage.
The coordination functions can allow workflow integration for takeover and recovery testing, cross-platform monitoring to maintain recovery capability, and cross-platform recovery management to provide an automated enterprise-level rapid recovery in the case of an outage.

GDPS provides the following functions:

- Monitoring — GDPS generates alerts for DCM resources in an abnormal state.
- Manual operations — You can query the status of DCM resources and perform planned operations on individual DCM managed resources. Examples of DCM resources can be a single application group, all application groups on a single cluster, or all clusters.
- Automation — You can detect failures associated with DCM resources and run automation scripts.
- Scripting — The scripting capability in GDPS provides workflow integration for actions taken on distributed servers and System z servers in the event of a planned or unplanned event.

GDPS provides the following functions:

- Monitoring — GDPS generates alerts for DCM resources in an abnormal state.
- Manual operations — You can query the status of DCM resources and perform planned operations on individual DCM managed resources. Examples of DCM resources can be a single application group, all application groups on a single cluster, or all clusters.
- Automation — You can detect failures associated with DCM resources and run automation scripts.
- Scripting — The scripting capability in GDPS provides workflow integration for actions taken on distributed servers and System z servers in the event of a planned or unplanned event.

GDPS provides the following functions:

- Monitoring — GDPS generates alerts for DCM resources in an abnormal state.
- Manual operations — You can query the status of DCM resources and perform planned operations on individual DCM managed resources. Examples of DCM resources can be a single application group, all application groups on a single cluster, or all clusters.
- Automation — You can detect failures associated with DCM resources and run automation scripts.
- Scripting — The scripting capability in GDPS provides workflow integration for actions taken on distributed servers and System z servers in the event of a planned or unplanned event.

GDPS provides the following functions:

- Monitoring — GDPS generates alerts for DCM resources in an abnormal state.
- Manual operations — You can query the status of DCM resources and perform planned operations on individual DCM managed resources. Examples of DCM resources can be a single application group, all application groups on a single cluster, or all clusters.
- Automation — You can detect failures associated with DCM resources and run automation scripts.
- Scripting — The scripting capability in GDPS provides workflow integration for actions taken on distributed servers and System z servers in the event of a planned or unplanned event.

GDPS provides the following functions:

- Monitoring — GDPS generates alerts for DCM resources in an abnormal state.
- Manual operations — You can query the status of DCM resources and perform planned operations on individual DCM managed resources. Examples of DCM resources can be a single application group, all application groups on a single cluster, or all clusters.
- Automation — You can detect failures associated with DCM resources and run automation scripts.
- Scripting — The scripting capability in GDPS provides workflow integration for actions taken on distributed servers and System z servers in the event of a planned or unplanned event.

A disaster is declared and includes the decision to recover all processing in Site 2. Now that GDPS DCM support provides the ability to manage VCS clusters from GDPS, you now have the ability to switch z/OS systems and data and VCS systems and data to Site 2 in a coordinated manner.

You can extend an existing GDPS script that previously performed failover and restart of System z resources to include statements automating the failover of VCS clusters to Site 2.

Summary
The DCM function of GDPS can provide a single point of control to monitor both System z resources and distributed server resources. DCM can also provide coordinated recovery for planned and unplanned events that can affect either the System z resources or the distributed server resources or both. In short, you can attain business resiliency across your entire enterprise with DCM.

**The DCM function of GDPS can provide a single point of control to monitor both System z resources and distributed server resources**

**DCM support**

IBM provides initial DCM support for Symantec Veritas Cluster Server (VCS) clusters. VCS is a clustering solution that can be used for both planned and unplanned events and can gracefully shut down applications and restart them on an available server. The failover could be to a local server in the same site or to a remote cluster located several thousand miles away for disaster recovery. VCS supports multiple operating systems, such as IBM-AIX®, SUN-Solaris, HP-UX, Windows, and Linux. It also supports multiple hardware, software, and database replication technologies.

IBM has also made a Statement of Direction that GDPS is planning to extend DCM support to include IBM Tivoli® System Automation Application Manager (SA AppMan). This capability is designed to allow GDPS and SA AppMan to work together to provide recovery and failover for System z systems running GDPS, and distributed systems managed with SA AppMan.

**How it works**

Figure 1 shows an example of a GDPS/PPRC configuration with production systems in Site 1 and the GDPS Controlling system (k-sys) in Site 2. Also, in the configuration are two VCS global clusters — an AIX production cluster in Site 1 and its failover cluster in Site 2, and similarly a Linux production cluster in Site 1 with its failover cluster in Site 2. A GDPS agent in each cluster sends a heartbeat to the GDPS k-sys through communication links.

The failure scenario for Site 1 in Figure 2 can occur when one or more of the following failures occur in Site 1:

- A Metro Mirror mirroring failure
- The loss of one or more production z/OS systems
- The loss of one or more VCS clusters in Site 1
Find out more

- For the Statement of Direction for GDPS V3.5 enterprise-wide infrastructure availability, see ibm.com/common/sri/rep_ca/1/897/ENUS308-001/ENUS308001.PDF.
- To learn more about GDPS, visit ibm.com/systems/z/advantages/gdps/index.html.
- For information on open LUN management and multiplatform resiliency for System z, see “Attack of the Clones” in z/OS Hot Topics Newsletter Issue 13, August 2005, GA22-7501-09.

Figure 1. Example of GDPS/PPRC configuration with VCS clusters

Figure 2. Example of GDPS/PPRC configuration with VCS clusters — Site 1 failure
No two computers’ time-of-day (TOD) clocks have exactly the same value. All TOD clocks eventually drift apart because of each clock’s individual characteristics.

Clock drift occurs between computers whenever there is no external form of synchronization between them. Problems associated with clock drift become more complex if the same time reference is required for all of the servers distributed across heterogeneous platforms in an enterprise.

In “This time around: Moving to Server Time Protocol” in z/OS Hot Topics Issue 17, August 2007, you read about the Server Time Protocol (STP) feature, which allows the servers and coupling facilities in your Parallel Sysplex® to maintain time synchronization with each other. STP uses a message-based protocol in which timekeeping information is passed over the externally defined coupling links between servers.

In this article, we look at Network Time Protocol (NTP) client support, which is new for STP. NTP client support allows you to extend the same accurate time reference to the heterogeneous platforms (both System z and non-System z) in your enterprise.

What is Network Time Protocol?
Network Time Protocol (NTP) is an Internet standard protocol for synchronizing the clocks of computers over a network. You can use NTP to synchronize a computer’s local TOD clock to another server or reference time source, such as a radio, satellite receiver, or a dial-out time service. NTP was originally developed by Professor David L. Mills at the University of Delaware. He still maintains it today, along with a team of volunteers.

What is Coordinated Timing Network?
As mentioned in “This time around: Moving to Server Time Protocol” in z/OS Hot Topics Issue 17, August 2007, STP introduced the concept of Coordinated Timing Network (CTN), which is a collection of servers and coupling facilities that are synchronized to a time value called Coordinated Server Time (CST). The servers that make up a CTN share a common identifier, which is called a CTN ID. Only servers with the same CTN ID can join the same CTN.

Time synchronization for all
To allow for precise sharing of data, your installation might have specific business requirements to provide accurate time relative to an external time standard, for time synchronization between its servers and coupling facilities. This requirement might include, for example, the accurate time stamping of DB2 transactions.

Beyond the sysplex, your installation might need to provide the same accurate time reference across heterogeneous platforms (both System z and non-System z). Typical accuracy requirements are within 100 milliseconds (ms) of the time reference. A financial institution, for example, might require this degree of accuracy for time stamping asset transactions.

The NTP client support provided by STP allows you to meet the requirement of accurate time across heterogeneous platforms. If you require time accuracy only within the System z platform, you can use the Hardware Management Console (HMC) to dial out to a time service.

NTP client support allows access to an NTP server as an ETS through an Ethernet connection using the HMC/support element (SE) local area network (LAN).

NTP client support is implemented in licensed internal code (LIC) that runs on the support element of any of the following servers with STP feature number 1021 installed:
- IBM System z10 Enterprise Class (System z10 EC)
- IBM System z9 Enterprise Class (System z9 EC)
- IBM System z9 Business Class (System z9 BC).

In this article, the term server also includes stand-alone coupling facilities. The IBM zSeries® z990 and z890 servers do not provide NTP client support. For these servers, you can use the HMC to dial-out to a time service. The zSeries z990 and z890 servers can participate in an STP-only CTN that includes a System z9 EC or System z9 BC, or a System z10 EC configured to use the NTP client.

NTP client support overview
An NTP server can be the external time source (ETS) for an STP-only CTN (which does not require a Sysplex Timer®), as well as for another non-system z platform running an NTP client (such as UNIX or Windows). The ETS can be a vendor-supplied NTP server.
Figure 1 shows a sample configuration in which the NTP server is connected directly to the HMC/SE local area network.

Redundant Internet NTP servers

For greater availability, you might choose to install and configure two NTP servers on both the PTS and the BTS. See Figure 2.

This configuration adds another local NTP server (Server B) for interfacing with remote NTP servers on the corporate network or even on the Internet. When two NTP servers are configured as shown, the NTP client selects the server with the lowest stratum level or with the lowest dispersion if both servers have the same stratum level.

In Figure 2, for example, an outage on NTP Server A would cause NTP Server B to become the ETS automatically (that is, without user intervention). Later, when NTP Server A resumes normal operations, it automatically becomes the ETS because it is the lowest stratum server. An outage on NTP Server B or C has no effect on the status of NTP Server A as the ETS.

Planning considerations

With proper planning for hardware, software, and connectivity, your installation can migrate its STP-only CTN to use an NTP server as its ETS. At a high level, you must review the following requirements from both the STP and NTP perspectives.

STP minimum requirements

The following are the minimum requirements for STP:

- A supported release of z/OS.
- Feature code FC1021 is installed concurrently on each STP-capable server or coupling facility in the CTN. Each z/OS image requires a reIPL after the installation of FC1021.
- Supported coupling links: InfiniBand (IFB) links, InterSystem Channel-3 (ISC-3) links, Integrated Cluster Bus (ICB-3 or ICB-4) links.
- Timing-only links, which permit server-to-server connection when no coupling facility is defined at either end. Timing-only links might require the purchase of additional coupling facility links.
- PTF for APAR OA13344, which enables z/OS for STP and updates the CLOCKxx member to use STP by default.

Figure 1. Sample configuration

In Figure 1, one of the NTP output ports of the NTP server is connected directly to the HMC/SE LAN. Its other NTP output ports can be connected to non-System z servers that require the same accurate time reference. NTP server access is driven from the Current Time Server (CTS), which we refer to as the Stratum 1 server, every ten minutes.

IBM suggests that you configure at least one NTP server for both the Preferred Time Server (PTS) and the Backup Time Server (BTS). Doing so ensures the continuity of the ETS function across a CTS reconfiguration. Here, a scheduled or unscheduled PTS outage can cause the BTS to become the CTS.

You can also configure the other System z10 EC, System z9 EC, or System z9 BC servers in the CTN to use the NTP server as an ETS.
The following are minimum requirements for using NTP:

- Server or coupling facility: System z10 EC, z9 EC, or z9 BC (Driver 67L).
- HMC driver upgrade — HMC V2.10.0 (Driver 73G) on System z10 or HMC V2.9.2 (Driver 67L) on System z9 EC or z9 BC.
- NTP server and LAN connectivity. The implementation of SNTP client on the support element supports NTP servers V3 or V4, SNTP servers V3 or V4, and IPv4 or IPv6 (System z10) or IPv4 (System z9).

The NTP or SNTP server can be a vendor-supplied external device or a local UNIX or Linux workstation running the NTP or SNTP server software and using an Internet time source, for example.

Implementation steps

With these minimum prerequisites met, you are ready to configure a CTN that supports NTP clients.

Assuming that the starting configuration is an STP-only CTN with one or more STP-enabled servers, here are the steps for configuring the STP-only CTN:

1. Establish connectivity between an NTP server and the System z10 or System z9 servers that you plan to use as the PTS and BTS.
2. Specify the CTN ID on each server or coupling facility to be configured.
3. Configure one or more NTP servers as the external time source (ETS).
4. Initialize the time, which includes time zone, leap seconds, and date and time. If you configure an NTP server, initialize the date and time using the ETS option. In addition, you must enter the time information on the server that will be the Current Time Server (CTS).
5. Assign the standard CTN roles (PTS, BTS, and arbiter).
6. Verify the configuration.

For more information about NTP and time synchronization, see these links:

- NTP Public Services Project: [support.ntp.org](http://support.ntp.org)
- NIST Web site: [tf.nist.gov/service/its.htm](http://tf.nist.gov/service/its.htm)
- Time reference servers for other parts of the world: [ntp.org](http://ntp.org).

Now is the time to let NTP client support help you extend the time synchronization functionality of STP to the heterogeneous platforms (both System z and non-System z) in your enterprise.

---

The NTP client support provided by STP allows you to meet the requirement of accurate time across heterogeneous platforms.
With a little help from your friends

Enabling multi-user access for the IODF

BY FRIEDRICH BEICHTER

While Hardware Configuration Definition (HCD) allows multiple users to read the same I/O definition file (IODF) simultaneously, this capability has not exactly facilitated the sharing of I/O configuration updates around your installation.

That’s because the process of updating the IODF has long been a solitary act, limited to one HCD user at a time. If someone on your team edits the IODF, everyone else is blocked from accessing this important file, even just to read it. This restriction applies even when your team needs to work with different, unrelated parts of the IODF.

Many HCD users at your installation might hold a stake in the contents of the IODF, but when it comes to actually updating this file, only loners need apply for the job.

That is, until we did away with the single-user limitation in z/OS V1R10. Now, HCD provides the ability for multiple users to read and update an IODF at the same time.

With IBM now recommending that you use one IODF for your entire installation, why not take this opportunity to modernize your procedures for I/O configuration work? Divide this work among several members of your team and get the changes done that much faster.

I me mine

The single-user access limitation for the IODF is not a problem if your installation has a small I/O configuration, and you are the only person responsible for maintaining the I/O definition.

However, if your IODF encompasses the I/O configurations for multiple systems, sysplexes, or remote sites, your installation would probably benefit from being able to share write access to the IODF. Here, the single-user access limitation can be a pain when your installation needs to carry out many configuration changes at one time.

From us (HCD development) to you: Don’t carry the world upon your shoulders.

Altogether now

With z/OS V1R10, HCD allows you to enable an IODF for read and update by multiple users at the same time.

Of course, this capability would be useless unless HCD provided it in a secure manner, which it does. HCD uses its own locking system to guarantee consistent updates of the data.

HCD locks the IODF just long enough for a change to be performed, that is, from the time the HCD user initiates an update action until the moment when the update is written to the IODF copy on DASD. If one HCD user attempts to update the IODF while another user is editing the file, HCD displays an informational message like the one in Figure 1 to indicate who is holding the lock.

If you happen to be a user in contention for the IODF, you might experience a delay of a few seconds before the message appears. HCD first makes several attempts (at one second intervals) to obtain the IODF for you before displaying the message. In many cases, the IODF will become available to you before it is necessary for HCD to issue the contention message.

It won’t be long

Enabling the IODF for multi-user access (MUA) has a small price in performance. Update actions will take slightly longer to complete, due to the locking and refresh actions that HCD performs. Hopefully this penalty will be more than offset at your installation by a shorter overall definition period for I/O configuration updates.

As always, it’s best to spend the time on planning and communication with your team to avoid or minimize contention for the IODF during I/O configuration changes.

Figure 1. Message CBDA340I indicates an IODF update in progress
All you’ve got to do
You can enable any of your work IODFs for multi-user access. On the HCD panel called Change I/O Definition File Attributes, specify yes for the multi-user access attribute. See Figure 2.

Get back
You can switch the IODF back to single-user access mode at any time. If you do, processing occurs for this file as done previously. That is, only one person can update IODF or multiple users can read it. In this mode, there is no additional locking of the IODF, thus no performance impact.

Any time at all
With IODF updates now considered a team effort, you might wonder how you can be sure that your view of the configuration data is the most current at any particular time.

Here is what you need to know:
- If you access the IODF for browsing only, you will not see any in-progress updates that other users might be making to the file at the same time.
- If you request an update on the IODF, HCD refreshes the in-storage copy of the data to include any changes that other users have made.
- If you initiate an update action on the IODF, HCD verifies that the device with which you are working is still available. If not, HCD informs you through a message and then refreshes its in-storage device list.

Switching an IODF to MUA is still a one-person job — that is, you need exclusive access to the IODF to make the switch. In addition, to access the IODF in MUA mode, your HCD users must have RACF ALTER access for the IODF.

At any given time, you can see which IODFs are enabled for MUA through the View I/O Definition File Information panel in HCD and the IODF dialog of Hardware Configuration Management (HCM).

Enabling an IODF for MUA requires that you have HCD for z/OS V1R10. If your sysplex includes z/OS systems earlier than V1R10, you need coexistence APAR OA22842 (PTF UA90385 for FMID HCS7720 and PTF UA90386 for FMID HCS7740).

Figure 2. HCD panel to change the multi-user access setting

For more information about enabling an IODF for multi-user access, see z/OS V1R10 Hardware Configuration Definition User’s Guide, SC33-7988, or go to ibm.com/servers/eserver/zseries/zos/hcm.
“Don’t mess with my stuff!”
Tape devices made unavailable to the system

BY STEVEN B JONES AND GIRIJA VARANASI

Don’t mess with my stuff!

How many times have you heard a teenager say that about his or her prize possessions? And it’s not just teenagers. We all want to make sure that what we own isn’t “reused” without our permission.

Recovery allocation, the processing behind messages IEF238D and IEF877E, is an important aspect of z/OS device allocation that helps to complete allocations that cannot find suitable devices. One way it does this is by bringing eligible offline devices online. But a system programmer can ask that the system not “mess” with an offline device by tweaking the right parameters on an allocation request to keep the device “off limits.” You do have a choice to let allocation either consider or not consider making use of those offline devices.

Taking devices offline—is it enough?
The options to use ALL or NO offline devices might be good enough most of the time. However, this all or nothing approach is not a good idea when the installation needs some of the eligible offline devices to stay offline and the rest to be available for use by allocation. There might be several reasons for this, for instance, when devices are being serviced, or when devices shared among systems have to be restricted to a few systems. Whatever the reason, it hasn’t been easy to make the system use some eligible devices and leave other eligible devices offline.

Too much of a good thing...
As part of its processing, recovery allocation lists all eligible devices in message IEF877E, and puts them in separate categories, like inaccessible devices, devices in offline libraries, and so on. Undoubtedly, this information is very valuable to the operator who is trying to select a device. But with increasing numbers of devices in libraries and esoterics (installation-assigned names for a set of devices, intended to be used when requesting a device), the device list can become too long and confusing to be useful. Wouldn’t it be nice to leave some of the listed devices out of the list, like those that are not connected to the system?
So what exactly do system programmers need?
System Programmers need tools to tell the system which offline tape devices not to use. They should also be able to undo their directive and make those devices available again. Don’t forget the new tools have to look familiar, be easy to use, come with necessary diagnostics, and not cause damage.

System programmers now have the power
With the z/OS V1R10 mark tape devices unavailable support, we gave system programmers more control in marking specific offline tape devices ‘UNAVAILABLE’ to the system. Of course, we also gave them the power to reverse their decision, and make devices available to the system. All they need to do is simply ‘command’ the system to mark specific devices as available or unavailable.

Our new, carefully crafted magic wands
The VARY DEV,UNAVAIL / AVAIL commands (in all their usual flavors, like the device range, the comma-separated list of devices) allow you to set and reset a new UNAVAIL state for a device. All offline, non-JES3, library, and non-library tape devices can now be marked unavailable with the VARY UNAVAIL command. When a device becomes online, it also made available.

You can display the new status using the current D U,,DEV and the new D U,TAPE,UNAVAIL,DEV commands. When the VARY command completes, the system issues new messages.

We serialized the new commands properly so that devices are not allowed to be unavailable and online, or to be marked unavailable while allocation is considering them.

System components forbidden from using devices marked unavailable
All those system functions that can bring an offline device online to complete an allocation will now ignore any device in the new “unavailable” state. For example, DDRSWAP will not choose an unavailable device itself, and will reject an operator-initiated swap, if the “swap-to” device is marked unavailable. Recovery allocation will not use unavailable devices while building the list of eligible devices for message IEF877E.

Device allocation will not use unavailable tape devices. Demand requests that specify UNIT=(devnum) and ask for an unavailable device fail because the request leaves no alternatives to consider. Non-demand requests will not add unavailable devices to the list of eligible devices.

Online, Offline, or “Don’t even think of using this device!”
More flexibility keeping devices offline
System programmers finally have a way to take a device offline, keep it offline, and keep it from cluttering up the lines of the IEF877E message. (On typical systems, the list of devices presented by IEF877E might be a lot shorter.) This should result in fewer operations errors and can improve the usability of the IEF877E and IEF238D messages

So stop worrying about your offline devices with these new z/OS V1R10 enhancements to device allocation. You have the power of choice.

Find out more
For more information, see z/OS MVS System Commands, SA22-7627.
Remember when it took hours to run an IPCS GRSTRACE or GRSDATA report to diagnose an ENQ serialization problem such as a deadlock? You entered IP VERBX GRSTRACE or IP GRSDATA from the IPCS command line and traveled to the cafeteria to get a cup of coffee. Next, you walked back to your office and waited a little longer. Finally, control returned to the terminal. All right, phase one was done, but the wait didn’t end there. You attempted to locate your resource of interest with a find command. Nevertheless, given the prior inability to filter on ENQ data, the find command also took an inordinate amount of time to complete. Are you done with that coffee yet? You might as well get another.

Say goodbye
Those days are history. The GRSDATA and GRSTRACE reports have been modernized to support highly flexible filtering capabilities including time range, SYSNAME, QNAME, RNAME, SCOPE, JOBNAME, and contention, among others. For example, if you are interested only in a specific QNAME or JOBNAME, you could enter IP VERBX GRSTRACE ‘QNAME(MYQNAME)’, or IP VERBX GRSTRACE ‘JOBNAME(MYJOBNAME)’, or combine the filtering attributes, such as IP VERBX GRSTRACE ‘QNAME(MYQNAME) JOBNAME(MYJOBNAME)’. Additionally, wildcard characters (?) and *) are acceptable for the SYSNAME, QNAME, RNAME, and JOBNAME keywords.

In addition to filtering, GRSTRACE now supports two different types of reports: summary and detail. The default summary report displays only contextually relevant ENQ user information identifying the owners and waiters of a resource; namely, the SYSNAME, QNAME, RNAME, JOBNAME, SCOPE, ASID, and TCB address. To complement this information, important ENQ request, contention, and grant times are displayed in the requested local, GMT, or UTC format. Now, you can easily identify how long a requester has owned, or waited for, a resource.

The detail report formats a lower level of ENQ information related more closely to internal processing for global resource serialization (GRS). Here are some of the fields displayed in the detail report:
• Modifications by GRS dynamic exits
• Addresses of internal control blocks
• PSW of the ENQ caller
• Address of the requesting TCB (when different from the owning TCB because of a directed ENQ).

GRSDATA panel
Who wants to remember this line command syntax for every GRSTRACE and GRSDATA query? Stop worrying. You don’t need to remember anymore. GRS now includes a new report panel that supports every line command query and can be used to invoke either report. This panel also generates line command syntax that you can copy for later reference. The new GRSDATA panel is available using IPCS option 2.6. (See Figure 1.) Note that the panel is always bypassed when using the direct line command syntax. A help panel is also included to explain the filter keywords, use of wildcards, and the difference between the GRSDATA and GRSTRACE reports.

Why two different views?
The GRSDATA report uses SDATA=GRSQ records. The GRSTRACE report uses GRS internal control blocks from the GRS address space and includes diagnostic data and configuration information about GRS.

When GRS is in star mode, GRSTRACE can show only requests from the local system. To see information that includes global resources from other systems, use the GRSDATA report. The amount of data included depends on the GRSQ setting of the local system. The GRSQ setting can be specified through the GRSCNFx parmlib member and changed dynamically using the SETGRS system command.

Additional latch requester information
The data returned by the DISPLAY GRSC,C system command has also been enhanced for latches to include the work unit address (TCB address for task mode requesters and the WEB address for SRB mode requesters) and the amount of time for which the latch has been in contention. The work unit address might be helpful in determining the cause of the contention and can be used to alleviate this contention through available system services. The contention times are helpful in distinguishing long-term contention from latches that are often in contention for short periods. Figure 2 shows sample output.
Figure 1. GRSDATA panel

LATCH SET NAME: LAR25SET1
CREATOR JOBNAME: GRJLAR25 CREATOR ASID: 0029
LATCH NUMBER: 0
REQUESTOR ASID EXC/SRH OWN/WAIT WORKUNIT TCB ELAPSED TIME
GRJLAR25 0029 EXCLUSIVE OWN 006E6CF0 Y 00:00:29.407
GRJLAR25 0029 SHARED WAIT 006E6B58 Y 00:00:14.098

Figure 2. Output from DISPLAY GRS,C
Feeling excluded? Introducing a GRSRNL=EXCLUDE migration

With global resource serialization (GRS), you can change the scope of ENQs through a parmlib-driven interface known as resource name lists (RNLs). A special option of GRSRNL=EXCLUDE is available at IPL time that instructs GRS to demote most global (scope=SYSTEMS) resource requests to local (scope=SYSTEM). Only requests that specify RNL=NO on the ENQ or ISGENQ macros are allowed to stay global. A major restriction of this option is that it has always required a sysplex-wide outage to migrate out of the environment. However, for some installations looking to use standard RNLs, this isn’t a viable option.

In z/OS V1R10, and back to z/OS V1R8 with APAR OA22578, GRS has provided an alternative to a full sysplex-wide outage. Using this new function, the migration can take place with a single GRS star mode image still active. To initiate the migration, use the existing SET GRSRNL=xx command. Previously, the command returned with a message stating that the function was not available in the GRSRNL=EXCLUDE environment. Now, GRS issues a confirmation message, ISG880D, to which you must reply before the function proceeds. After receiving this confirmation, GRS changes its internal queues to the new configuration and automatically issues an ISGLOCK lock structure rebuild to ensure that the queues and lock structure are coordinated. GRS then issues message ISG211I stating that the RNL change has completed. At that point, additional systems can be brought back in to the sysplex.

Deadlock getting you down?

GRS STAR users have been converting RESERVEs to global ENQs. However, getting the RNLs right can be time consuming. Luckily, the z/OS V1R10 ENQ/RESERVE/DEQ monitor can now collect only non-converted RESERVE requests. This goes a long way toward finding what needs to be converted.

Find out more

Find out more about all these new functions:

- GRSQ setting information in z/OS Hot Topics, Issue 16, “GRS at your command”
- "Migrating from GRSRNL=EXCLUDE to standard RNLs" in APAR OA22578 and z/OS V1R10 MVS Planning: Global Resource Serialization, SA22-7600
- z/OS MVS Diagnosis Reference, GA22-7588
- z/OS: MVS Interactive Problem Control System (IPCS) Commands, SA22-7594
In z/OS V1R10, system logger allows log stream duplex updates to take effect after a user-managed structure rebuild dynamically, that is, without the need for a sysplex-wide log stream disconnection. Allowing duplex updates to happen dynamically is more convenient for systems using log streams that are based on a coupling facility (CF) structure. With this new capability, you can easily change any combination of the log stream duplex keywords.

To enable data recovery, system logger maintains a duplex copy of CF log stream data, which can be stored in DASD staging data sets, local buffers in data spaces, or an additional CF structure in a system-managed configuration. When a request to update log stream duplex attributes becomes pending because of existing connections, pre-z/OS V1R10 systems require the log stream to first be disconnected on all z/OS systems in the sysplex. After the log stream is reconnected on each system, the pending duplex attributes take effect.

In z/OS V1R10, log stream duplex changes no longer require an application outage, thus making it easier for you to test duplex configuration changes and put them into production.

Simple example to start using staging data sets

The first step in using this new function is to change the log stream duplex option for a simplex-mode CF structure from local buffers to staging data sets. To use local buffers, the log stream definition initially specifies STG_DUPLEX(NO). To always use staging data sets instead of local buffers, run the IXCIAPU utility and update the log stream definition to specify STG_DUPLEX(YES) DUPLEXMODE(UNCOND).

After this update, the attribute is marked pending in the logger couple data set (LOGR CDS) while the log stream is connected. To apply the update, initiate a user-managed rebuild of the CF structure, either through an explicit SETXCF START command or internally from the cross-system extended services (XES) component of MVS or system logger.

When the rebuild is complete, the updates are no longer marked pending and are fully committed. The new attributes are used on each system, which means that your log stream is now duplexing to staging data sets as you requested (see Figure 1).

START command or internally from the cross-system extended services (XES) component of MVS or system logger.

When the rebuild is complete, the updates are no longer marked pending and are fully committed. The new attributes are used on each system, which means that your log stream is now duplexing to staging data sets as you requested (see Figure 1).

Figure 1. Using staging data sets
Changing duplex-mode log stream structures to conditionally duplex to local buffers

Now assume that you want to change the log stream duplexing scheme from unconditionally using logger staging data sets to conditionally duplexing to staging data sets or local buffers. However, in this case, the log stream is in a system-managed duplex-mode CF structure.

Initially, the log stream definition specifies the STG_DUPLEX(YES) and DUMPLEXMODE(UNCOND) attributes. To make the change, run the IXCMIAVU utility and change the log stream definition to specify DUMPLEXMODE(COND). This update is initially marked pending.

Because the log stream is already in a duplex-mode structure, you must transition to simplex mode before initiating the user-managed structure rebuild. First, stop the system-managed structure processing. Then initiate a user-managed structure rebuild.

After the rebuild is complete, the update becomes fully committed, and the log stream duplex copy is done conditionally to either staging data sets or local buffers. For the example in Figure 2, let’s assume that the CF structures are independent of composite failure, the systems are isolated, and the CF structures are nonvolatile. In this case, the logger conditionally duplexes to local buffers.

Finally, put your structure back into duplex-mode, if necessary, per your Coupling Facility Resource Manager (CFRM) policy.

Extra, extra, read more about it

With this new capability, you can easily change any combination of the log stream duplex keywords, as highlighted in our two examples. For more details, see the following IBM publications:

- z/OS MVS Setting Up a Sysplex, SA22-7625
- z/OS MVS Assembler Services Reference, Volume 2 (IARR2V-XCTLX), SA22-7607.

Read up on it, then try it — you’ll like it!

Acknowledgements

Thanks to Andrew Sica and Peter Redmond of z/OS system logger development for their contributions to this article.

---

**Figure 2. Conditionally duplexing to local buffers**

**Steps:**

a) Update the log stream definition:

   UPDATE LOGSTREAM DUMPLEXMODE (COND)

b) Stop the duplexing rebuild:

   SETCF STOP, REBUILD, DUMPLEX, STRNAME=aname, KEEP=OLD
c) Start a user-managed structure rebuild:

   SETCF START, REBUILD, STRNAME=aname
d) Restart the duplexing rebuild if necessary:

   SETCF START, REBUILD, DUMPLEX, STRNAME=aname

---

**Before**

Logger writes to duplex structure and staging data sets.

**Steps:**

- Structure A
- Structure B
- z/OS Logger
- z/OS Staging data sets

**After**

Logger writes to structure and local buffers. (Structure A is isolated from failure and nonvolatile.)

**Steps:**

- Structure A
- Structure B
- z/OS Logger
- Local buffers
- z/OS Staging data sets
In this article, we discuss the IBM recommended procedures for upgrading or replacing a CF in a Parallel Sysplex and provide tips on how to avoid some common pitfalls. These procedures assume that your configuration satisfies related IBM recommendations, such as having:

- Redundant failure-isolated CFs in structure preference lists and available with sufficient capacity (such as storage, processors, and links)
- Redundant failure-isolated XCF signaling paths.

Otherwise, you might need to consider taking down your Parallel Sysplex during the CF outage and bringing it up with a freshly formatted CFRM couple data set. Remember, CF upgrade and replacement might also require you to do structure resizing beforehand.

Start by logically removing the CF from your configuration:

1. Define a new CFRM policy that is a copy of the currently active CFRM policy, but without the definition of the CF that is being replaced. If the hardware identification will be different for the upgraded or replaced CF, just replace the CF definition. If the hardware identification is not changing, remove the CF definition and remove the CF name from structure preference lists. Be sure to use a new name for the new policy and retain the current policy, so you can restart it if necessary.

2. Record the output of the RO *ALL,D CF,CFNAME= system command. CHPID information will be needed later to take channel paths offline.

3. Use the SETXCF START,POL,TYPE=CFRM,POLNAME= command to start the policy defined in the first step. The CF will become delete pending and structure allocation will be prevented in the CF. This action does not affect the use of structures already allocated in the CF.

4. Remove all structures from the CF. The structures are listed in the output of the D XCF,CF,CFNAME= command. A single command might be all that is necessary: SETXCF START,REALLOCATE. REALLOCATE is the easiest way to move structures out of a CF that does not permit allocation, but it only works for structures that can be moved through a rebuild process. For example, you might need to provide dump data sets for structure dumps of structures pending deallocation. This is an important step. Special cases like this will need to be handled before the CF can be logically removed.

After logically removing the CF from use, remove the CF from a physical perspective:

1. On all systems, unconditionally take all channel paths to the CF offline using the CONFIG CHP system command and CHPID information that was recorded earlier. Besides being a good practice, this step serves as verification — unconditionally taking paths offline will not cause a loss of connectivity to the CF if a CF structure connector is using it.

2. Use the SHUTDOWN coupling facility control code command to shutdown CF operation. This command also provides verification — the CF is not shut down if structures are still allocated in it.
3. If needed, activate a new IODF. Note that SOFT=VALIDATE is a requirement in all N-1 partitions when you make changes to a CF element.

After the upgraded/replaced CF is running, bring it into operation:

1. On ALL systems, bring all channel paths to the CF ONLINE. Use the RO *ALL,D CF command to verify there is connectivity to the CF.

2. If the CF hardware identification did not change with the upgrade or replacement, re-start the original policy. Otherwise, the active policy should already have the new CF defined according to the policy update described in the very first step.

3. Use SETXCF START,REALLOCATE to perform rebuild processing to allocate structures according to the CFRM policy and into the new CF when appropriate.

If you are doing non-concurrent CF maintenance that does not require CFRM policy changes, use maintenance mode as an alternative to making CFRM policy changes.

Some common pitfalls and how to recover...

Take care using the CFRMPOL keyword in COUPLExx.

When there is a CFRM policy active from a previous IPL, as indicated on the CFRM CDS, the CFRMPOL keyword is ignored. To use this method to specify a new policy, you needed to have stopped the previous active policy through the SETXCF STOP,POLICY,TYPE=CFRM command or IPL the sysplex with a freshly formatted CFRM CDS.

If you were relying on CFRMPOL to start a new policy identifying a new CF, you might experience a WAIT0A3 when GRS cannot connect to the ISGLOCK structure. To get around this, IPL specifying GRS=TRYJOIN. Define and activate a new policy using the SETXCF START,POL,TYPE=CFRM,POL NAME= command.

After verifying that there are no policy changes pending and that the new CF has come into use, switch back to GRS STAR mode with the SETGRS MODE=STAR command.

OUCH! Don’t forget to remove structures from CF before replacing

Don’t leave structures allocated in a CF that you are removing. Doing so results in pending policy changes when you activate a new CFRM policy that does not define the old CF. You cannot remove the old CF from use from a software perspective until the structures are deleted. And, you might not be able to bring the new CF into use.

To correct this situation, you can reconnect to the old CF, and then use the SETXCF FORCE command to delete the structures. Or, you can do a sysplex-wide IPL with freshly allocated and formatted CFRM CDS with new policy definitions. Specify the new policy on the CFRMPOL keyword in the COUPLExx SYS1.PARMLIB member.

If the new CF comes into use without a problem, the deallocation pendings for the structures in the old CF are just “noise” and will not prevent the allocation of a new instance of the structure in the new CF.
What if CHPIDs to the CF are not operational or do not come online?

- Channel paths might need to be configured online. If the CHPIDs were not configured offline before the cables were disconnected from the CF, the results are unpredictable. To correct this problem, enter the command CONFIG OFFLINE followed by CONFIG ONLINE from the z/OS console or a CONFIGURE OFFLINE and ONLINE from the CF console.
- If dynamic changes were made, verify the IODF is correct. From a Hardware Configuration Definition (HCD) session, choose option 2.10 to run a processor type report against the new IODF. Verify that the CHPIDs that are connected to the partition and the processor type indicated on the TPATH statements match what is defined in the active CFRM policy.

Check for a hardware or cabling issue.

To verify the cable connection from a hardware perspective, from the HMC:

- Locate the image that owns or shares the channel path, and then right click on the image.
- Select Chpids menu option from the pop-up menu, and select the target CHPID.
- Open Chpid Operations from the task list, and choose Channel Problem Determination from the list
- Select Analyze Channel Information for details.
- Verify the status, node and flag information presented. If you have problems, work with the CE to resolve them.

Message IXC518I indicates CF was not in the CFRM Active Policy

The CF manufacturing code and plant, serial number, and partition specified in the policy must exactly match the hardware. If there is a typo, the CF will not come into use from a software perspective, even if it is physically online. Compare the node descriptor information that is returned from the hardware in the output from a D CF,CFNAME= command with what is specified in the CFRM policy. You can easily obtain the CFRM policy specifications using the D XCF,CF,CFNAME= command.

Find out more

We hope this article helps your CF install or upgrade go smoothly. For more information, see the "Coupling Facility Guidelines" topic in z/OS MVS Setting up a Sysplex, SA22-7625 and important updates at http://publibz.boulder.ibm.com/zoslib/books/oa18160.txt.

Figure 1. Analyze Channel Information panel
In a shared file system configuration, the sysplex root file system provides access to all the directories from all the systems in a shared file system configuration and handles redirection to the appropriate directories. Only one sysplex root file system is allowed in a shared file system configuration. If this file system is unavailable during migration or replacement, a sysplex-wide outage can occur.

Chapter 6.14 in the IBM Redbooks z/OS UNIX System Services z/OS Version 1 Release 7 Implementation, SG24-7035-01, describes how to replace the sysplex root file system without an IPL. However, it requires that all the z/OS UNIX processes cease and all file systems in the shared file system configuration be unmounted, and then mounted again after replacing the sysplex root file system. Several migration aids are provided for the migration from a hierarchical file system (HFS) to a zSeries File System (zFS), but all the migration aids require almost all of the z/OS UNIX work in the shared file system configuration to cease before you can migrate to a sysplex root file system. As a result, you cannot do sysplex root file system replacement without unmounting all the file systems in the shared file system configuration or doing a full migration to zFS file system.

Enter dynamic file replacement for z/OS V1R10

In z/OS V1R10, z/OS UNIX System Services provides a method of dynamically replacing the sysplex root file system in a shared file system configuration while it is in use without disrupting or ceasing any active workloads on the systems in the shared file system configuration. Use this method to replace the sysplex root file system dynamically or to migrate from an HFS sysplex root file system to a zFS sysplex root file system dynamically. In addition, it also allows you to switch from the HFS file system to the zFS file system, from zFS to zFS, from zFS to HFS, or from HFS to HFS sysplex root file system dynamically.

The replacement of sysplex root file system is transparent to applications and users on any system in the shared file system configuration. An MVS operator command on any system in the shared file system configuration can initiate the replacement.

The limits of magic

To support dynamic replacement without disruptions, the sysplex root file system must be a small file system consisting of directories, mount points, and symlinks. It must not contain code binaries or files. In addition, for dynamic replacement of the sysplex root file system to succeed, the system verifies the following points before the replacement.

- This method only supports the sysplex root file system in the shared file system configuration (although sysplex can be a single system).
- This method does not support dummy sysplex root as the sysplex root file system.
- The current and the new sysplex root file systems must be either HFS or zFS in any combination.
- All systems in the shared file system environment must be at z/OS V1R10 and higher.
- The sysplex root file system must be mounted as read-only at the time of replacement.

- No function shipping clients can exist for the sysplex root file system.
- Distributed File Service (DFS™) or an SMB server cannot export sysplex root or any directories in the sysplex root file system.
- Network File System (NFS) cannot remotely mount sysplex root or any directories in the sysplex root file system.
- The physical file system (PFS) of the sysplex root file system must be active in all the systems in the shared file system configuration.
- You must allocate the new sysplex root file system to contain all the active mount points and symbolic links. (This support does not copy directories, data, or symbolic links from one sysplex root file system to another, and you must set up the new sysplex root file system in a similar way to the current sysplex root file system.)
- The new sysplex root file system must not be HSM migrated, mounted, or in use at the time of NEWROOT replacement.
- You must ensure that the new sysplex root file system PFS is up in all systems in the shared file system configuration.
- The user ID (UID), group ID (GID) and the permission bits for the root directory of the new sysplex root file system must be the same as the current sysplex root file system.
- If the multilevel security (MLS) security label (seclabel) class is active and the MLFSOBJ option is active, the MLS seclabel for the new sysplex root file system must match the current sysplex root file system.
Go ahead, cast the magic!
Use the following command on any system in the shared file system configuration. The owning system of the sysplex root file system processes the dynamic replacement.

```
F OMVS,NEWROOT=New.
SysplexRoot.File.
System,COND=YES|NO
```

COND=YES means to proceed conditionally and is the default. If the sysplex root file system contains any objects more than the customary mount points and symbolic links that are in use, the BPXF245I message reports the activity and stops the command processing.

COND=NO means to proceed unconditionally. This is a disruptive operation if the sysplex root file system contains more objects than mount points and symbolic links that are in use. The BPXF245I message reports the activity but continues processing the command to replace the sysplex root file system with the new sysplex root file system. All active usage on the current sysplex root file system is broken on replacement of the new sysplex root file system; therefore, you might get an error I/O (EIO) return code. The active connections that are broken include the following:

- Any open files or directories in the sysplex root file system
- Current working directories (CWD) in the sysplex root file system (except the sysplex root CWD).

After the replacement, the root CWD(‘/’) is updated on all system in the sysplex to point to the new sysplex root file system, and then new operations go to the new sysplex root file system. Old connections in an old sysplex root file system might get EIO errors, and the system unmounts the old sysplex root file system immediately.

Mount parameters are preserved for the same file system-type sysplex root file system migration or replacement, and are dropped for different file system-type sysplex root migration or replacement. The BPXF247I message indicates that mount parameters are being dropped.

When the magic works
The following message indicates that the sysplex root file system migration is successful:

```
BPXF246I  THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED SUCCESSFULLY.
```

Consider the following points when the replacement is complete:

- You can resume normal work activities on the shared file system configuration.
- Don’t forget to update the BPXPRMxx parmlib member(s) with the new sysplex root file system information.
- Systems at a lower release level can join the sysplex.

When the magic doesn’t work
One or more of the following messages indicate that the sysplex root file system migration was unsuccessful. The reasons, return code, and reason code indicate the failure.

```
BPXF243E  FOMVS,NEWROOT COMMAND HAS BEEN TERMINATED DUE TO THE FOLLOWING REASON(S):
text
```

```
BPXF244E  FOMVS,NEWROOT COMMAND FAILED.
RETURN CODE=retcode REASON CODE=rsncode
```

Helpful charms and spells
- Use the `pax` command to copy the mount point directories, and symlinks from current sysplex root file system to new sysplex root file system. See “How to move z/OS UNIX directory trees with pax” in z/OS Hot Topics Newsletter Issue 18, February 2008, GA22-7501-14.
- For more details on the BPXF243E message reasons, and BPXF244E return code and reason code, see z/OS MVS System Messages, Vol 3 (ASB-BPX), SA22-7633-14.
- Use the `zlsow` utility on the z/OS UNIX Tools and Toys Web site at `ibm.com/servers/eserver/zseries/zod/unix/bpax1toy.html` to find the list of users or processes that are currently using the sysplex root file system before you issue the command to replace the sysplex root file system.
- Use the `fsview` utility on the z/OS UNIX Tools and Toys web site at `ibm.com/servers/eserver/zseries/` to list all the mount points and mounted file system information of sysplex root file system. You can also use `df` shell command to view this information.

The grimoires
We hope you can cast a little magic of your own. Find out more information in the following publications:

- z/OS MVS System Commands, SA22-7627
- z/OS Migration, GA22-7499
- z/OS UNIX System Services Planning, GA22-7800.
Enter the exits

A look at JES2 dynamic exits support in z/OS V1R10

BY TOM WASIK

Despite the fact that the test system was IPLed with your updated JES2 Exit 50 routine last night, it has been a quiet morning at work. So far.

Then, just before 10:00 am, your pager goes off. Having started a test run, the Application Group reports that your exit routine is abending.

A quick look through your code and you see the problem: You forgot to load the $JCT address. You fix the bug, then compile and link-edit the exit load module. The fix is ready. Now you just need to enter the following command before you can head off in search of that late morning coffee: $T LOADMOD(HASX 50A), REFRESH.

Aren’t you glad you made your JES2 exits dynamic in z/OS V1R10? So much nicer than having to restart JES2 (or relPL) to refresh an exit routine.

Dynamic exits make their entrance

The dynamic exits function of JES2, integrated in z/OS V1R10 and rolled back to earlier releases through APARs, allows you to dynamically:

- Load, refresh, or delete your installation-supplied load modules (exit modules)
- Refresh the tables within a specified load module
- Change the list of routines associated with an exit point.

To provide this capability, we enhanced the $T EXIT operator command to allow you to dynamically enable and disable exit points, and refresh the list of routines associated with exit points.

We also added the following new commands, which you can use to dynamically add, delete, and refresh installation-supplied load modules:

- $ADD LOADmod
- $DEL LOADmod
- $T LOADmod, REFRESH.

One thing that JES2 dynamic exits support will not allow you to do is change the exit modules that are part of the base JES2 product. Thus, you cannot use this function to apply IBM service.

Preparing for JES2 dynamic exits

Preparing to use JES2 dynamic exits will require you to do some “archeological investigation” of your exits. If your installation uses some code to initialize its JES2 exit routines, this code is driven only at JES2 initialization. With dynamic exits, your initialization code will not get control if an exit module is dynamically loaded after initialization.

Begin by locating the code that your installation has written to perform initialization for the exit routines. When you find it, determine what the code does and what changes are needed to enable the exit for dynamic loading.

Steps for making JES2 exits dynamic

First, you will need to determine whether there is any installation-supplied initialization code that “connects” (points to or plugs in) an exit module. You will want to revise this code so that it can cope with dynamic reloads of the exit module.

Initialization code for an exit module can vary from one installation to another, but it typically resides in these places on your system:

- At the start of the exit routine
- Exit 19 (initialization statement exit)
- Exit 24 (post-initialization exit).
Most exit modules do not require initialization code. Don’t be surprised if you find none for your exit. That’s good — these exit modules should work dynamically without modifications.

If you find initialization code that is associated with an exit, you need to determine what that code is doing (and where). If the initialization code is just obtaining some storage and setting a pointer to it in a JES2 data area (such as $USER1 or CCTUSER1), this is not a problem. You might, however, want to consider moving this code to a $$$$LOAD routine in the future.

However, if the initialization code is pointing a JES2 data area to a location in your exit module, you need to make some changes, such as moving this code to a $$$$LOAD routine. Also, consider adding a $$$$DEL routine to clear the pointer whenever the module is deleted or refreshed.

### Dealing with tables in modules

One approach is to prevent the first version of the exit module from being deleted. Here, you can use a $$$$DEL routine to check whether the pointers you set are pointing into the same instance of the module that the $$$$DEL routine resides in. If so, have the $$$$DEL routine set a return code of 8, which prevents JES2 from deleting that module. Otherwise, if the $$$$DEL call is for a later instance of the module, allow the deletion.

As an alternative, you can code DYNAMIC=NO on the $MODULE macro to prevent dynamic commands from operating on the module.

### Migration consideration

The JES2 dynamic exits functions changed the way a number of JES2 data areas work. Regardless of whether you decide to take advantage of the new functions, you must examine your exits to see if you are affected by the changes.

If you use $LMT, $XIT, or $XRT in your exit routines or in any code that references JES2 data areas, you might be affected, as follows:

- **$LMT** is a data area that represents a load module. With the JES2 dynamic exits functions, this data area can now represent a module that has been logically deleted or that is no longer in storage. Before looking at an $LMT, verify that it is valid for your purposes.
- **$XIT** is an array that represents the 256 possible JES2 exits and $XRT is an array that represents the routines associated with an exit. Some of the data in $XIT was moved to $XRT.
- **$XRT** is now a header with the data moved here from the $XIT, followed by an array of XRTEs that represent the routines associated with an exit. If your exit routine currently references $XRT, you need to restructure the routine.

### Find out more

For more information about the JES2 dynamic exits functions, see the JES2 publications for z/OS V1R10 and the documentation for APAR OA21346. ■
Finding time on the water slide at the public pool is never an easy task when everyone wants to swim on those hot summer days. While you waste your day waiting in line, work backs up and you begin to dream about having the pool all to yourself. Luckily, in the z/OS V1R10 Language Environment® your multithreaded applications no longer have to wait in long lines for heap pool cells. Heap pools is an optional storage allocation algorithm for C/C++ applications that when tuned correctly is much faster than the normal malloc() or free() algorithm. The new pool-count delimiter for the HEAPPOOLS runtime option now allows users to specify one to 255 pools of the same size where each pool is used by a portion of the threads for allocating storage. Heavily multithreaded applications that see high successful get heap requests for particular cell sizes will benefit the most from the enhancement.

If you are running a C/C++ application and want to reduce contention among multiple threads vying for pools of the same cell size, follow these tuning guidelines:

1. Run your application with the run-time options HEAPPOOLS(ON) or HEAPPOOLS(ALIGN) and RPTSTG(ON) for some time with a representative application workload. Then examine the HEAPPOOLS Statistics and HEAPPOOLS Summary sections of the Storage Report for Enclave.

2. Change the cell size in the HEAPPOOLS run-time option to the “Suggested Cell Sizes” shown in Figure 1. Use the default percentages in the HEAPPOOLS options and re-run the application with a representative application workload. Examine the storage report as outlined in Step 1, but instead of accepting the defaults use the “Suggested Percentages for current Cell Sizes.”

3. The values listed under “Suggested Percentages for current Cell Sizes” should be the optimal cell sizes and percentages to minimize storage use.

4. If there are a high number of “Successful Get Heap Requests” for a particular cell size, your multithreaded application can benefit from the new pool-count sub option. Identify cell pool sizes where many “Successful Get Heap Requests” are occurring and “Maximum Cells Used” is high; these cells might indicate excessive contention allocating elements in the cell pool and are excellent candidates for multiple pools. Determining the optimum number of pools to use for these cell sizes involves comparing performance measurements, like throughput when different pool-count values are used for a representative application workload.

Figure 1. Storage report for enclave
Figure 2 is an example of specifying HEAPPOOLS with the new pool-count delimiter.

```
HEAPPOOLS = ((ALIGN, (8, 4), 20, (16, 2), 10, 0))
```

Figure 2. New pool-count delimiter

Figure 3 is the expected output when you invoke the RPTSTG(ON) run-time option.

<table>
<thead>
<tr>
<th>HEAPPOOLS Statistics:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool 1.1 size: 8</td>
<td>Get Requests:</td>
<td>99922</td>
</tr>
<tr>
<td>Pool 1.2 size: 8</td>
<td>Get Requests:</td>
<td>99916</td>
</tr>
<tr>
<td>Pool 1.3 size: 8</td>
<td>Get Requests:</td>
<td>108245</td>
</tr>
<tr>
<td>Pool 1.4 size: 8</td>
<td>Get Requests:</td>
<td>108242</td>
</tr>
<tr>
<td>Successful Get Heap requests: 1-8</td>
<td></td>
<td>416325</td>
</tr>
<tr>
<td>Pool 2.1 size: 16</td>
<td>Get Requests:</td>
<td>51825</td>
</tr>
<tr>
<td>Pool 2.2 size: 16</td>
<td>Get Requests:</td>
<td>51825</td>
</tr>
<tr>
<td>Successful Get Heap requests: 9-16</td>
<td></td>
<td>103650</td>
</tr>
<tr>
<td>Requests greater than the largest cell size:</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

HEAPPOOLS Summary:

<table>
<thead>
<tr>
<th>Specified Element</th>
<th>Extent</th>
<th>Cells Per Extents</th>
<th>Maximum Cells</th>
<th>Cells In Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell Size</td>
<td>Size</td>
<td>Percent Extent</td>
<td>Allocated</td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>20</td>
<td>208</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>20</td>
<td>208</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>20</td>
<td>208</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>20</td>
<td>208</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>10</td>
<td>104</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>10</td>
<td>104</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3. HEAPPOOLS statistics

Specifying pool-count in HEAPPOOLS

Using HEAPPOOLS (ALIGN, (8, 4), 20, (16, 2), 10, 0) has the following results:

- The allocation of four cell pools for 8-byte cells with each pool using five percent of the heap allocation
- The allocation of two cell pools for 16-byte cells with each pool containing ten percent of the heap allocation.

Now that your application is tuned, relax and enjoy less contention and shorter wait times in one of your freshly configured heap pools.

More information

To use this function for HEAPPOOLS64, see the following:

- z/OS Language Environment Customization, SA22-7564
- z/OS Language Environment Concepts Guide, SA22-7567
- z/OS Language Environment Programming Reference, SA22-7562
- z/OS Language Environment Run-Time Messages, SA22-7566.
Do you use the Language Environment’s CEEPRMxx parmlib member? Have you ever wished for the ability to check syntax errors before activating a new member into production? Luckily, z/OS V1R10 introduces CEEPRMCC to provide more flexibility in CEEPRMxx member verification. With the new CEEPRMCC program, the Language Environment now supplies users the option to verify multiple CEEPRMxx members without the need to IPL or change system wide defaults using the SETCEE command. CEEPRMxx members requiring testing can be located in a partitioned data set extended (PDSE) outside of the parmlib concatenation, enabled with use of an optional DD statement.

Using a z/OS batch job or the TSO/E environment, users can now invoke the CEEPRMCC program displaying syntax errors for incorrectly formatted CEEDOPT, CEECOPT, and CELQDOPT runtime option groups. Using these additional member syntax verification methods, access to the operator’s console is no longer required to verify if a member passes the syntax test.

Under z/OS batch, CEEPRMCC expects the following inputs:

1. The PARM parameter. Use the PARM parameter of the EXEC job control statement to select one or more CEEPRMxx members.
   
   // PARM='CEE=(xx,yy,...,nn)'  
   
   The alphanumeric characters, xx,yy,...,nn are the suffix of the CEEPRMxx members to be checked.

2. Optional: CEEPRMCK DD statement. Use the DD statement to specify the data set where CEEPRMCC expects CEEPRMxx members to be located.

   //CEEPRMCK DD DSN=XXXX.TEST.PARMLIB,DISP=SHR

If no DD is specified, the CEEPRMCC program will use the default data set SYS1.PARMLIB (see Figure 1).

Under TSO/E, you can invoke the CEEPRMCC program through the CEEPRMCK CLIST, found in CEE.SCEECLST, using the following parameters:

1. The MEMBERS keyword parameter must always be specified, where xx are the two alphanumeric characters that specify the CEEPRMxx member suffix.

2. The DSN/DSNAME keyword is optional. If specified, you must fully enclose in single quotes when specifying the HLQ for your DSN. If no DD is allocated and the DSN/DSNAME keyword has not been specified, the CEEPRMCC program uses the default data set SYS1.PARMLIB. When both a DD is allocated and a DSN/DSNAME is specified, the DD overrides the DSN/DSNAME (see Figure 2).

---

```
//CEEPRMCK JOB <JOB CARD PARAMETERS>
().'/*********************************************************************
//| 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 CARD PARAMETERS> to meet your system        |
//|    requirements.                                            |
//| 2) Update the PARM='CEE=(xx,xx)' statement to include your |
//|    selected CEEPRMxx members to syntax check.               |
//| 3) Change the DSN= parameter to the appropriate PDS/PDSE.   |
//*********************************************************************
//CEEPRMCK EXEC PGM=CEEPRMCC,
//PARM='CEE=(xx,xx)'
//CEEPRMCK DD DSN=XXXX.TEST.PARMLIB,DISP=SHR
```

Figure 1. Complete example of CEEPRMCK job
Quick Note
If CEE.SCEECLST is not part of your program search order, you can execute CEEPRMCK with the following syntax using two single quotes when specifying the HLQ for your DSN.

```
exec 'CEE.SCEECLST(CEEPRMCK)' 'MEMBERS(R9)  DSN(''HLQ.TEST.PARMLIB'')'
```

Under successful parmlib member verification, the CEEPRMCC program produces the runtime options report for each suffix you specify as seen in the following example:

```
CEE3762I The Language Environment Parmlib checker has completed.

CEE3745I 11.14.01 Display CEEDOPT
CEE=(ME)
LAST WHERE SET OPTION
-----------------------
----------PARMLIB(CEEPRMME) POSIX(OFF)
PARMLIB(CEEPRMME) STORAGE (NONE,NONE,NONE,0)

CEE3745I 11.14.01 Display CEECOPT
CEE=(ME)
LAST WHERE SET OPTION
-----------------------
----------PARMLIB(CEEPRMME) STORAGE (NONE,NONE,20,2048)

CEE3745I 11.14.01 Display CELQDOPT
CEE=(ME)
LAST WHERE SET OPTION
-----------------------
----------PARMLIB(CEEPRMME) POSIX(OFF)
PARMLIB(CEEPRMME) STORAGE (NONE,NONE,30,3072)

CEE3745I 11.14.01 Display CEEROPT
CEE=(MS)
PARMLIB(CEERPRMS) CEEEROPT(ALL)

CEE3745I 11.14.01 Display CELQROPT
CEE=(MS)
PARMLIB(CEERPRMS) CELQROPT(NONE)
```

If errors are found while invoking the CEEPRMCC program, the system does not display the runtime options report, and error messages are written to the Language Environment message file as seen in the following example:

```
CEE3761I The following messages pertain to the call to the Language Environment Parmlib checker.
CEE3731I The following messages pertain to the system default run-time options in the CEEDOPT in CEERPRMME.
CEE3616I The string ‘NNE’ was not a valid or supported suboption of the run-time option STORAGE in this release.
CEE3762I The Language Environment Parmlib checker has completed.
```

The default DD name associated with the Language Environment Message file is SYSOUT. To change the default DD name that is associated with the Language Environment Message file, use the runtime option MSGFILE.

Now, armed with the power of CEEPRMCC, you can consider yourself safe from costly CEEPRMxx syntax errors. Further documentation you might find useful is:
- z/OS Language Environment Customization, SA22-7564
- z/OS Language Environment Run-Time Messages, SA22-7566.
z/OS Language Environment’s CEEROPT and the attack of the clones

BY CHARULATHA DHUVUR, JOHN MONTI, AND BARBARA NEUMANN

A long, long time ago in an operating system (MVS) far, far away, a common run-time environment was created called Language Environment. The environment was powerful with many configuration settings called run-time options. It was set up at its inception that a single set of run-time options would be the default for an entire interstellar installation. If an application (or set of applications) needed different defaults, they needed to be dealt with individually.

However, some clever imperial system programmers discovered that by creating clones of their Language Environment initialization modules and using a powerful tool called the Binder, multiple default options could be used in a single interstellar installation. By STEPLIBing to these clones, these applications could be conquered as a whole.

On a distant planet called z, a wise, old, wizened green mentor (let’s call him Y***) was speaking to his young z-NextGenner apprentice. Y*** said, “Clones, bad they are, fight them we must.” Young apprentice Skycoder did not understand and asked his mentor to explain further. Y*** said, “Clones, STEPLIBs they use, confusion they cause.” The apprentice thought back to his earlier training throughout the galaxy and slowly began to realize the problem.

Language Environment was made up of many modules. If only the initialization module was altered, it could get out of sync with other modules. Maintenance would become difficult to apply and manage. Errors, most difficult to debug, could occur from such an unorganized arrangement, and bring chaos wherever they were deployed.

Throughout the universe, interstellar installations could be huge, yet Language Environment only allowed a single set of default options. Some applications would be able to help themselves, but older applications would be defenseless, yet were of stellar importance to the installation. Of course, the system programmers had no choice but to STEPLIB to a clone. What was worse, this prevented some installations from using the CEEPRMxx support popular on most stars, meaning they had to continue to use the sinister ++USERMODs to install the installation defaults. What could be done to stop the proliferation of the clones?

“Y***”, young Skycoder said, “Could not CEEROPT help us?”

Y*** responded with a sad laugh, “Use it in CICS or library retention routine (LRR), we can.”

Skycoder yelled out, “Master Y***, have you not heard about z/OS V1R10? The force is with us; a virtual light saber is at our fingertips, ready to help us vanquish the clones! CEEROPT can now be used for all applications. With it, the clones can be defeated! We must simply educate the imperial system programmers to replace their STEPLIBed clones with a CEEROPT module instead. It is much easier to do, and maintenance becomes as easy as changing the oil on my land-speeder back home.”

He went on to explain to his mentor that by using CEEROPT, COBOL, PL/I, and C/C++ applications could use their own set of run-time options without re-binding or making JCL changes, if a STEPLIB already exists. And, best of all, if the clones were destroyed, CEEPRMxx could be used to control the interstellar installation’s default run-time options and the sinister ++USERMODs could also be destroyed.

Y*** said excitedly, “The teacher, become, you have...details, I want!” Skycoder went on to explain with the ease of an expert that nothing changes by default, but those installations that want to use this new support must set new keywords to the value ALL in their CEEPRMxx members:

| CEEROPT(COMPAT|ALL) | CELQROPT(NONE|ALL) |

He went on, with confidence, “The CEEROPT keyword controls all non-AMODE 64 applications. COMPAT is the default and requests the CEEROPTs only be loaded for CICS and LRR environments. ALL requests CEEROPTs be loaded for all environments. CELQROPT controls AMODE 64 applications. ALL requests a load of CELQROPT be attempted (NONE prevents the load attempt). A new CEEWQROP sample job is provided to assist with creating CELQROPTs. Of course, SETCEE and D CEE are also updated to support the new keywords.”

Y*** smiled at his young student and after a while said, “Destroyed the clones, you have; the force is strong in you.”

To learn more about CEEROPT and CELQROPT, see z/OS V1R10 Language Environment Customization, SA22-7564, z/OS V1R10 Language Environment Programming Guide, SA22-7561, and z/OS V1R10 Language Environment Programming Guide for 64-bit Addressing Mode, SA22-7569, for all the details.
Our contributors

**Bob Abrams** is a Senior Technical Staff Member in z/OS Development at IBM Poughkeepsie. He has been with IBM for 30 years. Bob’s focus is on reliability, availability and serviceability (RAS) and problem determination.

**Don Ault** has worked in z/OS Development for more than 30 years. He led the initial sysplex team, and the teams for z/OS UNIX System Services (z/OS UNIX), 64 bit virtual, and many other components. Don’s latest stint in real storage management (RSM) provided the new 64-bit virtual storage services, which are described in this issue.

**Friedrich Beichter** leads the design and development for HCD and HCM. He joined IBM at the Boeblingen Lab in 1977 and has since worked on z/OS, OS/390, z/VM®, and VSE.

**Scott Bender** leads the design and development for z/OS UNIX. He has worked in z/OS UNIX for 15 years.

**Chris Brooker** is a Staff Software Engineer in z/OS Global Resource Serialization (GRS) Development and the team lead for GRS Service. He holds an MS degree in Software Development from Marist College of Poughkeepsie, New York.

**James Cammarata** is a Senior Software Engineer for DFSMS Device Support at IBM San Jose. He has been with IBM for 24 years. An architect and software designer, James is the Development Technical Lead for the extended address volume (EAV).

**Bill Carey** is a Senior Software Engineer in the System z Software Strategy and Technology Organization. His 35 years with IBM includes time in systems management strategy, IPCS, TSO/E, ISPF, and file and print serving. Bill’s current focus is on XML strategy and design.

**Wai Choi** has worked in RACF Development for 11 years. Her main projects involve digital certificates: RACDCERT function in RACF and PKI Services. Wai is a frequent contributor to the RACF-L discussion list, specifically on issues related to digital certificates.

**Dario D’Angelo** is an Advisory Software Engineer and the team lead for IMS RAS Development. He has worked in Technical Support and IMS Level-2 Support in Europe and the United States for more than 15 years.

**Greg Daynes** is a Senior Technical Staff Member and the z/OS Installation Deployment Architect at IBM Poughkeepsie. His current responsibilities include z/OS software installation, migration, and maintenance strategies. Greg is a frequent speaker at SHARE and the IBM System z Expo.

**Noshir Dhondy** is a Senior Engineer at IBM Poughkeepsie with 35 years of experience. He is a member of the Parallel Sysplex and GDPS Development teams, and has provided GDPS technical marketing support since 1999. As an expert on time coordination, Noshir helped develop Server Time Protocol (STP).

**Charulatha Dhuvur** is a software developer in System z programming languages. She is a graduate of Rutgers State University of New Jersey.

**Justin Eastman** is a software developer for DFSMSdss at IBM Tucson, focusing on release development. He holds a BS degree in Computer Science and Engineering from Northern Arizona University, and an MS degree in Management Information Systems from the University of Arizona. Justin has been with IBM for ten years.

**John Fischer** has been with IBM for 18 years. For the last three years, he has worked in z/OS UNIX System Test. Previously, John was a developer and function tester for z/OS UNIX and was the original developer of HEAPPOOLS.

**Wai Choi** has worked in RACF Development for 11 years. Her main projects involve digital certificates: RACDCERT function in RACF and PKI Services. Wai is a frequent contributor to the RACF-L discussion list, specifically on issues related to digital certificates.

**Don Ault** has worked in z/OS Development for more than 30 years. He led the initial sysplex team, and the teams for z/OS UNIX System Services (z/OS UNIX), 64 bit virtual, and many other components. Don’s latest stint in real storage management (RSM) provided the new 64-bit virtual storage services, which are described in this issue.

**Friedrich Beichter** leads the design and development for HCD and HCM. He joined IBM at the Boeblingen Lab in 1977 and has since worked on z/OS, OS/390, z/VM®, and VSE.

**Scott Bender** leads the design and development for z/OS UNIX. He has worked in z/OS UNIX for 15 years.

**Chris Brooker** is a Staff Software Engineer in z/OS Global Resource Serialization (GRS) Development and the team lead for GRS Service. He holds an MS degree in Software Development from Marist College of Poughkeepsie, New York.

**James Cammarata** is a Senior Software Engineer for DFSMS Device Support at IBM San Jose. He has been with IBM for 24 years. An architect and software designer, James is the Development Technical Lead for the extended address volume (EAV).

**Bill Carey** is a Senior Software Engineer in the System z Software Strategy and Technology Organization. His 35 years with IBM includes time in systems management strategy, IPCS, TSO/E, ISPF, and file and print serving. Bill’s current focus is on XML strategy and design.

**Wai Choi** has worked in RACF Development for 11 years. Her main projects involve digital certificates: RACDCERT function in RACF and PKI Services. Wai is a frequent contributor to the RACF-L discussion list, specifically on issues related to digital certificates.

**Dario D’Angelo** is an Advisory Software Engineer and the team lead for IMS RAS Development. He has worked in Technical Support and IMS Level-2 Support in Europe and the United States for more than 15 years.

**Greg Daynes** is a Senior Technical Staff Member and the z/OS Installation Deployment Architect at IBM Poughkeepsie. His current responsibilities include z/OS software installation, migration, and maintenance strategies. Greg is a frequent speaker at SHARE and the IBM System z Expo.

**Noshir Dhondy** is a Senior Engineer at IBM Poughkeepsie with 35 years of experience. He is a member of the Parallel Sysplex and GDPS Development teams, and has provided GDPS technical marketing support since 1999. As an expert on time coordination, Noshir helped develop Server Time Protocol (STP).

**Charulatha Dhuvur** is a software developer in System z programming languages. She is a graduate of Rutgers State University of New Jersey.

**Justin Eastman** is a software developer for DFSMSdss at IBM Tucson, focusing on release development. He holds a BS degree in Computer Science and Engineering from Northern Arizona University, and an MS degree in Management Information Systems from the University of Arizona. Justin has been with IBM for ten years.

**John Fischer** has been with IBM for 18 years. For the last three years, he has worked in z/OS UNIX System Test. Previously, John was a developer and function tester for z/OS UNIX and was the original developer of HEAPPOOLS.
Neil Johnson is an Advisory Software Engineer for Parallel Sysplex Development at IBM Poughkeepsie. He joined IBM in 1997 after earning a BS in Electrical and Computer Engineering from Carnegie Mellon. Neil later earned an MS degree in Computer Science from Rensselaer Polytechnic Institute (RPI) while working on XES and XCF.

Nicholas Jones is a Staff Software Engineer at IBM Poughkeepsie. He has five years of experience in z/OS System Logger, including development, service, and test.

Steven B. Jones has been with IBM for 20 years, working in variety of BCP components, including GRS and XCF. He now leads the development teams for SMF, allocation, and scheduler.

Mike Koester is a software engineer for DFSMS Device Support. He has worked in Device Support Facilities Development for 18 years and has been the lead developer for 12 years. Mike has been with IBM for 26 years.

Kin Lau is a developer on the IMS RAS team. He has been with IBM for one year. Kin holds an MS degree from Stanford University.

Cecilia Carranza Lewis is a Senior Technical Staff Member at IBM San Jose. She has been with DFSMS for 25 years, and is the Technical Lead for EAV. Previously, Cecilia led numerous projects associated with sequential access methods.

Jason Luurs is a software developer at IBM Tucson. Before joining the DFSMSdss Development team, he worked in DFSMSdss Customer Support for eight years. Jason holds a BS degree in Management Information Systems from the University of Arizona and an MBA in Technology Management.

Charles Mari is a software engineer in the Systems and Technology Group at IBM Poughkeepsie. He currently works in the Real Storage Memory Management area of z/OS Core Technologies.

George Markouzios, CISSP, has contributed to RACF 1.8, B1 support in RACF 1.9, and RACF DB2 V9. He was a key developer for RACF digital certificates and PKI. George was the team lead for UNIX file system ACLs and wrote supporting code for the RACF checks.

Justin McCoy is a software engineer in the Systems and Technology Group at IBM Poughkeepsie. Since joining IBM in 2005, he has worked in System Verification Test (SVT) with a focus on z/OS UNIX and z/OS Language Environment.

Chris Meyer, CISSP, is a Senior Software Engineer in the z/OS Communications Server design team at IBM Research Triangle Park. Since joining IBM in 1983, he has spent his career in software development and design, with a focus on file systems, electronic payment systems, communications, and most recently, network security.

Robert Miller, Jr. is an Advisory Software Engineer at IBM Poughkeepsie. He is the Function Test Lead for z/OS System Logger.

John Monti is a Senior Software Engineer in the z/OS Language Environment development group. He is a frequent speaker at SHARE.

Mark Nelson, CISSP, has worked in RACF Design and Development for 20 years. His accomplishments include the RACF database and SMF unload utilities, the RACF/DB2 external security module, and RACF support for IBM Health Checker for z/OS. An award-winning speaker, Mark co-authored the book, Mainframe Security for the Security Expert: An Introduction to 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.

George Ng is a System z IT Specialist for Systems and Technology Group Lab Services at IBM Poughkeepsie. His expertise includes Parallel Sysplex and z/OS high availability and performance. George is a co-author of the IBM Redbooks STP Planning Guide, SG24-7280, and STP Implementation Guide, SG24-7281.

Lisa Novai is the Service team lead for z/OS UNIX Kernel Development at IBM Poughkeepsie. She has worked in the z/OS UNIX area for ten years.

Jun Ogata is a Staff Software Engineer with more than 15 years of experience in function testing RACF. He has also tested other z/OS components, most recently XML System Services. Jun strives to demystify the magic of RACF.

Walt Otto works at IBM Poughkeepsie, and owns several components that are “close to the iron,” including Loadwait/Restart. In his spare time, he rebuilds steam locomotives.

Steven Partlow is an Advisory Software Engineer in z/OS GRS Development and Service. He holds a BS degree in Computer Science from the University of Illinois at Urbana-Champaign.

John Paveza is a Senior Technical Staff Member at IBM San Jose. His background is in DASD control unit development. Recently, he switched gears and joined the DFSMS team working on system storage. In his spare time, he plays tennis, cycles, and enjoys life with three teenagers and his lovely wife.

Kathy Pfieffer is an Advisory Software Engineer at IBM Poughkeepsie. She is the curriculum manager for the IBM Academic Initiative System z9 Program.
Kurt Quackenbush is a Senior Software Engineer at IBM Poughkeepsie. He is the architect and team lead for SMP/E, and has been a member of that team since 1987.

Debbie Quick is a software engineer in the Systems and Technology Group at IBM Poughkeepsie.

Ahilan Rajadeva works in z/OS UNIX Kernel and File System Development at IBM Poughkeepsie.

Jan Redding is an Advisory Software Engineer for DFSMS Device Support at IBM Boulder. She has been with IBM for 24 years. Jan was a member of the Function Test team for EAV, and has extensive experience with z/OS installation and migration.

Wayne Rhoten is a Senior Technical Staff Member for DFSMS at IBM San Jose. He has been with DFSMS for 39 years. Wayne is a member of the Core Design team for EAV, and led the project for large format data sets.

Daniel Rosa is a Staff Software Engineer for the supervisor component at IBM Poughkeepsie. He has been heavily involved in the development of HiperDispatch. Dan has been with IBM for five years and holds a BS degree in Computer Science from Marist College.

Eric Rosenfeld, CISSP, joined IBM as a software engineer in 1985 and has been developing security products for 15 years. He is currently a developer for RACF and is the development product owner for z/OS Network Authentication Service.

Matthew Hank Sabins is a developer for the IOS component. His projects include development for MIDAW support, STP, and the IOS proxy for TS1120 tape encryption.

Patty Salone is a Senior Software Engineer at IBM Poughkeepsie. She has worked in Software Service for 27 years and currently specializes in Sysplex Services.

Donald Schmidt is a Senior Technical Staff Member in System z Strategy and Technology at IBM Poughkeepsie, where he is responsible for System z architecture and software design. His expertise includes HiperDispatch. Donald holds a BS degree in Computer Science from University of Shippensburg.

Ben Segal is a Staff Software Engineer in z/OS GRS Development and Service. Ben holds a BA degree in Mathematics with honors and a minor in Computer Science from Marist College of Poughkeepsie, New York.

Horst Sinram works on the design and development of z/OS Workload Management (WLM) at IBM Boeblingen. His projects include System z Systems Management and cooperative functions between hardware and software, such as Intelligent Resource Director, support of specialty processors, and z/OS Capacity Provisioning. Horst has been with IBM for 24 years.

Alan Smith is Senior Software Engineer in IMS RAS Development. His 29 years with IBM includes experience in application programming, systems programming, tools development, and IMS development.

Paula Spens is a software engineer in the Systems and Technology Group at IBM Poughkeepsie. She currently works in the Real Storage Memory Management area of z/OS Core Technologies.

Ralf Thelen is the WLM Test Team lead and the z/OS Quality Focal Point at IBM Boeblingen. He works on the design and execution of algorithm verification tests for z/OS Capacity Provisioning and WLM/SRM. Ralf has been with IBM for 18 years and holds a degree in Computer Science from The University of Bonn, Germany.

Mike Todd is a member of the IBM Academic Initiative System z and the team lead for the U.S. and Canada Master the Mainframe contests.

Elpidia Tzortzatos is a Senior Software Engineer in the z/OS Core Technologies department at IBM, where she leads the Memory Management Design and Development teams. She was instrumental in the successful delivery of several key enhancements in the memory management area, such as 64-bit real and virtual support and large page support.

Girija Varanasi is a software programmer on the Allocation Development team at IBM Poughkeepsie. Since joining IBM in 2001, she has worked in msys for setup, and the supervisor and allocation components.

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.

Laurie Ward, CISSP, is a designer and developer for RACF at IBM Poughkeepsie. Since joining IBM in 1985 (and RACF Development in 1988), she has helped to design and develop many RACF releases for both z/OS and z/VM.

Tom Wasik is the team lead for JES2 Design and Development at IBM Rochester. Before creating the JES2 dynamic exits support, Tom designed and developed NJE over TCP/IP and many of the JES2 SSIs. He has been with IBM for 24 years since completing graduate school at Syracuse University. Tom is a frequent speaker at SHARE and other user conferences.

Doug Zobre is a designer and developer for z/OS System Logger at IBM Poughkeepsie.
In the Hot Topics spotlight

The z/OS Hot Topics Web site will soon be featuring “Hot Spotlight” articles. These are exceptional stand-alone Hot Topics articles exclusive to the Web site. Six months between issues is too long to wait for some particularly hot topics. We had to find a way to get these articles to you while they were still red-hot. So come by the Hot Topics Web site in the coming months to see what we’ve placed in the new Hot Topics spotlight.

Did you know that this issue and all the past issues of z/OS Hot Topics are available on the z/OS Hot Topics Web site too? The site is a great place to catch up on any Hot Topics articles you might have missed. You can also point friends and coworkers to this site if you’ve read an article in this issue that you think would interest them. Check out the site at ibm.com/servers/eserver/zseries/zos/bkserv/hot_topics.html.

If you would like to receive e-mail notification when the Web site contains new issues of z/OS Hot Topics or Hot Spotlight articles, send us a note at newsletr@us.ibm.com.
Do you use the LookAt Web site on a daily basis? Do you use the Mozilla Firefox Web browser? Would you like to have the ability to search for message IDs available right at your fingertips?

If you’ve answered “Yes” to these questions, you will be excited to know that you can now search for message IDs in LookAt without having to take that extra step of opening the LookAt Web page. You can add the LookAt search function directly to your Firefox search bar. Doing so is a quick, two-step process.

First, install the Add to Search Bar plug-in for Firefox:

1. Open your Mozilla Firefox Web browser.
2. Copy the following Web address into your Firefox location bar: https://addons.mozilla.org/en-US/firefox/addon/368
3. Install the Add to Search Bar plug-in by clicking Add to Firefox. In the confirmation window, click Install Now.

Note: The Add to Search Bar plug-in works with Firefox 2.0 and above and needs to be installed only once.

4. After the plug-in is installed, restart Firefox to make the plug-in available.

Next, add the LookAt Web site to your Firefox search bar:

2. Right-click in the Message ID input box.
3. From the list of options, select Add to Search Bar.
4. In the confirmation window, click OK to make LookAt active on your Firefox search bar.

Now that you have installed the plug-in and made LookAt available for search on your Firefox search bar, you can test it out by typing a message ID in the search bar that is linked with the LookAt Web page.

If you have any questions about making the LookAt search function available on your Firefox search bar, use the LookAt feedback form at: ibm.com/systems/z/os/zos/bkserv/lookat/lookatfeed.html

We hope you enjoy this feature!