The fine art of z/OS

Also, updates on the Mainframe Skills programs, z/OS and Java, Migration to z/OS V1R13, and articles on V1R13 functions of note

Bill Neiman
z/OS Parallel Sysplex Development
Programming is as much an art as it is a science, and Bill Neiman, featured on this issue’s cover, communicates that wisdom. In our feature article “The fine art of sizing coupling facility structures: Techniques, tools, and tradeoffs,” Bill shares his expertise on sizing structures for the coupling facility, the heart of your Parallel Sysplex. With his years of experience on managing coupling facility resources, he offers important insights that can make your sysplex planning easier and your own sysplex a work of art.

This issue also goes back to school with a section devoted to the latest Mainframe Skills programs that have helped advance the arts and sciences of building mainframe environments for the next generation of system programmers. From IBM’s commitment to the Academic Initiative System z training to z/OS certification and mastery programs and leveraging online social media resources like Twitter and others, the future indeed looks bright for the mainframe, and you can read all about it.

We know how much you like regular updates on what’s going on in the world of storage with Stephen Branch’s “Ask Mr. Catalog”, and he’s back after a much needed hiatus. This issue introduces a new regular feature on migration with our resident experts Marna Walle and Shigeki Kimura, sharing their knowledge with heads up information and tips for making the smooth move from release to release. (See “Lucky you! z/OS V1R13 is a great place to be!” and “More z/OS V1R13 migration hints and tips from Professor Kimura.”) Be sure to look for their excellent migration advice in future issues of our newsletter.

Our latest issue, now published annually, also provides a great summary of Java functions for z/OS (call it our Java corner) with two in-depth articles “z/OS Java: Over a decade of excellence and enhancements!”, and “New and improved! Migrating to the latest z/OS Java APIs”. Both articles trace the progress IBM has made and continues to make with the latest enhancements to Java for System z.

And we haven’t forgotten about z/OS V1R13 by highlighting those functions and features we think you’ll really appreciate: for storage, VSAM RLS, catalogs, and tape allocation; for security, TR31 and interoperable key exchange, IRRXUTIL z/OS Health Checker check, and ISKLM on z/OS; tips and “how tos” on database management, and so much more. There are updates on global resource serialization support of FICON CTC devices, what’s new for InfiniBand, BCPi, Omegamon, RMF XML reports—a whole gallery to stroll full of fine functions available to you through z/OS V1R13.

So be sure to check out the host of articles in this issue to artfully handle the maintenance and management of your mainframe systems. Ooolala!! C’est magnifique.

Drop us a line at newsletter@us.ibm.com.

The Editors
Contents: Featured articles

4 The fine art of sizing coupling facility structures 22 And now! Introducing the enhanced 3270 user interface on OMEGAMON XE
BILL NEIMAN

7 Seeing BCPii with new eyes 25 achieve transformation through migration with the InfoPrint Server AFP
STEVE WARREN AND RITA BEISEL

9 How to analyze serialization delays using RMF postprocessor XML reports 28 Recycle your input: AFP transform enhancements
REGINE REICHERT AND JÜRGEN KUHN

14 Retro Ring GRS' new support of FICON CTC communication CHRISS BROOKER, BRYAN CHILDLS, AND JOE GENTILE

17 What's new with InfiniBand? 37 Put some muscle (and brains) behind your mainframes
PHIL MULLER AND GEORGE NG

19 Seamlessly integrating HiperSockets into the zEnterprise intra-ensemble data network MIKE FOX

11 Don't just emulate it! NINA GORADIA AND MIKE KASPER

14 Easy does it with DB tooling and training JEFF M. SULLIVAN AND JAVIER CHAVEZ

22 All that data got you down? How VSAM RLS can help DEBBIE BENJAMIN, ROITY PRIETO, AND TRACEE TAO

25 Ray Morse and Joan Ries

28 Ask Mr. Catalog STEPHEN BRANCH, PATRICIA CHOI, AND ROHAN KURANE

31 Lucky you! z/OS V1R13 is a great place to be! MARNA WALLE

35 Simplify your stack with IPv4 INTERFACE JEFF HAGGAR

37 Tuning up your tape: Maximizing tape allocations with SMStape ERIKA DAWSON

38 Endless opportunities: z/OS LDAP plug-ins JON FURMINGER AND SAHEEM GRANADOS

41 Seams lessly integrating HiperSockets into the zEnterprise intra-ensemble data network MIKE FOX

44 New and improved! Migrating to the latest z/OS Java APIs GINA YUAN, CLARK GOODRICH, AND MICHAEL WANG

47 Put some muscle (and brains) behind your mainframes JENNIFER ASH, KATHY PFEIFFER, AND MIKE TODD

54 How to manage your catalogs ERIC HARRIS AND RYAN WISNIEWSKI

60 Stay tuned for this important broadcast message: Implementing ISKLM on z/OS WILLIAM C. JOHNSTON

63 Wrap it up tight! ROAN DAWKINS AND RICHARD KISLEY

68 You, Me, and PMV MICHAEL LUCKS, DANIEL RINCK, AND NICOLE FAGEN

71 Simplify your stack with IPv4 INTERFACE JEFF HAGGAR

74 Endless opportunities: z/OS LDAP plug-ins JON FURMINGER AND SAHEEM GRANADOS

77 Nobody's deFault but mine LAURIE WARD AND BRUCE WELLS

80 Our contributors

83 zFavorites

83 Corrections and clarifications

Back cover: – Information, please
Determining an appropriate size for a coupling facility (CF) structure is often more of an art than a science, and getting it wrong can have serious consequences. Sizing the dozens or even hundreds of structures used in a customer shop can be tedious and time-consuming, but there are tools and techniques that can help you plan your configuration.

The basics
Before we dive into a procedural dissertation, it’s worth discussing the fundamental concepts underlying structure management.

What’s in a structure?
Any structure — list, lock, or cache — contains not only the application's data, but also additional control storage required by the CF to manage the structure. The structure objects of concern to an application include list or directory entries, data elements, lock tables, and event monitor controls. In addition, each kind of structure has a set of fixed or variable attributes that affect its size, such as whether a list structure’s entries are identified by keys, names, or neither, or whether a cache structure uses name class masking to streamline cache operations.

There are usually tradeoffs that can significantly affect how to size the structure. For example:

• System Logger structure sizes are very sensitive to how long data is to remain resident in the structure. By accepting more frequent offloads, an installation can make do with a smaller structure. On the other hand, a small structure exposes you to problems if something interferes with offload.

• Cache structures, by their nature, want to be as large as possible. In practice, you have to limit the size and accept some amount of reclaim activity, in which the CF reclaims infrequently-used directory entries to make room for new ones.

• Locking applications use a hashing algorithm to assign a lock table entry to a serialized resource. If the lock table is too small, multiple distinct resources get hashed to the same lock and performance suffers because of this false contention. Sizing lock structures requires balancing lock table size against an acceptable rate of false contention, while also ensuring that there is sufficient storage for the record data used in application recovery scenarios.

Why size matters
As a system programmer, you specify sizes for structures when you define the Coupling Facility Resource Management (CFRM) policy. What if you choose poorly? Consider these examples:

• MQSeries® might reject the structure because it doesn’t have the required number of entries for the number of queue-sharing groups it supports.

• A System Logger structure might encounter a structure full condition because of a transient problem that prevents it from offloading in a timely manner. The logging application (resource recovery services or RRS, CICS, and so forth) might then fail in various inconvenient ways.

• An XCF signaling structure might not be able to provide sufficient connectivity to allow a new system to join the sysplex.

• DB2® or VSAM RLS might experience poor performance because the objects it expects to find in the associated cache structure won’t fit, and the application is forced to initiate an excessive amount of I/O to get the objects from DASD.

• GRS, DB2, IMS™, or VSAM RLS might perform poorly because attempts to access the resources serialized by the associated lock structure are delayed by false contention.

• An application might fail to start because its attempt to allocate the structure it requires fails with a mysterious “invalid structure size” indication.

Often, it’s not at all obvious that the observed problems are the result of an inadequately-sized structure. The result is a callout, a service call to IBM®, and an extended and painful period of data collection, re-creates, and conference calls.

The greatest scientists are always artists as well - A. Einstein

Sizing philosophy: some words of wisdom
Here are the two critical, if-you-take-away-nothing-else, points for sizing structures:

• There is no such thing as “the” correct size for a structure.

A structure is correctly sized when it (1) avoids problems like the ones described above caused by insufficient structure object counts, and (2) the
structure doesn't waste unacceptable amounts of CF space.

- **Bigger is almost always better.** Almost without exception, you can make your sysplex more robust and resilient and your life easier by throwing extra storage at a structure.

When planning CF capacity, you want to keep some amount of unused “white” space in each CF to allow for rebuilds of structures, particularly in the event of a failure of some other CF. Aside from that consideration, there is no reason to fine tune structure sizes too precisely. You don’t want them to be enormous, as that can cause performance issues for some applications that scan the entire structure, but you should always size your structures with a little extra contingency space.

Always re-evaluate your structure sizes whenever the following occur:

- You introduce a new workload or an existing workload grows.
- You encounter a problem attributable to structure size.
- You upgrade to a new CFLEVEL.

### The toolbox

A good set of tools is available to help you size structure. The following sections describe the main tools for sizing structures.

**CFSizer**

The CFSizer is a web-based tool that you can find at the following web site:

[ibm.com/systems/support/z/cfsizer/](http://ibm.com/systems/support/z/cfsizer/)

The CFSizer is primarily useful when you are changing an existing workload or introducing a new one. Each input page presents the structures associated with a particular application (for example, DB2 or CICS®). The inputs are designed to characterize the application workload in relevant quantities, such as write rates for log structures, or the number of stored messages for MQSeries. It’s very important to take the time to collect the data necessary to describe your workload accurately—garbage in, garbage out.

When you submit your sizing request, the browser transmits it to a back-end application running on a z/OS® system at IBM Poughkeepsie. The back-end:

1. Converts the input workload description to an estimate of the structure object counts necessary to support that workload
2. Models the fixed and variable attributes of the structure
3. Submits a size calculation request to an actual CF using the IXLCSP application programming interface (API)
4. Returns the sizes calculated by the CF to your browser for display.

For applications that permit structure alter (that is, size or ratio changes), CFSizer returns two sizes, one that corresponds to the CFRM policy INITSIZE parameter and one to the policy SIZE parameter. The INITSIZE value represents the size necessary to accommodate the workload described by the inputs. The SIZE value estimates the size to which the structure might need to grow to handle potential workload increase. In most cases, CFSizer assumes that the number of structure objects might have to double, producing a highly conservative (that is, larger than necessary) upper bound.

CFSizer deliberately simplifies the inputs and structure models to reduce complexity. However, you might need to experiment with inputs to find a range of acceptable sizes. CFSizer is a good tool for this “what if” analysis, enabling you to try various workload characterizations to see what effect they might have on structure size.

Some important caveats to consider when using the CFSizer include the following:

- CFSizer provides a starting point. It is always the installation’s responsibility to decide whether the recommended sizes are suitable in that particular shop.
- CFSizer is not formally supported; however, we make every effort to correct defects and respond to questions as soon as possible.
Some limitations:

- An installation cannot run CFSizer against its own CFs. Because CFSizer’s calculations are performed by an IBM CF, the results pertain to whatever coupling facility control code level (CFLEVEL) is installed on that CF, normally the most current level. If you are running a lower CFLEVEL than CFSizer is using, the size recommendations will tend to be larger than needed in your environment—which is fine. You would rather have structures you will grow into rather than ones that are too small.
- It’s a chore to collect the required inputs. You must examine different sources for different structures—including logs, RMF™ records, proclib members, and so on.
- CFSizer doesn’t retain the inputs, so you might want to record them for use in re-sizing the structure in the future.

SIZER utility

The SIZER utility is not the same as CFSizer (a common confusion). It is an authorized z/OS program that you download as a package from the CFSizer alternate sizing techniques page at the following web site:

ibm.com/systems/support/z/cfsizer/altsize.htm

It is most useful when you are confident that your structures are adequately sized in your current environment, you are upgrading a CF, and you want to answer the question “If structure X supports this workload now, how big does it have to be to support the same workload in the new CF?”

The SIZER utility collects CFRM policy and structure attribute and object count information for all allocated structures using the IXCQUERY and IXLMG APIs. For each structure, it uses that information to direct an IXLCSIZE calculation to every accessible CF. The output is a table of size versus CF for every structure.

The key requirement is that you have the uplevel CF installed and connected to the system running the utility, which means you cannot use the SIZER utility to plan in advance of the CF upgrade.

Like CFSizer, the SIZER utility is not part of a product and is supported on a best-effort basis, but you can also leave comments or questions through the web site.

Miscellaneous

A few other arrows in the quiver:

- **Structure-full monitoring and alter**
  CFRM monitors structures periodically to determine whether they are filling up and reports when object counts exceed a specified threshold (CFRM policy FULLTHRESHOLD keyword) through z/OS message IXC585E. Structures that support alter can then be expanded to avoid a structure-full condition, either by the operator (SETXCF ALTER command) or by z/OS (if the CFRM policy specifies ALLOWAUTOALT(YES)). Be sure to update the CFRM policy to specify larger INITSIZE and SIZE values to avoid a future recurrence.

One very important point: You cannot address this situation by defining a small INITSIZE value and a much larger SIZE value. Paradoxically, making SIZE much larger than INITSIZE actually reduces the number of usable structure objects when the structure is allocated at its initial size, and might even make it impossible to allocate it at that size. The policy SIZE value should never be more than 1.5 – 2 times the INITSIZE value.

- **Structure rebuild**
  CFRM will normally try to preserve the existing structure object counts in allocating the new instance of the structure, allowing the structure to grow if necessary. That means that it is not strictly necessary to determine in advance how big the structure will be in a new CF—you can let rebuild figure it out for you, and then update the CFRM policy to reflect the new value. Usually, though, you'll be better served by doing the analysis in advance to make sure you don’t violate the white space recommendation.

Last words

Sizing structures properly might be an art but it is worth the effort. The concepts and tools described here should help clarify your objectives and help you plan productively. Above all else, remember the fundamental rule: You want your structures to be too big to fail.

References

See Setting Up a Sysplex, SA22-7625, for information on the following topics:

- “Resource allocation in the coupling facility”
- “Defining structures in the CFRM policy”
- “Sizing System Logger structures”

See Sysplex Services Guide, SA22-7617, for information on the following topics:

- “Structure rebuild processing”
- “Altering a coupling facility structure”

Review application-specific documentation for guidance on planning for structures associated with that application. Some examples:

- IMS System Administration, SC19-3020, the topic “Data Sharing in a Sysplex Environment”

MQSeries support packs, such as MP16 “Capacity Planning and Tuning for WebSphere MQ for z/OS” at the following FTP web site:

- z/OS Communications Server SNA Network Implementation Guide, SC31-8777
Do you think it’s pretty cool that your z/OS application can identify the levels of microcode installed on a CPC (central processor complex)? How about if you can get automatic notification when a remote CPC or image (LPAR) is down? Or what if the values for an image activation profile are identified, reset (if necessary), and then reloaded to activate a partition, which can eventually load the operating system into that partition? All these and much more are available today using z/OS Base Control Program internal interface (z/OS BCPii).

Busy z/OS BCPii team
Every release since its debut in z/OS Version 1 Release 10 (V1R10), BCPii makes significant enhancements to further enable authorized z/OS applications to access more and more System z® hardware controls.

Using z/OS BCPii, an authorized z/OS applications can:

- Query CPC, image (LPAR), capacity record, activation profile, and user-defined image group information.
- Set various configuration values for the CPC, image, and activation profiles.
- Issue commands against CPCs, images, or user-defined image groups to perform minor or even significant hardware- and software related functions. For example, you can create user-defined image groups to target multiple images with a single command!

Get your free samples here
Writing your own powerful z/OS BCPii application is easier than ever, thanks to our SYS1.SAMPLIB program samples. There are two Metal C samples available to help create z/OS BCPii applications:

- HWIXMCS1 provides an example of how to use the traditional z/OS BCPii APIs and how to construct a simple z/OS BCPii application.
- HWIXMCX1 provides an example of coding a z/OS BCPii Event Notification Facility (ENF) exit to field various BCPii-registered events.

What else can BCPii provide?
Okay, so maybe writing your own z/OS BCPii application isn’t your thing. Several z/OS components, as well as some vendor applications, use z/OS BCPii internally to improve performance and availability of the system. With z/OS BCPii installed and active on your system, you can also take advantage of these opportunities. For example, using advanced z/OS BCPii technology, the sysplex failure manager (SFM) has a feature called system status detection (SSD) that improves the responsiveness of your Parallel Sysplex by automatically removing a sysplex member that has become inactive.

Capacity provisioning manager (CPM) also uses z/OS BCPii technology to potentially add and remove temporary system capacity based on your installation’s capacity provisioning policies.

Q&A for BCPii
If communications between the CPC and BCPii goes down, which I am querying or monitoring for unsolicited events, how am I notified?

z/OS BCPii uses ENF events (ENF 68) to notify you if there is either a temporary or permanent communication error between the CPC that your application is running on and the CPC that you are monitoring.

A temporary error occurs when z/OS BCPii detects a loss of communications with the CPC has occurred, but now things appear to be working. Your application can do some sanity checking to ensure everything is fine.

A permanent communication occurs when z/OS BCPii detects a loss of communications with the CPC has occurred, but now things appear to be working. Your application can do some sanity checking to ensure everything is fine.

A permanent communication occurs when z/OS BCPii detects a loss of communications with the support element (SE). Your application’s event notifications related to this CPC and all images on this CPC are stalled until the z/OS BCPii “communication available event” is received. Beginning with z/OS V1R11, install APAR OA36632 on your systems for this new “communication available event” and for additional event registration recovery enhancements.
Here are some of the newer attributes that can be queried on the CPC, image, and activation profiles.

CPC attributes:
- Current power savings mode, supported power savings mode, and power savings mode allowed (zEnterprise™ hardware only).
- Server Timer Protocol (STP) configuration data.
- Numbers of pending general purpose processors, which are:
  - System z Application Assist Processor (zAAP)
  - System z Integrated Information Processor (zIIP)
  - Internal Coupling Facilities (ICF)
- Version number of the support element (SE) console application
- Engineering code and microcode installed on the SE
- List of IP addresses associated with a particular CPC
- Indicator if auto switch between primary SE and alternate SE is enabled

Image attributes:
- Workload unit capacity for the group profile associated with the image.

Activation profile attributes (most of these attributes you can set and query):
- Activation profile description, partition ID, and profile load type
- Operating mode
- Clock type and timer offset (days, hours, minutes, and whether the offset is east or west of GMT)
- Enablement value if the activation profile must conform to the current Licensed Internal Code Configuration Control (LICCC) (zEnterprise hardware only)
- Enablement value if the partition can view the processor unit activity data for one or more other LPAR on the same CPC
- Enablement value if the partition can be used to read and write any input/output configuration data set (IOCDS) in the configuration

These attributes:
- Activation profile attributes (most of these attributes you can set and query):
  - Activation profile description, partition ID, and profile load type
  - Operating mode
  - Clock type and timer offset (days, hours, minutes, and whether the offset is east or west of GMT)
  - Enablement value if the activation profile must conform to the current LICCC (zEnterprise hardware only)
  - Enablement value if the partition can view the processor unit activity data for one or more other LPAR on the same CPC
  - Enablement value if the partition can be used to read and write any input/output configuration data set (IOCDS) in the configuration

Commands include:
- Changing the power savings mode (zEnterprise hardware only)
- Activating using the activation profile
- SCSI load and SCSI dump for booting the Linux operating system
- Server Timer Protocol (STP) commands to configure your timer network configuration

Image attributes:
- Workload unit capacity for the group profile associated with the image.

Activation profile attributes (most of these attributes you can set and query):
- Activation profile description, partition ID, and profile load type
- Operating mode
- Clock type and timer offset (days, hours, minutes, and whether the offset is east or west of GMT)
- Enablement value if the activation profile must conform to the current LICCC (zEnterprise hardware only)
- Enablement value if the partition can view the processor unit activity data for one or more other LPAR on the same CPC
- Enablement value if the partition can be used to read and write any input/output configuration data set (IOCDS) in the configuration

Bonus tip: Using image activation profiles attributes

Your z/OS BCPii application must be consistent with the Hardware Management Console (HMC) interface. For example, image activation profiles have certain specialty engine attributes available to them depending on the mode selected for the image activation profile, regardless of the specialty engines that are actually installed on the CPC.

When displaying an image activation profile, the General tab on the HMC interface shows the mode and its possible values (ESA/390, ESA/390 TPF, Coupling facility, Linux on System z and z/VSE®). Based on the value you select for the mode, only certain specialty engine attributes are displayed in HMC. For example, if the image activation profile is in Coupling Facility mode, IFLs and other non-ICF specialty engine attributes do not display.

z/OS BCPii provides a number of HWI_NUM_* attributes that you can query to determine if the engine is available to the particular image activation profile. For example, to determine if a general purpose processor (GPP) is available to an image activation profile, query these four attributes:

HWI_NUM_GPP
HWI_NUM_RESGPP
HWI_NUM_SHARED_GPP
HWI_NUM_RES_SHARED_GPP

If any of the above attributes display a value greater than zero, the GPP is available in the image activation profile, and you can query other GPP attributes in the same image activation profile.

To determine if an IFA engine is available to an image activation profile, query these four attributes:

HWI_NUM_IFA
HWI_NUM_RESIFA
HWI_NUM_SHARED_IFA
HWI_NUM_RES_SHARED_IFA

If any of the above attributes display a value greater than zero, the IFA processor is available in the image activation profile, and you can query other IFA attributes in the same image activation profile.
Are you sometimes observing performance problems on your system and any attempt to find the culprit fails even though you collected system dumps, traces, and other helpful material? There is some likelihood that these performance problems are caused by contention situations that need to be analyzed with the right tools and performance reports. z/OS V1R13 RMF is designed to help you in this matter with the new Postprocessor Serialization Delay Report.

The report provides GRS enqueue and latch performance statistics as well as system suspend lock contention information and can be used to determine which address spaces have frequently requested locks and how long they were suspended on a particular lock.

You can generate this RMF Serialization Delay Report based on performance data gathered in SMF record 72, subtype 5 using the following report option:

REPORT(SDELAY)

Because this report can only be generated in XML output format, specify DDNAME XPRPTS in your postprocessor job. DDNAME XPRPTS lets you specify an output device of your choice, which can be SYSOUT, a permanent data set, or even an HFS directory into which the report will be written:

```sh
//XPRPTS DD PATH='/u/joe/ppxml/sdelay.xml', PATHOPTS=(OWRONLY,OCREAT,OTRUNC),RECFM=VB,LRECL=256, PATHMODE=(SIRUSR,SIWUSR,SIRGRP), FILEDATA=TEXT
```

Now that you have generated a report, there are multiple ways to view the XML output.

You can transfer the XML file in ASCII mode to your Windows workstation and use the RMF Postprocessor XML Toolkit to display the report in your web browser as an HTML document. This technique makes use of style sheet files that are packaged in the include directory of the RMF Postprocessor XML Toolkit.

If you are frequently analyzing postprocessor reports and you don’t want to transfer each XML report to your workstation, think about setting up an HTTP server that runs on your z/OS system and allows you to view XML reports that reside in an HFS directory remotely from your workstation. Just make sure that the include directory of the RMF Postprocessor XML Toolkit has been copied to HFS directory /u/joe/ppxml and that an HTTP pass rule exists for the directory.

Another convenient way is to use the RMF Spreadsheet Reporter that provides built-in support to generate and browse XML reports using the
You can also see the z/OS RMF User’s Guide, SC33-7990.

When you look at the Serialization Delay report, you will see two sections:

- Serialization Delay Summary
- Serialization Delay Details.

The Serialization Delay Summary section provides system-wide summary data for all address spaces at system level by presenting total numbers for all types of CMS, local and CML locks as well as for GRS latches and enqueues.

To do a more detailed analysis, you can look at the four detail sections that provide performance statistics for those 20 address spaces that have the highest total contention time in Figure 1.

The CMS Lock Details subsection, which contains detail data about CMS, CMS enqueue/dequeue, CMS latch, and CMS SMF locks.

The CML and Local Lock Details subsection, which contains detail data about CML and LOCAL locks.

The GRS Latch Details subsection, which contains detail data about GRS latches.

The GRS Enqueue Details subsection, which contains detail data about GRS ENQ requests with STEP, SYSTEM, and SYSTEMS scope.

Note that in Figure 2, the GRS Enqueue Details subsection shows not only twenty but more than forty address spaces with serialization delay data. While some of them are only among the top-twenty address spaces for ENQ SCOPE=SYSTEM and no serialization delay data exists for STEP and SYSTEMS scope, others are among the top-twenty address spaces for STEP and SYSTEMS or even for all ENQ scopes.

The style sheet files provided with the RMF Spreadsheet Reporter and the XML Toolkit also make it possible to sort every column in ascending or descending order. When you move the cursor over a column header, and the column contains sortable data, just click the header and you can always see the address space with the highest or lowest selected value at the top of the table.

Throughout this article we have described several practicable methods how to analyze serialization delays using RMF postprocessor XML reports. XML reports are also available for many other RMF postprocessor reports which can be handled in the same way.

Think about setting up an HTTP server that runs on your z/OS system and allows you to view XML reports that reside in an HFS directory remotely from your workstation.

Figure 2. SDELAY Report - Serialization Delay Details - GRS Enqueue Details
Don’t just emulate it!

Accessing ISPF on the web

BY NINA GORADIA AND MIKE KASPER

Logging on to TSO/E and ISPF applications through the web is the dream of any z/OS application developer or system programmer who has ever been stuck at the in-laws’ during a snowstorm or has tried to work in a remote location without their usual tool set. There are much larger, practical implications too, for letting new users take advantage of existing z/OS applications with only a web browser.

Until now, the most widely used approach to accessing TSO/E or ISPF remotely was with terminal emulation software. That’s where the 3270 data stream is intercepted by an application or web client that converts it to a format for display to the user. However, the biggest disadvantage to this approach is that your data is being interpreted graphically, almost like taking a picture of your data and then deciphering the picture to get it back into something close to its original form. Why not cut out the middleman and pass underlying data structures? With z/OSMF ISPF we can.

z/OSMF ISPF

z/OS Management Facility (z/OSMF) ISPF is a z/OSMF plug-in that sends and receives JSON data using HTTPS requests. Java™ Script Object Notation is a method of representing data that is growing in popularity because it is simpler and faster to parse than XML. JSON data is sent to a z/OS UNIX™ message queue for a TSO/E address space created for the user. On the other end of the queue, TSO/E or ISPF handle the response and return messages, panels, prompts, or control data to z/OSMF ISPF in JSON format. Because ISPF and TSO/E handle the conversions to JSON, custom ISPF panels are rendered automatically too.

Unique features

When you look at the ISPF panel display, you notice several unique features of z/OSMF ISPF. One is the online “Help” link accessible in the upper-right hand corner of the page. Click this link to open a new window with help information for the z/OSMF ISPF plug-in. Another link is for “TSO Messages”. This opens up a window where you can view the history of messages you’ve received, up to 1000 at once. We will review the “Settings” window later.

The most unique feature of z/OSMF ISPF is the customizable display of ISPF screens. If you open a split screen, you can either choose a tabbed display,

![Figure 1. Example of an ISPF split screen displayed in a web browser](image)
or you can split the browser window vertically, horizontally, or both. You can see up to four screens at once. Navigation in ISPF offers more options than with 3270 too, such as clicking on links, but function keys still work in your web browser and the key map can be hidden.

An added feature of z/OSMF ISPF is that it allows up to ten parallel TSO/E user sessions at once, unlike 3270, as long as the logon procedures being used are configured to allow for ISPF profile sharing. When enabling ISPF profile sharing, you must allocate the ISPF profile data sets with a disposition of shared, rather than new. The data sets exist already, or are temporary data sets. Any 3270 users or logon procedures used must specify SHRPROF when starting ISPF, unless you have already made this the default setting. For more information, see Get all the details on page 13.

**How does this work?**

By now, you might be wondering how we do this. When you log on to z/OSMF you always have to enter a user name and password, which is verified by a SAF-compatible security product and gives your z/OSMF user session the security credentials you have been granted. The first time you start the z/OSMF ISPF application or link to it from another application, z/OSMF prompts you for TSO/E-specific user settings as shown in Figure 2.

Then z/OSMF calls the Common Event Adapter (CEA) TSO/E Address Space Manager to create a standard TSO/E address space for you just as if you had logged onto TSO/E using a 3270 emulator, except that there is no full screen logon panel because your credentials have already been established.

As you can see in Figure 2, the user can choose a logon procedure, region size in KB, account number, and group name, as well as several z/OSMF ISPF-specific settings such as screen colors and whether to prompt for user settings every time. Even if you uncheck this box, you can still change settings by using the Settings link. The initial
values are not pulled from your SAF-compatible security product, but instead are set to system default values, until a user changes them and saves them as the new z/OSMF ISPF settings. The defaults are:

- Logon procedure IKJACCNT
- ServerPac supplied logon proc
- User region size of 50000 KB
- Account number *
- No group name.

Line mode logon is used, so that logon procedure, region size, and account number are all required, but the group name is optional.

After the TSO/E user settings are provided, this information is passed to CEA from z/OSMF ISPF, and then CEA creates the TSO/E address space. To limit resources, CEA currently allows a maximum of 50 concurrent sessions. When TSO/E gets control in the new address space, it detects being called in this new type of address space, and then calls CEA to create a security environment with the user's credentials and builds a line mode LOGON command with all their settings.

CEA returns control to TSO/E and the LOGON command is parsed. The logon proc and any PARM commands run. If this launches ISPF, z/OSMF ISPF receives the ISPF panel JSON and displays it to the user. If the logon procedure results in a TSO/E READY prompt instead, z/OSMF ISPF detects this too and tries to start ISPF itself. If the logon procedure leads to another type of screen or application, z/OSMF ISPF fails with an error message explaining an ISPF session was not created for the user and asks the user to try again.

Differences between 3270 and z/OSMF ISPF

So what's the catch? Well, in some cases your system might first need to be configured to allow z/OSMF ISPF to work smoothly. For example, if you do not normally use account numbers you might need to authorize users to Account number, which is an asterisk (*) at least. By default, the ISPF task is set up to use the logon procedure IKJACCNT, which is supplied by IBM. A user can choose a different logon procedure and save it, as long as it is properly configured for ISPF. In addition, every user’s security profile requires:

- TSO/E segment
- OMVS segment
- z/OS UNIX home directory.

Any users who are only defined in the SYSLUADS data set cannot log on with z/OSMF ISPF because no enqueue on the SYSIKJUA resource can be obtained.

As some TSO/E LOGON exits are not set up to handle users whose security environment has already been established, without any enqueue on SYSIKJUA, LOGON exits might need to be modified. Exits can test the logon work area (LWA) that distinguishes users started by CEA and bypass normal processing when it is on, to avoid prompting users for security information after it has already been set up.

z/OSMF ISPF plug-in users cannot switch to READY mode from their ISPF session or use full-screen applications outside of ISPF, such as with OMVS, TELNET, GDDM®, or MVSSERV. Commands that do not work in ISPF fail here too, including TSOLIB, LOGON, or LOGOFF.

Under z/OSMF ISPF, there is no support for most VTAM® terminal macros for full screen applications, such as GTTERM or STFSMODE, but you can use GTSIZE or GETDEVSZ to get the screen size. The TSO/E session manager is not available, so logon procs using PGM=ADFMDF03 instead of PGM=IKJEFT01 fail. In addition, cross-memory TSO/E messages are not viewable, such as job notifications.

Exits

We hope that this overview helps you make the most of the z/OSMF ISPF plug-in.

See z/OSMF Configuration Guide, SA38-0652, for more details on z/OSMF.

Find more details about profile sharing in z/OS ISPF Planning and Customizing, GC34-4814.

For details about using CEA to create TSO/E address spaces which can be accessed using z/OS UNIX message queues, see z/OS MVS Callable Services, SA22-7613.

The most unique feature of z/OSMF ISPF is the customizable display of ISPF screens.
With the System z platform’s recent statement of direction to discontinue IBM Enterprise Systems Connection Architecture (ESCON®) support, Global Resource Serialization (GRS) must now provide support for Fibre Channel connection (FICON®) channel-to-channel (CTC) communication in the service stream for z/OS V1R11, V1R12, and V1R13. Major migration planning is probably necessary for one or more of your installations’ GRS complexes. It’s also possible that this support doesn’t affect your installation. It all depends on your GRS configuration. Let’s decide whether your installation is affected by this support, and if so, let’s see which migration actions are needed.

Some Ring history
The ring, the original design for serializing global resources in z/OS, still has value in today’s sysplex environment. The star, which offers many advantages over the ring complex, is the recommended configuration. If a ring configuration is required, we recommend using a sysplex and the embedded Cross-System Coupling Facility (XCF) communication links, rather than GRS-managed CTC communication. If your installation uses a star configuration or an XCF ring, you can stop reading now.

If you are using GRS-managed CTC devices (as defined in the GRSCNFxx parmlib member) because of the need for inclusion of non-sysplex members in the GRS complex, pay close attention. In the future, System z might not support ESCON, so GRS is upgrading its CTC usage to support FICON CTC communication.

Initial prep work
Regardless of the migration method chosen, there is some initial preparation:

• Apply the program temporary fixes (PTFs) for OA38230 to all the systems before an IPL of any system.
• Make sure all FICON CTC adapters, in addition to the existing ESCON basic channel-to-channel (BCTC) adapters, have the proper configuration and connection to each z/OS system in the GRS complex.
• Determine the FICON CTC adapter device numbers from the hardware configuration.
• Add the corresponding FICON CTC device numbers to the end of the existing list for GRSCNFxx members used across the complex, as shown in Figure 1.

Recall that GRS-managed CTC devices are solely for GRS’ use and cannot be added to the system dynamically. The CTC devices must be defined in GRSCNFxx at IPL time. Also recall that the order of the CTC devices listed in GRSCNFxx is significant for any given remote system. The CTC device listed last is the first CTC device considered for primary usage. Provided GRS’ initialization of that CTC device is successful, the next-to-last CTC device gets set up as an alternate, the next as an additional alternate, and so forth. If your installation is testing the GRS FICON CTC support on hardware that supports ESCON, IBM recommends adding, rather than replacing, the FICON CTC device numbers below the existing CTC device numbers, as shown in Figure 1.

There are two separate ways to migrate to the updated GRSCNFxx members with the additional FICON CTC devices, depending on the needs of the installation.

Retro-Ring migration method: rolling IPL
Migrating to FICON CTC communication means a planned outage. Because at least one system remains active in the current GRS ring throughout the migration, the advantage of performing a rolling IPL is better availability.

Use the following procedure to minimize the chance of a ring disruption in the migration of GRS-managed CTC communication:

1. Follow your normal procedures for bringing down the system and preparing to IPL. Don’t forget to enter the Z EOD command to write buffered System Management Facilities (SMF) records, and system log and LOGREC data, and wait for message IEE334I: HALT EOD SUCCESSFUL.
2. Quiesce the systems from the ring. On any system, enter the VARY GRS(X),QUIESCE command. Wait for message ISG013I: SYSTEM X QUIESCED GLOBAL RESOURCE SERIALIZATION. The VARY GRS(X),QUIESCE command is important because it prevents a ring disruption from removing the QUIESCED system. You can only use VARY GRS(X),QUIESCE when there is at least one XCF=local or monoplex system in the complex using CTC communication.

3. Stop the system. Now it’s safe to stop system X. Do so by initiating a system reset on the Hardware Management Console (HMC). For Parallel Sysplex®, the sysplex failure management (SFM) policy and the status update missing intervals (specified on the INTERVAL keyword in COUPLExx) start counting down as soon as you use the system reset.

4. Purge system from the ring. After system X stops, if it’s an XCF=local system or a monoplex, on any system, issue VARY GRS(X),PURGE. If system X is a member of a sysplex, issue VARY XCF,X,OFFLINE and respond to the prompts as usual. In either case, wait for message ISG013I: SYSTEM X PURGED FROM GRS COMPLEX, which means system X is removed from the ring.

5. IPL with a new GRSCNFxx. IPL system X. When the system is ready, issue D GRS to verify the system (and the other systems) show up as “ACTIVE” in the existing GRS ring. If the JOIN process on the existing ring fails, make sure GRSCNFxx is correct and the purge of the previous instance from the ring was successful. Try this step again.

6. Check the CTC devices. Verify that the D GRS output lists the new CTC device numbers as IN-USE, ALTERNATE, or conditionally QUIET. If the CTC device is IN-USE or ALTERNATE, the TARGET SYSNAME field must be accurate. If a CTC status is QUIET, it is online, but the other side of the CTC device is not communicating. If the system on the other side of the CTC device has not been re-IPLed to pick up the new CTC definitions, then QUIET status is expected. Scenario II in Figure 2 shows that GRS will not attempt to use a new CTC device until the system at the other side of the link is re-IPLed to pick up its corresponding CTC device. If all of the systems are running with the new CTC device, it’s unusual when a CTC device remains QUIET. If there are one or more QUIET CTC devices, look for a CTC problem on the system at the other end of the link. If the CTC device is unexpectedly “DISABLED”, look for the corresponding ISG046E message from the SYSLOG. There is probably a parmlib member or device configuration issue. After you address the issue, try to VARY the device online.

7. Repeat step 1 through step 6 for each system in the GRS ring.

**Rolling it out**

Figure 2 illustrates how the rolling IPL eventually starts using the new FICON CTC (FCTC) devices. After the first system is restarted with the original CTC and FCTC devices (scenario II in Figure 2), GRS continues to use the original CTC device and JOIN the existing GRS ring. However, as the second and third system are restarted (scenario III and IV in Figure 2), GRS uses the FCTC rather than the BCTC because, on the IPLing system, GRS always tries to initialize the CTC devices in last-to-first order as they appear in GRSCNFxx.

**Retro-Ring migration method: complex-wide IPL**

A GRS complex-wide IPL depends on the installation’s policies and procedures regarding planned outages of that scale. The preparation is the same with regard to applied PTF, FICON setup, and GRSCNFxx settings. The orderly shutdown can use the same initial steps to avoid ring disruptions.

In the case of a mixed complex, it is important for the last system to be an XCF=local or monoplex system. Whenever a mixed GRS complex transitions into a pure sysplex, VARY GRS(X) is no longer allowed. It’s also recommended that you IPL that system alone with GRS=START to establish the ring of one with all of its CTC devices QUIET before starting the remaining systems. See the TryJoin warning section that follows.

**Few differences**

There are very few external changes associated with this new function. As implied above, the CTC syntax in GRSCNFxx stays the same for FICON CTC communication. For GRS use as the initial migration path, use a combination of FICON and ESCON CTC communication.
There are a few small message updates:

- Message ISG045I is now issued with the phrase “NOT A SUPPORTED CTC” added when the specified device number, while a valid CTC device, is a serial channel-to-channel (SCTC) device rather than an FCTC device. This enhancement is strictly for FICON CTC communication.
- Message ISG046E might get issued for a hardware error with a new reason code of X’10’, which indicates the loss of paths to the CTC device. Unlike other hardware errors that can occur during GRS ring processing, GRS doesn’t vary the device offline, but instead tries to detect when paths are found and re-establish communication. GRS still considers the link DISABLED until it finds the path and a ring disruption can still occur.

**TRYJOIN warning**

Do not use GRS=TRYJOIN if any systems in the GRS complex use GRS-managed CTC communication. When the GRS complex equals the sysplex, the TryJoin option is planned to resolve correctly. But in a mixed ring, or a ring where GRS manages all of its CTC communication directly, a single system should specify GRS=START and the remainder GRS=JOIN. Otherwise, there exists a race condition for the initializing systems to form a split ring where data integrity is compromised. This is still true with FICON CTC support.

**Closing remarks**

IBM has long recommended the usage of GRS=STAR over a ring topology because of a star configuration’s superior scalability, response time, and RAS characteristics. However, we recognize that ring configurations are not going away for various reasons. The enhancement provided by APAR OA38230 allows GRS-managed CTC communication to use FICON technology, but cannot enhance the robustness of the ring configuration in general. For more information, see the documentation that accompanies APAR OA38230.

**Thanks Charlie!**

The authors are thankful to Charlie Favell in z/OS System Test for his numerous contributions to GRS FICON CTC support and his assistance with this article.

---

**The order of the CTC devices listed in GRSCNFxx is significant for any given remote system.**

---

**Figure 2. Example of a rolling IPL on a three-system GRS ring**
What’s new with InfiniBand?

BY PHIL MULLER AND GEORGE NG

To “InfiniBand and Beyond” (z/OS Hot Topics Newsletter, Issue 22, March 2010) provided an overview of the InfiniBand technology and general planning considerations for migrating to InfiniBand coupling links from earlier coupling-link technologies such as Integrated Cluster Bus (ICB) and InterSystem Channel-3 (ISC-3). Be aware that IBM is in the process of phasing out these older technologies. With the introduction of the zEnterprise System, support for ICB-4 was discontinued, so consider moving off of ICB-4 as soon as possible.

IBM System z has also issued a Statement of General Direction indicating that the IBM zEnterprise 114 (z114) and zEnterprise 196 (z196) servers will be the last to offer ordering of ISC-3 coupling links, which were first introduced in October 2000.

Figure 1 depicts an overview of coupling link technologies over time:

In July 2011, a new generation of 12x InfiniBand (12x IFB) and 1x InfiniBand (1x IFB) coupling links was introduced on the zEnterprise System, along with a new protocol — 12x IFB3. This new protocol is designed to help improve performance at short distances, that is, less than 150 meters, or 492 feet. At the same time, the number of 1x IFB coupling links was doubled for unrepeated distances of 10 kilometers or less.

12x IFB coupling links on z114 and z196

With the introduction of the third generation of coupling links, the HCA3-O fanout, it is possible to use an improved IFB protocol, 12x IFB3. This protocol is designed to provide improved synchronous service times of up to 40% faster than when using the previous version. However, the new 12x IFB3 protocol can be used only under these conditions:

- Both ends of the link must be connected to an HCA3-O fanout.
- A maximum of four channel path identifiers (CHPIDs) are defined per HCA3-O fanout for all logical partitions (LPARs) combined. For example, the 12x IFB3 protocol is used when four CHPIDs are assigned to an HCA3-O port and:
  - All four CHPIDs are shared across z/OS LPARs residing in the same sysplex.
  - Each CHPID is in use by a different sysplex.
  - CHPIDs are defined as SPANNED and are shared across z/OS LPARs in multiple channel subsystems (CSSs).

The 12x IFB3 protocol is not used under either of these conditions:

- More than four CHPIDs are assigned to the port.
- More than four CHPIDs are assigned to the port, but some CHPIDs are offline, bringing the number of online CHPIDs below five.
- Under these two conditions, the port will continue to use the 12x IFB protocol. See Figure 2.

What’s new with InfiniBand?

BY PHIL MULLER AND GEORGE NG

To “InfiniBand and Beyond” (z/OS Hot Topics Newsletter, Issue 22, March 2010) provided an overview of the InfiniBand technology and general planning considerations for migrating to InfiniBand coupling links from earlier coupling-link technologies such as Integrated Cluster Bus (ICB) and InterSystem Channel-3 (ISC-3). Be aware that IBM is in the process of phasing out these older technologies. With the introduction of the zEnterprise System, support for ICB-4 was discontinued, so consider moving off of ICB-4 as soon as possible.

IBM System z has also issued a Statement of General Direction indicating that the IBM zEnterprise 114 (z114) and zEnterprise 196 (z196) servers will be the last to offer ordering of ISC-3 coupling links, which were first introduced in October 2000.

Figure 1 depicts an overview of coupling link technologies over time:

In July 2011, a new generation of 12x InfiniBand (12x IFB) and 1x InfiniBand (1x IFB) coupling links was introduced on the zEnterprise System, along with a new protocol — 12x IFB3. This new protocol is designed to help improve performance at short distances, that is, less than 150 meters, or 492 feet. At the same time, the number of 1x IFB coupling links was doubled for unrepeated distances of 10 kilometers or less.

12x IFB coupling links on z114 and z196

With the introduction of the third generation of coupling links, the HCA3-O fanout, it is possible to use an improved IFB protocol, 12x IFB3. This protocol is designed to provide improved synchronous service times of up to 40% faster than when using the previous version. However, the new 12x IFB3 protocol can be used only under these conditions:

- Both ends of the link must be connected to an HCA3-O fanout.
- A maximum of four channel path identifiers (CHPIDs) are defined per HCA3-O fanout for all logical partitions (LPARs) combined. For example, the 12x IFB3 protocol is used when four CHPIDs are assigned to an HCA3-O port and:
  - All four CHPIDs are shared across z/OS LPARs residing in the same sysplex.
  - Each CHPID is in use by a different sysplex.
  - CHPIDs are defined as SPANNED and are shared across z/OS LPARs in multiple channel subsystems (CSSs).

The 12x IFB3 protocol is not used under either of these conditions:

- More than four CHPIDs are assigned to the port.
- More than four CHPIDs are assigned to the port, but some CHPIDs are offline, bringing the number of online CHPIDs below five.
- Under these two conditions, the port will continue to use the 12x IFB protocol. See Figure 2.
The 1x InfiniBand coupling links have been enhanced to alleviate the impact of high subchannel and link buffer usage. Driver 93 on z114 and z196 servers delivers support for 32 (rather than 7) subchannels and link buffers per CHPID on 1x IFB coupling links. Consequently, these servers can process more requests in parallel without requiring additional physical links.

Additionally, because of the increased capability to handle more concurrent requests on each CHPID, HCA3-O LR fanouts have four ports per feature rather than two ports per feature. This enhancement allows connections to more central processor complexes (CPCs) with each fan out, while still supporting more concurrent requests to each CPC.

Summary
For customers migrating to a z114 or z196 server and exploiting the high-performance ICB coupling links, the 12x IFB3 protocol delivers the performance improvements needed for synchronous communication.

Customers currently using ISC-3 links within a single data center should consider migrating to 12x IFB coupling links to experience performance improvements and to benefit from the ability to assign multiple CHPIDs to a single link.

For extended-distance solutions, the greater number of link buffers supported by 1x IFB coupling links can result in performance improvements, depending on inter-site distances. Using 1x IFB coupling links can also result in a reduction in the number of coupling links required to connect multiple sites, which means fewer fan outs and fewer dense wavelength division multiplexing (DWDM) ports.

Find out more
The 12x and 1x InfiniBand coupling links are designed to meet the coupling requirements of current and future generations of System z servers. For more information, see:

- IBM white paper Coupling Facility Configuration Options, ZSW01971USEN
- Infiband Coupling Links on System z, SG24-7539
- System z Connectivity Handbook, SG24-5444
- The Parallel Sysplex website: ibm.com/systems/z/advantages/pso/index.html

For more information about the 12x IFB3 protocol, see the following:

- IBM zEnterprise 196 enhancements deliver faster access to data: IBM United States Hardware Announcement 111-121 (July 12, 2011) ibm.com/common/ssi/rep_ca/1/897/ENUS111-121/ENUS111-121.PDF

STG z/OS Lab Services has extensive experience in delivering InfiniBand coupling link assessment and planning services. The Lab Services organization can help accelerate 12x IFB and 1x IFB installations by providing customized studies and migration plans. For more information, send an email to stgls@us.ibm.com.
Consider the intraensemble data network (IEDN), a 10 Gb Ethernet network dedicated to data communication within the IBM zEnterprise System ensemble. IEDN provides unique security and management advantages for virtual servers within your zEnterprise ensemble. On z/OS the hardware that implements this network is 10GB OSA-Express3 or later, configured with CHPID type OSX.

You can use the IBM zEnterprise Unified Resource Manager to manage IEDN. zEnterprise Unified Resource Manager lets you divide the IEDN into virtual networks and control access to those virtual networks from virtual servers and appliances that are part of the ensemble.

The zEnterprise Unified Resource Manager also interacts with hypervisors and firmware within the zEnterprise ensemble to enforce access restrictions to IEDN virtual networks, preventing a virtual server or appliance from attaching to a virtual network it is not permitted to access. To further reinforce virtual network isolation, z/OS Communications Server does not permit IP packets to be forwarded between IEDN virtual networks.

In this article, we explore the seamless integration of HiperSockets™ and the IEDN that is provided starting in z/OS V1R13.

An IEDN use case
Consider a zEnterprise ensemble that consists of multiple nodes. Each node consists of an IBM zEnterprise 196 (z196) CPC that has multiple logical partitions running z/OS attached to an IBM zEnterprise BladeCenter® Extension (zBX) that can host several blades and appliances. Also assume that this zEnterprise ensemble is hosting a three-tiered business application, in which requests from the world area network (WAN) come into web servers on blades that communicate with application servers on z/OS that in turn communicate with database servers on different z/OS systems or partitions.

Web traffic would enter the ensemble onto the web virtual network, and the web server would make corresponding requests to the z/OS application servers on the back-end virtual network. The application servers would then communicate with the z/OS database servers using the database virtual network. The application servers might communicate with either same-CPC or different-CPC database servers depending on configuration, load, and so forth.

Three-tiered virtual network
In this example, the IEDN might be divided into three virtual networks (Figure 1). The first virtual network, named web, carries the incoming traffic to the web servers and their replies to clients. The second virtual network, named back-end, is used for communication between the web servers and the application servers. Finally, the third virtual network, named database, is used for communication between the application servers and the database servers. The web servers would be permitted to the web and the back-end virtual networks, and the application and data servers would only be permitted to the back-end and database virtual networks.

Using IEDN and HiperSockets
A key concept for the security and isolation of these tasks is that the web, back-end, and database networks are all virtual networks within the IEDN, with access controlled by hypervisors and firmware that are communicating with zEnterprise Unified Resource Manager. Using the IEDN for this communication provides these security and management features.

What does this mean for the application server to database server IEDN traffic? In our example, when
To create an IQDX interface that corresponds with each OSX OSA Express interface on the stack.

The dynamically created IQDX interfaces inherit all VLAN and IP address characteristics from their corresponding OSX OSA Express interfaces. The good news is that traffic that uses HiperSockets through IQDX is subject to the same security and isolation characteristics and zEnterprise Unified Resource Manager control provided in the IEDN, without requiring additional zEnterprise Unified Resource Manager or Communications Server configuration. These new IQDX interfaces are “tucked into” the existing OSX OSA Express interfaces, as shown in Figure 2:

### How to operate it

In general you do not have to control or operate these dynamically created IQDX interfaces. Instead you operate and control your OSX OSA Express interfaces as usual, and Communications Server manages the corresponding IQDX interfaces for you under the covers. This keeps the IQDX interfaces and corresponding HiperSockets routing transparent and seamless to you.

For example, static and dynamic routes are added over an OSX OSA Express interface and not added over IQDX interface. When the TCP/IP stack routes traffic over an OSX OSA Express interface that has an associated “tucked” IQDX interface, the stack can detect that the next hop is to another virtual server on the same CPC that also has IQDX available. In this case the traffic will use IQDX and HiperSockets instead of the OSX OSA Express interface. Communications Server performs similar under the covers processing for IP Security and packet tracing.

Because this support “hides” HiperSockets within the OSX interfaces, you can now exploit HiperSockets without having to configure a HiperSockets interface to virtual servers. This feature means that when a virtual server moves from one CEC to another within the system, the IQDX interfaces can be automatically configured without manual intervention.

### How to enable it

HiperSockets is able to implement this function using a new channel function - Internal Queued Direct I/O Extensions (IQDX), which you can configure on one IQD channel path ID (CHPID) in a CPC that is a member of a zEnterprise ensemble. You can use a new channel parameter in the Hardware Configuration Definition (HCD) to configure IQDX.

If IQDX is available, z/OS Communications Server will use it by default, so once you’ve configured IQDX in HCD you don’t need to do anything else to implement this function. Communications Server dynamically creates the necessary TRLE and INTERFACE statements to create an IQDX interface that corresponds with each OSX OSA Express interface on the stack.

The dynamically created IQDX interfaces inherit all VLAN and IP address characteristics from their corresponding OSX OSA Express interfaces. The good news is that traffic that uses HiperSockets through IQDX is subject to the same security and isolation characteristics and zEnterprise Unified Resource Manager control provided in the IEDN, without requiring additional zEnterprise Unified Resource Manager or Communications Server configuration. These new IQDX interfaces are “tucked into” the existing OSX OSA Express interfaces, as shown in Figure 2:

### How to operate it

In general you do not have to control or operate these dynamically created IQDX interfaces. Instead you operate and control your OSX OSA Express interfaces as usual, and Communications Server manages the corresponding IQDX interfaces for you under the covers. This keeps the IQDX interfaces and corresponding HiperSockets routing transparent and seamless to you.

For example, static and dynamic routes are added over an OSX OSA Express interface and not added over IQDX interface. When the TCP/IP stack routes traffic over an OSX OSA Express interface that has an associated “tucked” IQDX interface, the stack can detect that the next hop is to another virtual server on the same CPC that also has IQDX available. In this case the traffic will use IQDX and HiperSockets instead of the OSX OSA Express interface. Communications Server performs similar under the covers processing for IP Security and packet tracing.

Because this support “hides” HiperSockets within the OSX interfaces, you can now exploit HiperSockets without having to configure a HiperSockets interface to virtual servers. This feature means that when a virtual server moves from one CEC to another within the system, the IQDX interfaces can be automatically configured without manual intervention.

### How to enable it

HiperSockets is able to implement this function using a new channel function - Internal Queued Direct I/O Extensions (IQDX), which you can configure on one IQD channel path ID (CHPID) in a CPC that is a member of a zEnterprise ensemble. You can use a new channel parameter in the Hardware Configuration Definition (HCD) to configure IQDX.

If IQDX is available, z/OS Communications Server will use it by default, so once you’ve configured IQDX in HCD you don’t need to do anything else to implement this function. Communications Server dynamically creates the necessary TRLE and INTERFACE statements to create an IQDX interface that corresponds with each OSX OSA Express interface on the stack.

The dynamically created IQDX interfaces inherit all VLAN and IP address characteristics from their corresponding OSX OSA Express interfaces. The good news is that traffic that uses HiperSockets through IQDX is subject to the same security and isolation characteristics and zEnterprise Unified Resource Manager control provided in the IEDN, without requiring additional zEnterprise Unified Resource Manager or Communications Server configuration. These new IQDX interfaces are “tucked into” the existing OSX OSA Express interfaces, as shown in Figure 2:

### How to operate it

In general you do not have to control or operate these dynamically created IQDX interfaces. Instead you operate and control your OSX OSA Express interfaces as usual, and Communications Server manages the corresponding IQDX interfaces for you under the covers. This keeps the IQDX interfaces and corresponding HiperSockets routing transparent and seamless to you.

For example, static and dynamic routes are added over an OSX OSA Express interface and not added over IQDX interface. When the TCP/IP stack routes traffic over an OSX OSA Express interface that has an associated “tucked” IQDX interface, the stack can detect that the next hop is to another virtual server on the same CPC that also has IQDX available. In this case the traffic will use IQDX and HiperSockets instead of the OSX OSA Express interface. Communications Server performs similar under the covers processing for IP Security and packet tracing.

Because this support “hides” HiperSockets within the OSX interfaces, you can now exploit HiperSockets without having to configure a HiperSockets interface to virtual servers. This feature means that when a virtual server moves from one CEC to another within the system, the IQDX interfaces can be automatically configured without manual intervention.

### How to enable it

HiperSockets is able to implement this function using a new channel function - Internal Queued Direct I/O Extensions (IQDX), which you can configure on one IQD channel path ID (CHPID) in a CPC that is a member of a zEnterprise ensemble. You can use a new channel parameter in the Hardware Configuration Definition (HCD) to configure IQDX.

If IQDX is available, z/OS Communications Server will use it by default, so once you’ve configured IQDX in HCD you don’t need to do anything else to implement this function. Communications Server dynamically creates the necessary TRLE and INTERFACE statements to create an IQDX interface that corresponds with each OSX OSA Express interface on the stack.

The dynamically created IQDX interfaces inherit all VLAN and IP address characteristics from their corresponding OSX OSA Express interfaces. The good news is that traffic that uses HiperSockets through IQDX is subject to the same security and isolation characteristics and zEnterprise Unified Resource Manager control provided in the IEDN, without requiring additional zEnterprise Unified Resource Manager or Communications Server configuration. These new IQDX interfaces are “tucked into” the existing OSX OSA Express interfaces, as shown in Figure 2:
ensemble, the moved virtual server can automatically start using HiperSockets to communicate with other virtual servers within the destination CEC, with no configuration changes required. The reverse is also true: if a virtual server moves out of a CEC that it was sharing with other virtual servers, it will automatically stop using HiperSockets and use OSX to continue communicating with those virtual servers.

You can use Netstat to display information about the dynamically created IQDX interfaces. They are visible through the use of the Netstat DEVlinks/-d command, that displays data counts so you can see how much traffic was routed over IQDX. The Netstat ARp/-r command (for IPv4) or the Nestat ND/-n command (for IPv6) gives you information about which IP addresses are reachable using IQDX.

**Getting the most out of IQDX for the least cost of CPU**

To help you manage your speed/latency versus processor (CPU) tradeoff with HiperSockets, Communication Server provides a HiperSockets multiple write assist with IBM zIIP function that can minimize HiperSockets usage of the general purpose processors for large write operations, and the IQDX traffic is eligible for this function.

If you do not want to use IQDX to send large amounts of data over the IEDN, you can code the AUTOIQDX NOLARGESEND parameter on the GLOBALCONFIG statement in your TCP/IP profile. If you code this parameter, the Communication Server TCP/IP stack will direct the data over your TCP/IP profile. If you do not want to use IQDX to send only smaller, request-response type traffic over the IQDX interface when this feature is enabled, which empowers you to make your own performance versus CPU cost decisions for different types of IEDN HiperSockets traffic.

You get all this flexibility and versatility without sacrificing the security and management strengths of the IEDN.

**References**

- **IBM Europe, Middle East, and Africa Hardware Announcement, ZG12-0035, dated March 6, 2012:**
  
  ibm.com/common/ssi/rep_ca/5/877/ENUSZG12-0035/ENUSZG12-0035.PDF

- **IBM z/OS Communications Server: IP Configuration Guide, SC31-8775**

- **IBM z/OS Communications Server: IP Configuration Reference, SC31-8776**

- **IBM z/OS HCD Reference Summary, SX33-9032**
The OMEGAMON® enhanced 3270 user interface is a new z/OS address space that brings Metal C and extensive 64-bit storage to the existing 31-bit VTAM session manager in the OMEGAMON product suite.

Look for the following enhancements to the OMEGAMON enhanced 3270 user interface:

- New presentation services layer that employs APL to produce a new hybrid best-of-breed user interface.
- New workspace definitions that can be cloned and eventually customized to suit your site's needs.
- Support for REXX, basic ISPF, and HTML workspace definitions that align with other IBM z/OS products.
- Use of standard SQL to drive data collection to any OMEGAMON XE agent in Tivoli Monitoring. This is significant because it means that one screen can contain data from multiple agents.

Currently, the enhanced 3270 user interface supports OMEGAMON XE on z/OS and OMEGAMON XE for CICS on z/OS products, but other agents will follow. The beauty of it is that with a single 3270 session, in a single address space, you can view data across your enterprise, as shown in Figure 1:

The workspace design, navigation, and data in this interface let you identify the root cause of a problem quickly and focus on the critical data needed for problem determination.

Using a native z/OS user ID (one defined on z/OS) and password or pass phrase, this interface authenticates all logon requests to the OMEGAMON product. If you define a security class, the interface also validates the logon attempt for authorization. You can also have individual data queries validated for authorization to access data from different OMEGAMON agents, allowing multiple users with different levels of security to log on to one OMEGAMON enhanced 3270 user interface.

About OMEGAMON XE on z/OS V5.1.0

Using the OMEGAMON enhanced 3270 user interface, OMEGAMON XE on z/OS V5.1.0 provides faster problem determination and availability management. You can now view an Enterprise Summary of multiple sysplexes and navigate easily from sysplex and central processing complexes (CPCs) to LPAR to address space level analysis. And we built customer-prioritized problem solving scenarios into the new workspaces to help you solve your high priority problems quickly.

When you log on to OMEGAMON XE on z/OS V5.1.0, you’ll see a workspace containing a summary of the known sysplexes, displaying critical performance data that you can use to quickly identify any sysplex that might be experiencing problems. From this workspace, you can evaluate sysplex, CPC, and LPAR overview data to...
determine if further analysis or action is required. An exceptions subpanel on the Sysplex and CPC Overview workspaces offers highlighted and colored exceptions in categories based on status thresholding similar to that of prior interfaces. Using subpanel navigation, you can quickly analyze the details of problems and take corrective action, if necessary.

Two other unique high level displays are the Sysplex LPAR Overview and Top Consumers workspaces:

- The Sysplex LPAR Overview presents several key performance measures for each LPAR in the sysplex.
- The Top Consumers workspace offers an LPAR perspective of top consuming address spaces for different performance indicators in the sysplex. These indicators include CPU, real storage, virtual storage, CSA, ECSA, I/O, enqueues, and the service classes that are the lowest performers. This workspace is especially important because monitoring and managing address spaces and service classes of top resource consumers is key to keeping your z/OS system running smoothly.

Move easily to different environments
Each workspace displays your currently selected sysplex and SMF ID in the upper-right corner of the screen. In many cases, you can type in these fields to switch to a different Sysplex or SMF ID (a 4-character string uniquely identifying the z/OS image within a sysplex), quickly moving a workspace from one environment to another without traversing the navigation tree.

Take action!
OMEGAMON XE on z/OS V5.1.0 also introduces several Take Action commands. These commands can be run from either the Tivoli Enterprise Portal or the new 3270 user interface. V5.1.0 provides the following agent functions:

- Swap in address space
- Mark address space nonswappable
- Mark address space swappable

Integrated IBM Health Checker for z/OS information on the 3270 user interface
In another great feature, OMEGAMON XE on z/OS integrates IBM Health Checker for z/OS check information for z/OS systems from the existing Tivoli Enterprise Portal on the enhanced 3270 user interface. (Note that this requires that IBM Health Checker for z/OS software be installed, configured, and running, which is a snap—see IBM Health Checker for z/OS User’s Guide, SA22-7994.))

And two new attribute groups
Finally, the 3270 user interface in OMEGAMON XE on z/OS V5.1.0 now features two new attribute groups, Host Configuration and Sysplex Configuration, to provide monitoring status and configuration information for sysplexes and CPCs. Although these attributes are intended primarily for workspaces, you can also use them to create monitoring situations and automation policies in your Tivoli Enterprise Portal.

About OMEGAMON XE for CICS on z/OS V5.1.0
When you log on to OMEGAMON XE for CICS on z/OS V5.1.0, you’ll see a workspace containing a summary of each known CICSPlex®, displaying critical performance data you can use to quickly identify any CICSPlex that might be experiencing problems.

For example, a CICS region within a CICSPlex might be experiencing slow response time. You would use the CICSPlex-wide service level analysis workspace to display the service level classes and then display the CICS regions affected to determine which CICS regions are experiencing the problem and which transactions are impacted.

OMEGAMON XE for CICS on z/OS V5.1.0 adds support for CICSPlex views of your CICS regions for both the Tivoli Enterprise Portal and the enhanced 3270 user interface. You can either use these views to create rules to classify your CICS regions into CICSPlexes or, if you use CICSPlex / System Manager (CP/SM) to manage your CICS regions, OMEGAMON XE for CICS on z/OS can reflects these definitions directly in the new CICS workspaces and subpanels.
Using CICSplex-wide queries

OMEGAMON XE for CICS on z/OS V5.1.0 groups CICS regions into CICSplexes to provide CICSplex-wide queries. For example, in the Task Details workspace, you can see whether a task has related tasks (tasks with the same unit of work ID) on other CICS regions within the CICSplex. You can also switch from there to viewing the related task directly from your current Task Details workspace. This means that you can quickly view related information even when the task is in another CICS region within the CICSplex.

You’ll see your currently selected CICSplex and CICS region displayed in the upper-right corner of the screen of each workspace. In many cases, you can type directly in these fields to switch to a different CICSplex or CICS region within the selected CICSplex, quickly moving a workspace from one environment to another without traversing the navigation tree.

To get a CICSplex-wide view of your resources to aid in problem determination, you can issue CICSplex scope queries. For example, let’s assume that a program defined in multiple CICS regions within a CICSplex is having a problem in only one of the CICS regions. You can use the program summary display to quickly see everywhere in the CICSplex that the program is defined and view the program attributes from both the CICS table definition and the program itself in each CICS region. From there, you can quickly determine if the program in one CICS region is a different version from the versions in other CICS regions in the CICSplex. For example, you’ll be able to see if the program is configured differently or has a different length than the other programs in the CICS region experiencing the problem.

Using commands

- Use the FIND command to issue CICSplex-wide queries that quickly locate resources within your CICSplex. For example, use the FIND PROGRAM WKDL82 command to create a summary display of all CICS regions within the current CICSplex with the WKDL82 program defined. Enter the FIND command without operands to display a menu of items that you can search on, as we show in Figure 2:

  ![Figure 2. Using the FIND command to display a search menu](image)

  - Issue Take Action commands against resources in the enhanced 3270 user interface to terminate tasks or change the status of resources. For example, you can open a file or disable a transaction. If you want to simplify security for your installation environment, secure the Take Action commands you issue through both the Tivoli Enterprise Portal or the 3270 user interface.

Summary

The OMEGAMON enhanced 3270 user interface offers efficient problem determination and availability. Install this OMEGAMON XE for z/OS V5.1.0 and see the difference it will make in keeping your system well-tuned and managed. Shed a beacon of light on z/OS performance and management of your data and improve your z/OS product portfolio by adopting the new release today!
Advanced Function Printing (AFP) is going through some transformations. This article is a “how to” with tips to ease your move to the new from AFP print transforms.

If you use the from AFP transforms with Infoprint Server and have installed z/OS V1R12 or V1R13, you are aware of the new transform product IBM Print Transforms from AFP for Infoprint Server for z/OS V1.1. This product is a replacement for the previous product Infoprint Transforms from AFP to PCL, PostScript®, or PDF V2.1 for z/OS. The team designed the new product to coexist with the old product so that customers aren’t forced to make an immediate transition.

The new product uses the same Infoprint Server interface and configuration options, but has different names for transform commands, filters, and daemons. Here is a comparison of the names of the transform commands, filters, and daemons for the previous Infoprint Transforms and the new Print Transforms.

### Moving to Print Transforms from AFP

There are several steps to consider when planning your migration to the new transform product. Follow these required steps to complete the migration:

1. Edit the aopxfld.conf configuration file to add the new transform class entries.
2. Stop and restart the Infoprint Server Transform Manager daemon.
3. Edit the Infoprint Server printer definitions to modify a printer definition.
4. Edit the JCL and shell scripts to call out the new transform.
5. Add the new .aokcinit file to the install path bin subdirectory.
6. If you encrypt the transformed PDF documents, you might need to make changes to the AOP_PASSWORD_EXIT environment variable.

Print Transforms from AFP for Infoprint Server for z/OS Version 1 Release 1, G324-2634, provides a section on migration to the IBM Print Transforms with useful information about these steps. It also explains how to use the Printer Inventory Definition Utility program (PIDU) to manage objects in the Infoprint Server printer inventory. See the Considerations and References section at the end of this article for references to product names, numbers, and documentation.

This article will show you how to edit the Infoprint Server printer definitions and the JCL and shell scripts for the new transforms (steps 3 and 4).

#### Tips for Step 3: How to edit the Infoprint Server printer definitions

As shown in Table 1, the previous transform product and the new product use different transform filter names. The Infoprint Server printer definition can refer to the transform filter names. If you want your current printer definitions to use the new Print Transforms, you need to modify the filter section of the printer definition. This could be very time consuming if you have a large number of printer definitions to modify.

<table>
<thead>
<tr>
<th>Transform product</th>
<th>Transform commands</th>
<th>Transform filters</th>
<th>Transform daemons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infoprint Transforms from AFP</td>
<td>afp2pcl</td>
<td>afp2pcl.dll</td>
<td>afp2pcl.dll</td>
</tr>
<tr>
<td></td>
<td>afp2pdf</td>
<td>afp2pdf.dll</td>
<td>afp2pdf.dll</td>
</tr>
<tr>
<td></td>
<td>afp2ps</td>
<td>afp2ps.dll</td>
<td>afp2psd</td>
</tr>
</tbody>
</table>

| Print Transforms from AFP      | afpxpcl            | afpxpcl.dll       | afpxpcl.dll       |
|                                | afpxpdf            | afpxpdf.dll       | afpxpdf.dll       |
|                                | afpxps             | afpxps.dll        | afpxpsd           |

Table 1. Name changes for the new Print Transforms from AFP
To reduce the time and effort of the conversion, the new Print Transform product includes a shell script tool called aokmodfilter.sh, which is located in the following directory:

```
/usr/lpp/Printsrv/samples
```

This tool can be used to modify the filter attribute of the printer definitions.

A sample printer definition

Example 1 shows a printer definition the pidu command generates that contains filter entries for the previous Infoprint Transforms product.

```
create printer DUP0010_AFP2PCL
  printer-codepage = ISO8859-1
  filters = {
    line -> "afp2pcl.dll -cus %filter-options"
    modca -> "afp2pcl.dll -cus %filter-options"
    text -> aopfiltr.so
  }
  printer-type = ip-printway
  location = "area printer"
  destination = D10PCL
  output-class = O
  successful-retention-period = 0024:00:00
  failure-retention-period = 0024:00:00
  retry-limit = 3
  retry-time = 0000:05:00
  print-queue-name = text
  printer-ip-address = 9.10.109.248
  dcf-routing = yes
  begin-dataset-exit = ANFUXBD2
  end-dataset-exit = ANFUXED1
  pcl-orientation = landscape
  description = AFP2PCL
;
```

Example 1. A printer definition generated by the pidu command

Example 2 shows you how to use the new aokmodfilter.sh tool. This example shows how the tool converts the previous printer definition transform filter name from afp2pcl.dll. The tool will also convert the transform filter names afp2ps.dll and afp2pdf.dll to afpx2ps.dll and afpxpdf.dll, respectively.

Run the tool /usr/lpp/Printsrv/samples/aokmodfilter.sh. The output is shown:

```
=>aokmodfilter.sh
  Migrating afp2 filters to afpx
  AOP065I 137 definitions were exported to /tmp/inventory.dump.
  modify statements written to /tmp/filter.defs
  removing file /tmp/inventory.dump

1. Examine the tool's output file
   created in /tmp/filter.defs to verify
   that the modify commands are
   correct. The tool creates a single
   modify command for each printer
   definition in the active printer
   inventory with a filter value that
   refers to a transform filter name
   used by the previous transform
   product.

   For the printer definition used
   in this example, the tool generates
   the following modify command:

```
modify printer DUP0010_AFP2PCL
  filters={line -> "afpxpcl.dll -c us %filter-options"
    modca -> "afpxpcl.dll -c us %filter-options"
    text -> aopfiltr.so
  };
```
```

2. Review and edit the /tmp/filter.defs file to remove any modify commands for the printer definitions you do not want to change.

3. Use the pidu command to modify
   the original printer definitions.

   The output is shown:

```
=>pidu < /tmp/filter.defs
  AOP063I printer DUP0010_AFP2PCL was replaced.
```
```
Example 2. Using aokmodfilter.sh to modify printer definitions
```

Example 3 shows the modified printer definitions with the filter values that now refer to the new Print Transforms:

```
create printer DUP0010_AFP2PCL
  printer-codepage = ISO8859-1
  filters = {
    line -> "afpxpcl.dll -c us %filter-options"
    modca -> "afpxpcl.dll -c us %filter-options"
    text -> aopfiltr.so
  }
  printer-type = ip-printway
  location = "area printer"
  destination = D10PCL
  output-class = O
  successful-retention-period = 0024:00:00
  failure-retention-period = 0024:00:00
  retry-limit = 3
  retry-time = 0000:05:00
  print-queue-name = text
  printer-ip-address = 9.10.109.248
  dcf-routing = yes
  begin-dataset-exit = ANFUXBD2
  end-dataset-exit = ANFUXED1
  pcl-orientation = landscape
  description = AFP2PCL
;
```
```
Example 3. Modified printer definitions
```

The result of these steps is that all future jobs that refer to the sample Infoprint Server printer definition will now run the new IBM Print Transform afpxpcl.dll. You can use the new aokmodfilter.sh tool to quickly change all of your printer definition filter values that referred to the previous transform filter names.

**Tips for Step 4: How to edit the JCL and shell scripts**

As shown in Table 1, the previous transform product and the new transform product use different transform command names. The AOPBATCH and AOPPRINT JCL procedures make use of the transform command names and in turn are
used by any UNIX shell scripts that run the transform. The JCL in this discussion applies only to the AOPBATCH and AOPPRINT procedures. Other batch JCL procedures that use the Infoprint Server printer definitions to produce transformed output are part of step 3 and are not covered in this step.

Example 4 shows a sample AOPBATCH JCL procedure that uses the previous Infoprint transform command afp2pcl:

```c
Example 4. AOPBATCH JCL that uses the Infoprint transform command afp2pcl:

/* AOPPRINT - OS/390 Print Server batch print procedure */

//TRANSM EXEC PGM=AOPBATCH,
// PARM='/afp2pcl -o //DD:OUTPUT -jattr//'DD:ATTRS //DD:INPUT' /OUTPUT DD DSN=PRODUCTN.AFP.INPUT,DISP=(NEW,CATLG,DELETE), //DCB=(RECFM=VB,LRECL=1024),SPACE=(CYL,(1,1)); //INPUT DD DSN=PRODUCTN.AFP.INPUT,DISP=SHR //ATTRS DD *
form-def=f1cp08
page-def=p1cp08c
fail-on-transform-error=error
trailer-transform-error-page=warn

/* STDOUT DD SYSOUT=* */
/* STDERR DD SYSOUT=* */
/* STDENV DD * */
AOP_MVS_RETURN_CODES=YES
AOPCONF=/etc/Printsrv/aopd.conf
NLSPATH=/usr/lpp/Printsrv/En_US/%N:/usr/lib/nls/msg/En_US/%N
LIBPATH=/usr/lpp/Printsrv/lib:/lib:/usr/lib
PATH=/usr/lpp/Printsrv/bin:/bin:/usr/bin

//STDENV DD *

Example 4. AOPBATCH JCL that uses the Infoprint transform command afp2pcl
```

Using symbolic links

The straightforward way to switch to the new Print Transforms is to edit the JCL to change the command to afpxpcl. Editing each JCL file works well, but can be tedious, especially if many JCL files need to be changed. And if there are other teams with their own AOPBATCH JCL outside of your control, the change becomes very troublesome.

An alternative method is to use symbolic links to re-route the commands. Once the symbolic link process is complete, AOPBATCH and AOPPRINT JCL procedures will process with the new Print Transforms.

The following steps in Example 5 allow you to create a symbolic link from afp2pcl to afpxpcl. The same steps can be used to link afp2ps to afpxps and afp2pdf to afpxpdf. Example 5 uses the /usr/bin directory, but you can use any suitable directory.

1. Create a symbolic link from afp2pcl to /usr/lpp/Printsrv/bin/afpxpcl in /usr/bin:
   ```bash
   => ln -s /usr/lpp/Printsrv/bin/afpxpcl /usr/bin/afp2pcl
   ```
2. You will need to modify the PATH environment variable specified in the aopstart EXEC. Or you can edit the
   ```bash
   export PATH=.:/usr/bin:$PATH
   ```
3. If you modify the aopstart EXEC, you will have to stop (AOPSTOP) and restart (AOPSTART) Infoprint Server to activate this change.

Example 5. Using symbolic links for transform commands

All of the AOPBATCH and AOPPRINT JCL jobs that refer to afp2pcl will now run the new Print Transform command – afpxpcl. As shown in the JCL code in Example 4, you can make use of UNIX scripts to run the transform commands. You will need to change any scripts that refer to the previous Infoprint transform to use the new Print Transform commands. Using symbolic links is a good alternative to editing the scripts.

Example 6 shows you how to create symbolic links for scripts:

1. Follow step 1 in the previous JCL example (Example 5) to create the symbolic link.
2. Modify the PATH environment variable specified in the UNIX shell running the script:
   ```bash
   => export PATH=../usr/bin:$PATH
   ```

Example 6. Using symbolic links for UNIX scripts

When these steps are complete, all future script commands that refer to afp2pcl will now run afpxpcl.

Acknowledgments

Thanks to Tony Mingo and Tariq Choudhry for their assistance.

Considerations and references

- To obtain IBM Print Transforms from AFP for Infoprint Server, install APAR OA38585 on your z/OS V1R12 or V1R13 system. The new IBM Print Transform product provides equivalent functions to the previous Infoprint Transforms V2.1 product. The APAR also includes the new features described in this article. The product names and numbers are:
  - IBM Print Transform from AFP to PCL for Infoprint Server for z/OS V1.1 (program number 5655-TF1)
  - IBM Print Transform from AFP to PDF for Infoprint Server for z/OS V1.1 (program number 5655-TF1)
  - IBM Print Transform from AFP to PostScript for Infoprint Server for z/OS V1.1 (program number 5655-TF3)

- Print Transforms from AFP for Infoprint Server for z/OS Version 1 Release 1, G324-2634, provides complete information for migration and the steps covered in this article:
  - [publibz.boulder.ibm.com/epubs/pdf/aokfa100.pdf](publibz.boulder.ibm.com/epubs/pdf/aokfa100.pdf)
  - For downloading any Infoprint publication, see the IBM Publications Center: [ibm.com/shop/publications/order](ibm.com/shop/publications/order)
Recycle your input: AFP transform enhancements

BY RAY MORSE, TONY MINGO, AND TARIQ CHOUHRY

The new IBM Advanced Function Presentation (AFP) transform products provide features that you will want to investigate. The following products include these new features:

• IBM Print Transform from AFP to PCL for Infoprint Server for z/OS V1.1 (program number 5655-TF2)
• IBM Print Transform from AFP to PDF for Infoprint Server for z/OS V1.1 (program number 5655-TF1)
• IBM Print Transform from AFP to PostScript for Infoprint Server for z/OS V1.1 (program number 5655-TF3)

These features can help you detect problems with the AFP input data stream and prevent documents with incorrect output from being produced for the user.

In previous transform products, whenever a problem was found with the input AFP data stream, the transform processing either ignored the problem and continued, or stopped processing, sometimes with an error message appended to the output document. In either case, transform processing did not issue any messages to the Infoprint Server message log. If a production job generated the input data stream error, the error message appeared in the user's output, with no notification sent to you. In fact, transform processing left a message in the log stating the process was successful, and issued a return code of 0! You could not monitor the message log or implement return-code checking to prevent the printing or distribution of incorrect output. You only learned of the problem when notified by the user.

The new AFP Print Transform products provide features for error monitoring, message logging, and conditional return-code processing.

Feature one: Use the fail-on-transform-error attribute

The AFP Print Transform products provide a new error fidelity setting to adjust which type of errors cause a job to fail. You can set transform processing to halt or continue when problems occur with the input.

This feature is controlled by an attribute called fail-on-transform-error.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>error (default)</td>
<td>The transform stops when an error occurs.</td>
</tr>
<tr>
<td>no</td>
<td>The transform continues when an error or warning occurs.</td>
</tr>
<tr>
<td>warning</td>
<td>The transform stops when an error or warning occurs.</td>
</tr>
</tbody>
</table>

Table 1. Values for the fail-on-transform-error attribute

Choosing which values to set

Two phases occur when a document is produced:

1. In the development phase, a new document is created.
   During this phase, you want to quickly find and correct any problems with the input data. You want to see all the problems that transform processing can find and report in one pass. For this phase, set the attribute to no.

2. In the production phase, the document is put into production.
   After the document is in production, you do not want the document to be produced if there are errors in the input. During this phase, set the attribute to warning or error to stop transform processing.

Feature two: Include or suppress the message trailer page

Using the new transform products, you can include or suppress the trailer message page. You can choose whether error messages, warning messages, or both are appended to the document. You might want messages appended to the document for easy reference during the development phase.

However, after the document is placed into production, you longer receive the document output. At that point, you want the messages in the Infoprint Server message log so they can be queried. The messages are always written to the message log and the transform's standard error output file (STDERR). This feature is controlled by the trailer-transform-error-page attribute.
Value | Explanation
--- | ---
warning (default) | Transform processing writes warning messages and error messages to the trailer page.
error | Transform processing writes error messages to the trailer page.
no | Transform processing does not produce a trailer page.

Table 2. Values for the trailer-transform-error-page attribute

Feature three: Use MVS-like return codes to control incorrect input

The new transform products provide return codes that you can use to conditionally control processing steps. This capability is controlled by a z/OS UNIX environment variable called AOP_MVS_RETURN_CODES. This environment variable can be used with the AOPBATCH utility, the AOPPRINT procedure, or in a z/OS UNIX script. Note that the Print transform products ignore AOP_MVS_RETURN_CODES if you add it to the AOPXFD. CONF file. The sample batch print procedure that follows shows you how to set the AOP_MVS_RETURN_CODES environment variable.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| NO (default) | Transform processing returns z/OS UNIX exit values.  
0 Success. However, an input data stream error or warning might have occurred.  
>1 The transform command failed. |
| YES | Transform processing produces MVS return codes on exit.  
0 Transform processing completed successfully with no warning or errors in the input data stream.  
4 Transform processing encountered a warning-level problem in the input data stream.  
8 Transform processing failed or an error occurred in the input data stream. |

Table 3. Values for the AOP_MVS_RETURN_CODES environment variable

The Print transform exit return code values are influenced by the combined values of the fail-on-transform-error attribute and the AOP_MVS_RETURN_CODES environment variable.

<table>
<thead>
<tr>
<th>Job attribute</th>
<th>Field in ISPF printer definition processing section</th>
<th>Transform class environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail-on-transform-error</td>
<td>Fail on error</td>
<td>AOP_FAIL_ON_ERROR</td>
</tr>
<tr>
<td>trailer-transform-error-page</td>
<td>Trailer error page</td>
<td>AOP_TRAILER_ERROR_PAGE</td>
</tr>
</tbody>
</table>

Table 4. Transform class environment variables

Table 5 includes the transform return code values that are based on the environment variables.

<table>
<thead>
<tr>
<th>fail-on-transform-error</th>
<th>AOP_MVS_RETURN_CODES =YES</th>
<th>AOP_MVS_RETURN_CODES =NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Successful: 0  Warning: 4  Error: 8</td>
<td>Successful: 0  Warning: 0  Error: 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>Successful: 0  Warning: 4  Error: 8</td>
<td>Successful: 0  Warning: 0  Error: 1</td>
</tr>
<tr>
<td>WARNING</td>
<td>Successful: 0  Warning: 8  Error: 8</td>
<td>Successful: 0  Warning: 1  Error: 1</td>
</tr>
</tbody>
</table>

Table 5. Transform return code values that are based on the two variables
A sample batch print procedure

The following example, which uses the AOPBATCH utility, shows you how to use these new features to ensure that only correct output is delivered to the user. In this example, transform processing fails if an error occurs in the input data stream. An error message is not included on the trailer page. Any error messages are logged in the Infoprint Server message log and the transform’s standard error output file (STDERR).

```csh
//*--------------------------------------------------------
//* AOPPRINT – z/OS Print Server batch print procedure
//*--------------------------------------------------------
//TRANSFRM EXEC PGM=AOPBATCH,
//OUTPUT DD DSN=PRODUCTN.OUTPUT.PCL,DISP=(NEW,CATLG,DELETE),
// DCB=(RECFM=VB,LRECL=1024),SPACE=(CYL,(1,1))
//INPUT DD DSN=PRODUCTN.INPUT.DOCUMENT,DISP=SHR
//ATTRS DD *
form-def=f1fdef
page-def=pipdef
fail-on-transform-error=error
trailer-transform-error-page=no
/*
//STDOUT DD   SYSOUT=* 
//STDERR DD   SYSOUT=* 
//STDENV DD *
AOP_MVS_RETURN_CODES=YES
PATH=/usr/lpp/Printsrv/bin:/bin:/usr/bin
LIBPATH=/usr/lpp/Printsrv/lib:/lib:/usr/lib
NLSPATH=/usr/lpp/Printsrv/En_US/%N:/usr/lib/nls/msg/En_US/%N
AOPCONF=/etc/Printsrv/aopd.conf
/*
/* PRINTIT EXEC PGM=IEBGENER,REGION=212K,COND=(0,LT,TRANSFRM)
```

Notes and references

The new IBM Print Transform products provide equivalent functions to the previous Infoprint Transforms V2.1 product. To get the new features described in this article, install the PTF for APAR OA38585.

The *Print Transforms from AFP for Infoprint Server for z/OS* manual, *G325-2634*, provides a very good explanation of the attributes and environment variables. You can find the publication on these websites:

* z/OS Internet Library:
  [ibm.com/systems/z/OS/zos/bkserv](ibm.com/systems/z/OS/zos/bkserv)

* IBM Publications Center:
  [ibm.com/shop/publications/order](ibm.com/shop/publications/order)
Lucky you! z/OS V1R13 is a great place to be!

BY MARNA WALLE

By now, you probably have read about the change in z/OS release cycles with z/OS V2, starting after z/OS V1R13. What this means to you right now is that you'll have longer to migrate to z/OS V1R13, and you'll even have time to start exploiting some new functions before you have to begin a subsequent migration. Use this extra time to enhance your z/OS V1R13 migration plan with some items that will help you out in the long run. It’s like having your cake and eating it too!

Of course, you are using the critical z/OS V1R13 Migration book as your planning guide for the migration. But, are you using the latest level of the book? We've done a refresh of the z/OS V1R13 Migration book with some late-breaking items. If you are not using book number GA22-7499-20, you’re down-level. You can pull the latest level of the book (as well as the customized z/OS V1R11-V1R13 and z/OS V1R12-V1R13 books) from the website:


Let’s look at a couple of directions that you could include into your migration plans. The first two are migration actions in which the end of the runway is now is sight, and that have definite benefits after you’ve done them. By incorporating these migration actions into your z/OS V1R13 migration schedule, you’ll be in a great position for exploiting some enhancements that you may have been missing up until now.

Move to the console operating mode of DISTRIBUTED

The planned last release on which SHARED mode is possible is the release after z/OS V2R1. DISTRIBUTED console operating mode offers enhancements that have been desired for a long time: possible reduced times when IPLing and when joining a sysplex, breaking the old limit of having 99 MCS, SMCS, and subsystem console per sysplex by allowing up to 99 MCS, SMCS, and subsystem consoles active per system, and possibly reducing downtime from consoles related outages. Check out z/OS Planning: Operations for complete instructions on how to move from SHARED to DISTRIBUTED mode. Note that in z/OS V1R13, the default will be DISTRIBUTED if you don’t specify your preference on CON= in IEASYSxx. Use IBM Health Checker® for z/OS check ZOSMIGV1R13_CNZ_CONS_OPER_MODE to see if you are using the default and will be affected.

Use IBM Health Checker for z/OS check JES2_Z11_UPGRADE_CK_JES2 to see if you have met all the preconditions to upgrade to z11 level.

Got IBM Health Checker for z/OS turned on?

IBM Health Checker for z/OS offers quick assistance to busy system programmers who would like to make sure that they are configured for maximum availability and for quick feedback on migration positioning. If you’ve already got IBM Health Checker for z/OS up and running, bravo! For those of you who haven’t yet activated IBM Health Checker for z/OS, why not use the extra time before z/OS V2 is available to set it up? It doesn’t take very long to get IBM Health Checker for z/OS initialized, and although it may take some time to review all the check output, it is well worth it. You can get benefit from the checks that you’ve already reviewed, while leaving some check reviews for later if you need to.

Move to z11 JES2 checkpoint level

You could have guessed that the days of JES2 z2 checkpoint level were ending soon, because z11 level has been around since z/OS V1R11. The last planned release on which you can use z2 is z/OS V2R1. By moving to z11 level now, you can take advantage of the following enhancements: higher limits for JQEs, JOEs, and BERTs which will allow you to have more jobs and pieces of output handled, SAPI (SSI 79) and Extended Status (SSI 80) support for selection by transaction job name and the transaction job ID, and Extended Status (SSI 80) can return the transaction information within the terse SYSOUT section.
More z/OS V1R13 migration hints and tips from Professor Kimura

By Shigeki Kimura

In this hints and tips article for migrating to z/OS V1R13, we’ll explore some considerations and recommendations as a result of important functional changes to the release, including changes of behavior to consider after the migration. Professor Kimura also shares some hints and tips observed during the z/OS V1R13 Early Support Program (ESP) for participating customers.

For a good summary of migration actions for V1R13, be sure to read “Lucky you! z/OS V1R13 is a great place to be!” on p. 31.

**Hint #1**
Removing PPT PGNAME(DFHSIP) as a result of the new IBM default PPT entry

By applying APAR OA36376 for z/OS V1R13, an entry for program DFHSIP (CICS driver) has been added to the IBM default program properties table (PPT) with the attributes of NOSWAP and NOPREF (as described in the documentation for DOC APAR OA37028).

To make the new default PPT effective, you can now safely remove the PPT PGNAME(DFHSIP) entry from SCHEDxx parmlib member.

**Hint #2**
Applying APAR changes to handle SAF authority for the System Logger address space

z/OS V1R13 introduced a new IXGCNFxx parmlib member to specify the logger monitoring intervals for warning and action messages. System Logger address space IXGLOGR issues a DISPLAY LOGGER,IXGCNF command during initialization to display the current parameters, even when no IXGCNFxx members are requested.

To avoid SAF authorization failures, you need to ensure that the proper security authority, such as read access to RESOURCE(MVS.DISPLAY.LOGGER) CLASS(OPERCMDS), has been defined for the IXGLOGR address space.

If you do not specify the new IXGCNFxx members, you can avoid this migration action by applying the APAR OA38846 because during initialization System Logger now only issues the command when the new IXGCNFxx members are requested at IPL.

**Hint #3**
Viewing ISPF panel changes to ISPF information for new allocation of non-SMS-managed sequential data set

Before z/OS V1R13, the allocation option (OPT3.2) in Interactive System Productivity Facility (ISPF) opened and closed new non-SMS-managed sequential data sets to write end-of-file (EOF) markers.

Beginning in z/OS V1R13, this action is no longer performed for both OPT3.2 new allocation and OPT3.4 AL new line command panels in order to support the new feature of z/OS V1R11 DFSMS. As a result, the last referenced date is no longer updated on the panel output right after the OPT3.2 and OPT3.4 new allocation of non-SMS managed sequential data set. Instead, ***None*** is displayed in the Referred column in OPT3.4 panel. This behavior is now consistent whether the data set is SMS-managed or not.
**Hint #4**
Understanding changes to the message log as a result of DFSMSdss COPYVOLID operation

z/OS V1R13 DFSMSdss, through APAR OA36296, changed to use the IEEVARYD service to vary the target volume offline during a COPY or RESTORE FULL or TRACKS with COPYVOLID specified when the target volume becomes a duplicate of the source volume. As a result:

- VARY dddd,OFFLINE DFSMSDSS INTERNAL VARY is no longer displayed to the logs.
- Solicited message IEF281I dddd NOW OFFLINE is no longer issued.

You need to watch for the new message:

**IEF880I dddd NOW OFFLINE BY ADRSBRTN**

It lets you know that the vary offline to the device has been done and indicates the action was the result of a DFSMSdss operation with COPYVOLID specified.

**Hint #5**
New support for the new BPX DEFAULT.USER profile in the RACF FACILITY class

When you share the RACF database between z/OS V1R10 and V1R12 with both BPX.UNIQUE.USER (new in z/OS V1R11) and BPX.DEFAULT. USER profiles defined, you can enable automatic assignment of unique IDs on z/OS V1R12 systems and enable default OMVS segment processing on z/OS V1R10 systems. However, when you migrate to z/OS V1R13 and share the RACF database between z/OS V1R11 and V1R12, all of the systems support the new BPX.UNIQUE.USER profile, and default OMVS segment processing will no longer occur on any system even if both profiles are defined.

DOC APAR OA38474 has been taken for the *RACF Security Administrator’s Guide*, SA22-7683-15, to clarify this situation. Also, note that z/OS V1R13 is planned to be the last release to support BPX.DEFAUL. USER profile. See "Nobody’s deFault but mine" on page 77.

**Hint #6**
Updating IKJTSO for authorization change to IEBCOPY

z/OS V1R13 DFSMS changed IEBCOPY so that it is no longer linked AC(1) and will no longer run as authorized, but the entries for IEBCOPY in SYS1.SAMPLIB member IKJTSO00 have not been removed. While there is no harm in IEBCOPY being in IKJTSOxx, as it will not run authorized, it could lead to confusion if it is left in.

You need TSO/E APAR OA37218 to correct the SAMPLIB member IKJTSO00 to remove the entries for IEBCOPY from the AUTHCMD, AUTHPGM, and AUTHTSF statements. It is advisable for you to update your own IKJTSOxx parmlib members to reflect this change. ISPF APAR OA37408 is recommended to apply before this action.

For a complete list of migration actions for z/OS V1R13, see z/OS Migration, GA22-7499-20 at the following web site:

[publibz.boulder.ibm.com/epubs/pdf/e0z2m191.pdf](publibz.boulder.ibm.com/epubs/pdf/e0z2m191.pdf)
How does that great Beach Boys song go? Let’s see, SPOOL .. SPOOL .. SPOOL .. SPOOL .... SPOOL migrations. Wait wait ... I guess the words are not correct. On the other hand, z/OS Release 13 JES2 SPOOL migration is every bit as cool and original as the song. This article introduces some basic migration concepts and considerations.

Does your SPOOL configuration contain volumes that reside on old, antiquated DASD? Would you like to merge those volumes and place the resulting volumes on DASD using the latest and greatest technology? SPOOL migration allows an installation to consolidate SPOOL volumes in a non-disruptive and expeditious manner. Actions which previously could have taken weeks can now be accomplished within a few hours. During and after the migration, applications seamlessly can read and write from migrating volumes without any interruption.

Yes, it is really true that applications need not be quiesced during the migration. All existing SPOOL addresses (MTTR/MQTRs) are automatically handled and mapped during and after the migration by JES2. At migration end, all migrated data now exists on the target and the source volume data sets may be removed. This article will address:

- Pre-migration planning
- Migration phases and brief discussion on what occurs
- After-migration mapped volume status

**SPOOL migration example**

SPOOL migration is a simple way to quickly get data off old hardware and onto new hardware with new features. The best way to wrap your head around the concepts of SPOOL migration is to let us walk you through an example. For our example, we will use the simple case of migrating an existing volume, SPOOL2, into the existing volume, SPOOL5. Let’s go through a few basic steps for performing a SPOOL migration.

Get organized

First, identify the DASD volumes that you wish to migrate. These are potential Source volumes for the SPOOL migration command. Then identify the DASD volumes that will contain the migrated data and therefore are the potential Target volumes. For our example, we want to migrate the SPOOL data on Source volume SPOOL2 to the SPOOL data set on Target volume SPOOL5. Figure 1 logically illustrates the SPOOL migration we intend to perform.

Once you have identified the volumes you want to migrate, you must verify your space requirements. Issue SD SPL(SPOOL2),MIGDATA for source volume SPOOL2 in our example. MIGDATA displays two pieces of information. The first piece, SPACE_USED, indicates the highest-used track in the SPOOL2 data set. This is a high-water mark of SPOOL usage on SPOOL2 and it defines the size of the data within the data set that will be considered for migration.

To determine the total free space required on the Target volume for the migration, take the SPACE_USED value and add 4 tracks to account for temporary JES2 structures that are stored along with the migrating data.
SPOOL migration looks for in-use tracks in the Source volume data set to migrate, from the beginning of the data set to the track identified by SPACE_USED.

Now determine the available space on Target volume SPOOL5 by issuing the JES2 command $D SPL(SPOOL5),MIGDATA. This time, look at the second piece of MIGDATA, LARGEST_FREE, which reports the size in tracks of the largest contiguous free space on the volume. In our example, LARGEST_FREE for SPOOL5 must be greater than or equal to SPACE_USED for SPOOL2 plus 4 tracks. If the criteria is met, then SPOOL5 can become a Target volume for SPOOL2 migrated data. Rerun these $D SPL,MIGDATA commands right before starting the SPOOL migration, to ensure that the space requirements have not changed. Figure 1 provides a logical look at the concepts of SPACE_USED and LARGEST_FREE.

Next, consider improving the performance of your SPOOL migration by reducing the amount of data to migrate and the competition for the volumes involved. One way is to place the Source volume in DRAINING state using the JES2 command $P SPL. A DRAINING volume will not allow jobs to acquire new space on the volume, reducing contention for the volume and the data to be migrated. Or, you can set the volume as RESERVED. A reserved volume is similar to a volume in DRAINING state in that work can be selected on the volume but new space cannot be acquired.

Reserving the Source volume reduces the competition of jobs accessing the volume. Reserving the Target Volume keeps processing other than the SPOOL migration from using the free space found earlier with MIGDATA. Use the JES2 command $T SPL(SPOOL2),RESERVED to reserve the Source volume SPOOL2 in our example, $T SPL(SPOOL5),RESERVED for the Target Volume.

A SPOOL migration requires roughly the same amount of time it takes to write a data set on your system that is the same size as your SPOOL data set. Informal IBM laboratory testing typically migrated a one million track data set in a matter of a few hours (your mileage may vary).

Crank 'er up!

You made your preparations, you confirmed your migration space requirements, so now it is time to migrate! Enter the JES2 command to perform our example SPOOL migration:

$M SPL(SPOOL2),TARGET= SPOOL5

Output displayed at the console indicates the migration has started and indicates the first phase of SPOOL migration, INITIALIZING, has started:

$HASP808 Migration of SOURCE=SPOOL2 volume to TARGET=SPOOL5 volume RC=9 -- Migration INITIALIZING phase started.

There are six key phases in SPOOL migration:

- PENDING – migration cannot start right away
- INITIALIZING – general configuration work is being done
- SETUP – gets all multi-access SPOOL (MAS) members on the same page
- COPY – allocated tracks from the data set on the Source volume are copied to the Target volume. Most of the time for a SPOOL migration is spent in the COPY phase.
- CATCHUP – tracks changed on the Source volume during the COPY phase by running applications are recopied to the Target volume. The time spent in this phase is directly related to the amount of non-migration activity that occurs on the Source volume during the COPY phase.
- CLEANUP - general cleanup at the end of the migration

Normally, console messages only indicate the beginning and end of the SPOOL migration, as seen in Figure 2. Messages that indicate the beginning and end of each phase are written to SYSLOG. All these messages can be routed to the console by running the JES2 command $T DEBUG,VERBOSE=YES The $D SPL command can also provide status and progress information. During an active SPOOL migration, the Source volume has a STATUS of MIGRATING. The parameter PHASE displays the current phase of an active SPOOL migration (see Figure 2). A better progress indicator is the MPERCENT parameter, which displays the percentage completed. A majority of the time spent in a SPOOL migration is spent in the COPY phase, so MPERCENT is the best method for seeing progress in that phase.
Are we there yet?
The $HASP808 RC=33 indicates that the SPOOL migration completed (see Figure 2). But is it really done? Well, the data that once resided in the data set on volume SPOOL2 now exists in the data set on volume SPOOL5. So you can delete the data set on volume SPOOL2, and as far as JES2 is concerned you can power off that DASD and roll it out the door and use it for target practice!

In our example, however, jobs still think they have space allocations on SPOOL2. The JES2 command $D SPL in Figure 2 shows volume SPOOL2 with a STATUS of MAPPED. That means a logical control block exists representing SPOOL2 (a DAS, or direct access SPOOL data set control block). The SPOOL migration moved SPOOL2 data track by track, without any understanding of uses of the data, pointers within that data (MTTR/MQTR), etc. Jobs and applications still reference SPOOL2 data and JES2 auto-magically remaps their old SPOOL2 MTTR/MQTR pointers to the new location in the SPOOL5 data set using the SPOOL2 DAS mapping information.

So when does JES2 no longer require the MAPPED SPOOL2 DAS? When the last job that thinks it has space allocations on SPOOL2 goes away, the logical representation of SPOOL2 will go away. The JES2 command $DJQ,SPOOL=VOL=SPOOL2 shows jobs that think they have space allocations on SPOOL2. Keep in mind that the Target volume SPOOL5 cannot itself be migrated until all volumes mapped to SPOOL5 have gone away.

Just the tip of the iceberg
We have only touched on the basics of SPOOL migration. There is so much more to know! For example, there is a special type of SPOOL migration called a MOVE Migration! And, did you know that you can cancel a SPOOL migration any time before the CATCHUP phase starts? What about using Extend SPOOL to make more room for your migration? For more information on these and other topics, go to the SPOOL migration web site: ibm.com/systems/z/OS/zos/jes2_spoolmigration.html

Be sure to check out the FAQ link, which covers many of the basic SPOOL migration questions. There is also a link to the migrating SPOOL volumes overview section in z/OS JES2 Commands, a must-read for persons planning to use SPOOL migration. ■
Put some muscle (and brains) behind your mainframes

BY JENNIFER ASH, KATHY PFEIFFER, AND MIKE TODD

Hear what Maria Boonie, IBM VP System z Operating System Development, has to say about IBM’s commitment to the future of the mainframe by helping you develop the System z skills and people resources you need for your business:

IBM is committed to assisting schools worldwide to provide highly skilled students for the mainframe community. With the help of our clients and partners, we (the mainframe community) achieved a significant milestone of over 1,000 member schools using and teaching System z!


Many of you partnered with universities, became members of advisory boards, taught classes, and participated with IBM on-campus visits. Thank You. The IBM Academic Initiative for System z continues to expand offerings for schools, but just as important, we provide alternatives and resources for you to build your mainframe skill base.

This article can help you learn how to obtain new mainframe skills and increase the skills already within your business.

Adding and building mainframe skills in your business

Today’s mainframe IT landscape is continuously evolving. As new workloads emerge and new business applications are designed to optimize unique hybrid computing models, every company needs a skills strategy and succession plan for current and future mainframe training needs.

There are numerous sources of mainframe skills programs and training offerings available worldwide, ranging from college and university courses and degree programs to independent vendors and training organizations, user group communities (such as the mainframe-oriented SHARE and Guide SHARE Europe), as well as certification tests all that assist with skills development needs. In addition to using these external skills offerings, companies also often build and implement internal training programs for employees, using a variety of techniques including on-the-job training, mentoring and coaching, and knowledge transfer learning activities with subject matter experts.

A host of resources

In this article, we’ll talk about the following resources available to help you build mainframe skills in your business. And the good news is that many of these programs are sponsored by IBM at no or reduced cost to you! We can help you:

• Participate in the IBM Academic Initiative® System z
• Locate schools teaching enterprise systems to recruit new talent
• Partner with IBM to build new skills for your business
• Advertise job opportunities to students and experienced professionals
• Use existing resources to assist with building employees skills

• Build System z credentials through certification and mastery tests
• Stay current and connected with System z through Social Media channels

IBM Academic Initiative System z

The IBM Academic Initiative System z is an innovative program that works with schools worldwide to ensure that the next generation of mainframe experts will be available to help businesses leverage the superior security, availability, scalability, and efficiency of the mainframe platform. In this program, IBM provides the following resources to Academic Initiative members and their students:

• Access to mainframes worldwide
• Access to course materials
• Faculty skills development
• Awards, sponsorships, contests
• A connection with the mainframe community

System z mainframe skills are in high demand as businesses continue to deploy smarter System z technologies. Students from IBM Academic Initiative member schools are developing enterprise computing skills to help support the future changing business requirements of clients.

Visit the following web site for more information about the program:

ibm.com/university/systemz

Questions about mainframe skills and enterprise systems technologies? We encourage the community (educators, students, industry, partners) to contact the Academic Initiative System z help desk at zskills@us.ibm.com.
Locate schools that teach enterprise systems
You'll find a list of participating schools and educators worldwide that are currently investigating or actively teaching mainframe technologies online at the following web site:

ibm.com/university/systemz/schools.html

If you are interested in establishing a relationship with a school to recruit top talent, reach out directly to the contacts listed on this website. They want to hear from you!

If you are interested in teaching mainframe and enterprise systems technologies at a new or existing school, contact the Academic Initiative System z help desk at zskills@us.ibm.com.

Partner with IBM to build new skills for your business
IBM wants to partner with companies using System z to build new skills. IBM invests in these partnerships because we want our customers to have the future skills to leverage IBM's mainframe technologies. IBM will assist and enable a school of your choice to teach enterprise systems.

Follow these steps to begin a partnership with IBM and a school:

1. Identify a school of your choice and ask yourself these questions: Where does your company recruit new talent today? Do you have an existing relationship with someone at a college or university? Don't limit your thinking to Computer Science. Consider IT, MIS, IS, CSE and Business majors. IBM has courses and modules that can be included in various majors to expose students to enterprise systems thinking.

2. Contact IBM. Send a note to the following address: zSkills@us.ibm.com

Include in the Subject: “Partnership with IBM and School xyz.” Include a statement about why you want to partner with IBM and the school of your choice.

Advertise System z job opportunities to students and experienced professionals
The IBM Academic Initiative System z makes it easier than ever to connect with students and professionals with mainframe skills. Check out this website: Systemzjobs.com

The web site offers a no-cost, secure and easy-to-use job board for employers to post worldwide job openings. Since its launch in February 2011, several thousand job seekers have registered. One of them might just be the candidate you're looking for!
If your company is recruiting mainframe talent, simply set up an account and post your job description. Job seekers with mainframe skills are searching the listings every day, and they’ll connect with your company to apply.

Whether you’re hiring interns, co-ops, graduating students or experienced professionals, Systemzjobs.com helps you to connect with the right candidate.

Use existing resources to build employees skills

There are a variety of resources available that businesses can use to enhance employee mainframe skills. Some are available at no charge, others at discounts and market rates. Consider a mix of free and for fee training resources when building your training plans.

IBM System z course materials for the community

Over 16 System z foundation courses are available to assist the mainframe community with building skills. Materials range from a 15-week semester length course to smaller modules that can be used to supplement other course materials. There is no charge to use these materials. Visit this site for information:

ibm.com/systems/z/education/skills_coursematerials.html

z/OS Introduction and workshop

This class is a one week introduction to the z/OS operating system and a great way to get started with System z. It is geared towards those who are new to the System z environment, the z/OS operating system, and major software and subsystems like CICS, DB2, Rational® Developer for System z and WebSphere® Application Server. It is located in Dallas, Texas. There is no cost to attend if you are an IBM Academic Initiative member or a member of IBM PartnerWorld.

ibm.com/PartnerWorld/wps/servlet/ContentHandler/pw_home_pub_index

Enterprise Computing Community (ECC)

The mission of the ECC is to improve undergraduate education of large systems and graduate a new generation of talent that will provide industries with the ability to secure, sustain, and grow their operations. The community is open to educators, students, and industry professionals. There is no cost to become a member. Join today at the following site:

ecc.marist.edu

Enterprise Systems Certificate program

This reduced fee 100% online education program is available through Marist College and is designed to educate personnel who are new to the field as well as those with experience in System z and z/OS. There are a variety of certificates you can select from in either a systems programmer track or an application development track including DB2, COBOL, Assembler, and IMS. Classes contain a combination of lecture and “hands on” experience.

www.idcp.org

The Enterprise Computing Research Lab

Available through Marist College, this laboratory is available to academic and industry researchers from the Enterprise Computing Community. Not a member? You can join at the following site:

ecc.marist.edu

You must use the equipment for research, applied research, and research training. Some of the research areas include:

- Mathematical modeling of wave propagation
- Data mining
- Business intelligence and predictive analytics
- Performance benchmarking
- Virtualization and cloud computing

Interested in learning more about the program? Visit the following site:

www.marist.edu/compscimath/researchlab/

IBM Training e-learning bundles

This reduced-fee bundle of e-Learning courses provides a broad spectrum of z/OS education. Students have unlimited access to 150 titles encompassing over 1000 hours of course content:

ibm.com/jct03001c/services/learning/itles.wss/us/en?pageType=classroom_guaranteed&subChapterInd=C&sortBy=6&subChapter=360&x=15&y=13

Guaranteed-to-run classes from IBM

Find out about other fee-based guaranteed-to-run IBM System z (z/OS, z/VM, Linux) classes:

ibm.com/jct03001c/services/learning/itles.wss/us/en?pageType=page&c=V422430P00140S49

Build System z credentials through the IBM System z mastery test

Certification has long been recognized as a differentiator of employee and student skills. Whether you’re a seasoned System z expert or just starting out on System z, IBM provides an opportunity to help you gain official recognition of your technical knowledge and expertise.

IBM System z mastery test

IBM Mastery tests are used to verify that an individual has achieved a foundation of knowledge and understanding of a particular subject matter. IBM System z and z/OS Fundamentals Mastery Test (Test #000-Z05) is based on the contents contained in the Introduction to the New Mainframe: z/OS Basics textbook, SG24-6366. This textbook covers topics ranging from an introduction to z/OS and its environment, application programming on z/OS, online workloads on z/OS, and systems programming on z/OS.
The intended audience for the System z mastery test is any student who has taken enterprise system courses or any industry professional who is new to the mainframe platform (that is, with 1 – 3 years of experience). Experienced professionals who have worked with non-mainframe platforms and want to become more knowledgeable about the mainframe environment and its value also are good candidates.

IBM System z Mastery test information (including assessment test information)
The System z mastery test is a proctored test and is available worldwide at over 5,000 authorized Prometric testing locations. Test information is as follows:

- Number of questions: 73 multiple choice
- Time allowed in minutes: 90 minutes
- Required passing score: 71%
- Cost: $75.00

Note: For a limited time, IBM Academic Initiative members and their students can take the IBM System z mastery test at no charge. Visit the following web site for more information:

ibm.com/university/systemz/masterytest/

This test is also offered at SHARE annually.

Assessing your current skills

If you would like to assess your skill level and readiness before taking the IBM System z mastery test, you can take a fee-based ($10.00) web assessment test. Passing this assessment test does not result in achieving a credential. It is designed to provide feedback and guidance on test preparation. The assessment can be summarized as follows:

- A timed test experience with a full length test in an environment similar to the actual mastery test
- A reasonable indication of test readiness
- Available 24 x 7 at a low cost.

To learn more about the assessment test (including registration information), visit this site:

ibm.com/certify/mastery_tests/samZ05.shtml

Preparing to take the IBM System z Mastery test

Preparing to take an IBM System z test involves several simple steps:

1. Go to ibm.com/certify and select “Mastery test” under Related links.
2. Under the IBM System z category, select test number 000-Z05.
3. Read through the overview and test objective information. Prepare for the test by following the recommended test preparations.
4. To register to take the System z Mastery test or to locate the test center nearest you, visit Prometric at the following site to start building your credentials today!

www.prometric.com/IBM/default.htm

Stay current and stay connected with System z skills through Social media

The IBM Academic Initiative System z has many social media sites you can follow to stay current on skills news and opportunities. Be it through your Twitter feed or Facebook account, you can stay connected. There is a Facebook page that you can “Like” to be updated with all the latest news and events:

www.facebook.com/EnterpriseAcademicInitiative

In addition, there is a Facebook page for the Master the Mainframe contest, held annually in the fall in North America. You can “Like” the page here:

www.facebook.com/MasterTheMainframe

Many of the student mainframe contests running worldwide have their own Facebook pages and can be found linked off of the Master the Mainframe Facebook page. There are various System z pages, such as Systemz Mainframe (www.facebook.com/IBMSystemz), IBM System z on Campus, and IBM Mainframe Professionals.

Tweeters can easily find the Academic Initiative System z on our account:

www.twitter.com/IBM_AI_System_z

While similar to the Facebook page, we’ll be posting interesting tidbits from tours and events to keep you in the moment with some of the activities we host. You can also follow these feeds:

- General System z Twitter feed:
  @ibm_system_z
- Other important announcements and interesting information regarding the system:
  @newtosystemz

System z related social media also exists on LinkedIn through groups like the “System z Advocates”, “Mainframe Experts Network”, “IBM Mainframe”, and “DB2 Professionals”.

If videos interest you, the YouTube channel for IBM System z hosts a number of interesting videos on personal experiences, hardware/software history, announcements, and more:

www.youtube.com/user/IBMSystemz

Be sure to connect with us!

Acknowledgments

We’d like to thank Alan June (Manager, System z Program Management) and Don Resnik (IBM System z Client Skills Leader) for their contributions to this article.
n the beginning...can you picture the Java situation on OS/390® in 1997 and 1998? Would seasoned MVS and OS/390 customers use this new language and environment that found roots in a distributed culture? Likewise, would programmers and customers from heterogeneous and UNIX machine cultures adapt to machines with JCL, MVS data sets, batch processing, and SMP/E installs? Would IBM improve performance and deliver Java SDKs on OS/390, and then later on z/OS with the same adherence to industry compatibility and timetables as the rest of the IBM Java platforms? Would these SDKs be able to exploit the unique characteristics of IBM System z machines and z/OS?

Yes!

Over the years, z/OS Hot Topics has published periodic updates on z/OS Java SDK specifics. Instead, this article steps back to highlight the overall journey of over a decade of Java deliveries on z/OS.

Decade of continuous products

As Figure 1 shows, IBM has steadily delivered new Java SDKs on z/OS designed to be compliant with Java Standard Edition (SE) application programming interfaces and that include new improvements in base technology, performance, and exploitation of System z hardware for over a decade.

Excellence in technology

With an ever-increasing and significant investment in Java-based assets across its software portfolio, Java has become essential to IBM’s software strategy and to many of its customers’ roadmaps. The z/OS platform is certainly no exception to this trend. Over the last twelve years, Java has become pervasive and inherent across a diverse set of execution environments on z/OS, establishing itself as a first class language for implementing business logic for transaction processing environments such as CICS, IMS, DB2, and WebSphere Application Server.

Batch options

With the JZOS toolkit, WebSphere Compute Grid, and in z/OS Version 1 Release 13 (V1R13) the recent addition of the z/OS Batch Runtime (the Java COBOL batch container), customers have a wide variety of options for using Java-based assets for batch processing. Many of these environments allow for ease of extension of existing COBOL assets with new Java assets and are designed to enable re-use across online transactional processing (OLTP) and batch spaces.

Extensions, business logic, innovation

As a result of the strategic value of Java, IBM continued investment in Java technology for z/OS. For instance, the IBM SDK includes extensions to allow access and exploitation of z/OS services such as Security Server Resource Access Control Facility (RACF), System Management Facility (SMF), and Workload Manager (WLM). As such, IBM Java on z/OS provides all the benefits of allowing for platform agnostic business logic, while also allowing for deep exploitation of the qualities-of-service for z/OS deployments. In addition, IBM completely re-engineered its
Java Runtime Environment (JRE) to accelerate innovation with its collection of the IBM J9 Java virtual machine (JVM) and the IBM Testarossa Just-in-Time (TR-JIT) compiler. It is worth noting that IBM’s first J9-based JSE delivery was the z/OS IBM 142 64-bit SDK demonstrating more innovation for Java on z/OS.

With IBM Java5, the J9 JVM and JIT compiler became the common runtime environment across IBM JRE, JSE and JEE solutions. By providing a single point of innovation, the technology has reached unmatched levels of robustness while accelerating IBM’s ability to bring features such as full-speed-debug, ahead-of-time compilation, shared-classes and compressed-references to market in a timely and effective manner.

Garbage collection
The IBM J9 Virtual Machine is a clean-room implementation of the Java virtual machine specification. Its design is lightweight, scalable, and versatile and it’s packaged with the IBM J9 Virtual Machine is a state-of-the-art garbage collector. The garbage collection offers a diverse set of policies to best address the requirements and constraints of specific applications. For instance, the generational concurrent garbage collection policy offers an innovative solution for applications that create many short lived objects. This policy has been proven essential for enabling Java-based transaction processing environments. More recently, the IBM J9 Virtual Machine has continued to innovate with technology like the new balanced GC policy, which is intended to address trends for ever-growing Java heaps.

IBM TR-JIT
The IBM TR-JIT is a powerful optimization engine with an established track record for industry leading performance. Besides being equipped with a complete set of traditional, object-oriented and Java-based optimizations, TR-JIT has performance aggressive, deep optimizations on IBM System z. With optimizations like idiom recognition, which enable exploitation of some powerful z architecture, an instruction scheduler that re-orders instructions to get optimal processing bandwidth out of the z processor pipeline, and a specialized register allocator, the TR-JIT is a fit-for-purpose optimization engine for System z.

Growing with you
With the System z move to a high-frequency pipeline design, a growing challenge around the discrepancy in cache and core speeds was observed. Compounding this issue was a clear trend showing application footprints out-growing the 31-bit address space on z/OS, implying that the migration to 64-bit was becoming inevitable, yet also more costly than ever. IBM Java6 64-bit offerings included an innovative new feature that allowed the compression of 64-bit object references to a 32-bit representation. This feature was critical to mitigating the increase in footprint and throughput that was impeding 64-bit migrations. Also beginning with z10™, the JVM was an early exploiter of 1 MB large pages reducing memory management overhead to provide relief for 64-bit application performance.

As of System z10™, the z processor includes a hardware implementation of the IEEE 754r decimal floating point (DFP) standard. The IBM SDK6 BigDecimal implementation was extended to exploit DFP hardware to accelerate and improve decimal arithmetic primitives in SDK6.

Innovate with architecture
With the System zEnterprise 196, IBM Java V6.0.1 continued innovation with aggressive use of new architecture features such as the facilities for:
- High word
- Interlock-update
- Non-destructive operand
- Conditional load/store

The new JVM also added an instruction scheduler to optimize use of the z196 out-of-order pipeline. When combined with features for improved path-length, data-cache, and instruction-cache locality, SDK601 has been shown in some settings to provide significant up-lift for Java running on System z.

Performance
Beyond innovation in the SDKs, Java performance has also been a focus area for the hardware. With the pipeline evolution super-scalar to high-frequency, and most recently out-of-order execution offered in System z, Java on z/OS has seen a trend for significant improvements in Java performance with each new processor generation. The large caches and translation look-aside buffers (TLB) on System z have also offered a critical performance advantage for Java. In addition, new architectural features...
such as traps, large-pages, pre-fetching, and the interlock-update facilities were co-designed, and added directly in support of Java.

Pricing
When considering pricing advantages provided by specialty engines such as IBM System z Application Assist Processor (zAAP) and IBM System z Integrated Information Processor (zIIP), the improvements in performance offered by innovation in the hardware and SDK continue to have had a dramatic impact on the price-performance of Java on z/OS.

Your comments
From the very beginning, you, the customer, had requirements that went beyond high-performing deliveries designed to be compatible with Java SE APIs. For one thing, many of you wanted to use security functions like cryptography under Java to necessitate unique interaction with the z/OS Integrated Cryptographic Service Facility (ICSF) and RACF. Some of you wanted to access legacy MVS data sets, use batch processing, or both.

Over the years, IBM answered by providing Java security function and enhancements unique to z/OS. In 1998, you met the Java Record IO (JRIO) function to access MVS data sets. In 2006, came the addition of JZOS. Since then, JZOS improvements have been continuous. Now, JZOS has supplanted JRIO, which will be removed in future Java SDKs. (See “New and improved: Migrating to the latest z/OS Java APIs” on page 44.)

SDK7 highlights
SDK7 is the latest delivery in the family of IBM Java. SDK7 is an implementation of the Java7 standard that includes:
• Meeting the demands of large data algorithms through new I/O support,
• Enabling better scalability and throughput on large multi-core systems like System z using concurrency libraries
• Supporting dynamic language to enable re-use of the Java runtime as a single point of innovation for other dynamic languages
• Simplify day-to-day programming tasks with a large set of new syntactic features

Like SDK601, SDK7 comes with the J9R26 runtime, which:
• Uses a significant set of z196 instructions
• Includes a large set of new optimization features
• Provides the new balanced GC policy

As a result, SDK7 has the potential for improved throughput for many applications while also reducing footprint and startup time.

New JZOS services for Java to inter-operate with WLM, zLogStream, and DSCB and extended access volumes continue the trend of enabling z/OS services. And IBMJCECCA has been extended to include exploitation of the AES security keys, while new RAS features improve its availability.

Not the end
Java on z/OS is critically important to IBM, and we know you expect the next decade to continue the record of the past!

Find out more
• z/OS Java web site: ibm.com/systems/z/OS/zos/tools/java/
• z/OS Java SDK products (including formal names): ibm.com/systems/z/OS/zos/tools/java/products/allproducts.html
• z/OS WebSphere Performance Paper: ibm.com/PartnerWorld/wps/servlet/ContentHandler/whitepaper/systemz/java_websphere/performance...
New and improved! Migrating to the latest z/OS Java APIs

BY GINA YUAN, CLARK GOODRICH, AND MICHAEL WANG

A lot has happened to z/OS Java over the years. In this article, you will find information on the recent “deprecation” of Java Record IO (JRIO) APIs, how to use the JRIO Tracker to find JRIO usage at run time, benefits of JZOS offerings, the high performance benefits of a new set of JZOS APIs for working with z/OS sequential data sets, and guidance on migrating from deprecated JRIO APIs to the latest JZOS APIs. Whew! That’s a lot of good stuff happening.

For a summary of overall z/OS Java functions, see “z/OS Java: Over a decade of excellence and enhancements!” on p. 41.

z/OS Java SDK 7 products: the last releases to support JRIO
The IBM Java Record I/O (JRIO) component of the z/OS Java SDK products is deprecated as of z/OS Java Software Development Kit (SDK) Technology Edition Version 6, Release 0, Modification 1 (z/OS Java SDK 6.0.1) products. IBM z/OS Java SDK 7 — according to the product announcements of October 4, 2011 — is planned to be the last release to support the JRIO component. With years of enhancements, the IBM JZOS Batch Toolkit (JZOS) component of the z/OS Java SDK products has become the z/OS Java strategic direction for accessing z/OS legacy data sets. JZOS not only includes equivalent functions as JRIO, but also provides new, additional capabilities that are not available in JRIO. Customers and ISVs are strongly recommended to migrate existing Java applications to and develop new applications using JZOS.

z/OS Java deliveries in 2012 have added major enhancements for z/OS customers to migrate to JZOS and access z/OS data sets. A new tool, JRIO Tracker, which was introduced in SDK 6.0.1 SR 1 and SDK 7 SR 1, was developed specifically to help users locate applications that use JRIO and make it easier for them to plan migration to JZOS in time for JRIO withdrawal in a future release. As the z/OS Java strategic direction, JZOS continues to be actively enhanced with new and improved functions. The most recent enhancement is a new set of APIs, also shipped with SDK 6.0.1 SR 1 and SDK 7 SR 1. These APIs improve performance when working with sequential data set I/O.

The JRIO Tracker and sample migration code
The JRIO Tracker, part of the JRIO component since z/OS SDK 6.0.1 SR1 and SDK 7 SR1, was developed to help users determine whether their applications use JRIO. The tool helps users decide if migration actions to the JZOS component will be necessary and identifies affected workloads.

The tool provides two approaches to track instances of JRIO applications. The primary approach is to use the z/OS Console ID Tracking facility for console reports. The supplemental approach is to use a new JRIO tracking command option, -DRIOJATRACK, when you need to log additional information through the UNIX log.

Using the z/OS Console ID Tracking Facility for instance reporting
To determine whether your applications use JRIO, you can turn on the z/OS Console ID Tracking facility and then run your applications. Any applications that call JRIO during the run are recorded and can be displayed on the z/OS operation console. Each application instance shown on the console includes a tracking information field that consists of the name of the application that uses JRIO, the line number where the application makes the first call to JRIO, and the name of the application class that makes the immediate call to JRIO. The console data also displays the occurrence frequency of each application instance.
The z/OS Console Tracker discards any tracking information for an instance entry that exceeds 28 bytes and appends an asterisk (*) to truncated tracking information to alert users. To allow you to view the complete tracking information without console restriction, a supplemental approach is provided, as described in the following section.

Using a JRIO tracking command to get more information

To obtain additional information not shown on the console display, you can call the application with a new JRIO-specific command option:

-DRIOJATRACK

The result is that complete tracking information without truncation is written to a UNIX file with a default file name or a file name of your choice. The log file will also contain the calling stack of the invoked instance, including all calls in the call stack before the execution point when the instance was reported.

For detailed usage and examples about the JRIO Tracker, see the IBM Java Record I/O (JRIO) to IBM JZOS Batch Toolkit Migration and Sample Code website:

ibm.com/systems/z/os/zos/tools/java/products/jzos/sdk601_jrio2jzos_mig.html

This website also includes a JZOS JRIO-Equivalent Sample Code section that has sample code for you to download. With assistance from these samples, you will find that migrating from JRIO to JZOS is fairly straightforward.

JZOS benefits

JZOS makes it easy for customers to access key z/OS system services directly from their Java applications. Consisting of a set of Java classes and additional C, C++, and Assembler code, JZOS provides a thin layer on top of key z/OS system services. Customers who are familiar with the existing z/OS system services APIs will feel right at home using JZOS to accomplish those same tasks in Java.

JZOS has an extensive list of features, which include various methods for you to perform traditional data set access on z/OS and provide many additional benefits over JRIO. These benefits include, but are not limited to:

- New APIs that enable high-speed record mode data set I/O for z/OS sequential data sets, with performance comparable to COBOL
- A thin wrapper for z/OS C library I/O functions that support all file open modes and record formats supported by the C I/O library
- A factory class for creating portable readers and writers to text files, including MVS data sets
- APIs for allocating, deleting, and renaming MVS data sets
- A return-code-passing capability for improved I/O error and exception reporting
- Dynamic allocation of Virtual Storage Access Method (VSAM) and non-VSAM data sets
- Access method services (IDCAMS) commands to create and delete VSAM data sets, list catalog entries with LISTCAT, and more
- Functions such as LOCATE and OBTAIN to find data set information, search a catalog, read partitioned data set (PDS) directories, and log stream data support
- APIs to submit z/OS batch jobs

And for even more information about MVS data set I/O with JZOS, see the “MVS Data Set I/O” section in Chapter 4 of the JZOS Batch Launcher and Toolkit Installation and User’s Guide, SA23-2245.

New JZOS APIs with performance enhancements

The most recent enhancement to JZOS is a set of new APIs added in Java SDK 6.0.1 SR 1 and SDK 7 SR 1 for z/OS that dramatically improve data set I/O performance when interacting with traditional sequential data sets on z/OS.

This new set of APIs is in addition to the already existing com.ibm.jzos.ZFile API used for accessing z/OS data sets. The ZFile API wraps z/OS C library I/O functions and can be used for data set I/O in stream mode or record mode. It also supports all file open modes and record formats supported by the C I/O library.

With the introduction of this new set of data set I/O APIs, JZOS is now capable of record-level I/O to z/OS sequential data sets with throughput and processor (CPU) costs similar to COBOL or Assembler. This is a huge benefit for customers who are updating their applications from COBOL to Java, because they no longer have to sacrifice data set I/O performance when migrating to Java.

The new APIs, com.ibm.jzos.RecordReader and com.ibm.jzos.RecordWriter, wrap native z/OS basic sequential access method (BSAM) for data set interaction. Use them whenever

Whew, again!! That’s quite a list with a lot of good functions!
very high speed I/O is required by the Java application. The performance gains can be better realized when the data set has a large blocking factor.

Some restrictions
There are a few restrictions related to the new high speed I/O APIs. Only sequential data sets benefit from improved I/O performance. There are no benefits for other data set types, such as VSAM data sets. In addition, writing to VSAM data sets is not supported through the new APIs, and tape data sets are also not supported.

Some guidelines
To help customers decide when to use the various I/O APIs for data set interaction, here are some guidelines on JZOS I/O API usage:

- For sequential record mode access to binary and text data sets, use the new RecordReader and RecordWriter APIs. Use ZFile in cases where applications require functions that are not provided by these APIs (positioning, for example.).
- For generalized access to VSAM data sets (KSDS, RRDS, or ESDS), use ZFile.
- For POSIX-file-stream-style access to MVS data sets, use ZFile, or the simpler and more platform-independent FileFactory if only text mode access is required.

Performance data
The graph in Figure 1 shows the improvement (by percentage) using the new JZOS RecordReader and RecordWriter APIs as compared to existing JZOS and JRIO APIs. This test performed sequential read and write operations to a single data set with a 27,920-byte block size, an 80-byte logical record length (LRECL), and 100,000 records:

The first two columns in this graph show improvements of 16% wall clock time and 89% CPU time when using RecordReader as compared to using ZFile READ in a multithreaded environment.

In other tests using sequential data sets with block size of 800 bytes or smaller, a wall clock improvement of between 40% and 49% was measured using RecordReader and RecordWriter versus other existing Java APIs. CPU time improvement was best for data sets with large block sizes when using RecordReader and RecordWriter.

Summary
IBM z/OS Java SDK 7 is planned to be the last release to support the JRIO component. JZOS has become the z/OS Java strategic direction for accessing z/OS legacy data sets. In order to make your migration to JZOS as simple as possible, the JRIO Tracker, introduced in z/OS SDK 6.0.1 SR1 and SDK 7 SR1, will help you determine if your applications use JRIO. z/OS Java SDK 6.0.1 SR 1 and SDK 7 SR 1 also introduced new JZOS APIs, com.ibm.jzos.RecordReader, and com.ibm.jzos.RecordWriter, which make Java a competitive alternative to COBOL in reading from and writing to z/OS sequential data sets. Now might be a good time to consider converting COBOL programs to Java.

Given the many additional benefits of JZOS over JRIO, and the recent performance enhancement to JZOS with z/OS legacy sequential data set I/O, it’s definitely a smart move to JZOS, with everything it has to offer!

References
- z/OS Java website: ibm.com/systems/z/OS/zos/tools/java/

![Percent improvement using new JZOS BSAM RecordReader and BSAM RecordWriter versus existing JZOS and JRIO APIs](image)

Figure 1. Using the JZOS RecordReader and RecordWriter as compared to previous JZOS and JRIO APIs
SAM Record Level Sharing (RLS) lets you do locking on a more granular level, which can drastically reduce data set contention for frequently updated data sets in a Parallel Sysplex, allowing higher I/O rates to them. Using RLS, you can share data across z/OS systems in a Parallel Sysplex. This data can reside in local virtual buffers, the coupling facility (CF) cache structure, or on DASD. RLS supports a variety of options and parameters that you can use to control resources, and determine where and how much data to store. Further, you can manage RLS data to help tune applications for better performance. When you understand the basics, managing the RLS data is as easy as 1-2-3!

Managing local virtual buffers
To make it easier to manage how much virtual storage you want RLS buffers to use on each system across a Parallel Sysplex, you can specify the following parameters in the IGDSMSxx parmlib member:

• To set the amount of below-the-bar 31-bit SMSVSAM data space storage to be used for RLS buffers, specify the RLS_MAX_POOL_SIZE parameter.
• To specify how much virtual storage is to be used for above-the-bar 64-bit buffers, specify the RLSAboveTheBarMaxPoolSize parameter.

To maintain buffers in the pool size specified, RLS Buffer Management Facility (BMF) uses a Least Recently Used (LRU) approach. Aged buffers are freed to ensure buffers are available for new CIs to be read in. (Buffer age is the time since the CI in the buffer was referenced.) To avoid buffers being paged out, you must ensure there is enough real storage to back up the total amount of buffer pool sizes requested. A paged-out buffer is immediately freed by the LRU.

The LRU algorithm closely monitors RLS buffer pools. Pools of different buffer sizes grow and shrink dynamically as applications request data. That means that no request fails because of unavailable buffers. Instead, buffers can be stolen and reused, regardless of the buffer age, to satisfy a request’s need for buffers. In the rare case when a buffer steal is not possible, the request is put to sleep until the LRU process can run and reclaim buffers.

Data spaces are limited to a maximum of 2 GB. Up to 1728 MB of the SMSVSAM data space is available for buffers with the remainder reserved for control information. RLS must reclaim storage allocated for buffers if the data space storage becomes constrained. Processing spikes can result in pools that temporarily grow beyond the amount of storage specified.

IBM recommends that RLS_MAX_POOL_SIZE is no more than 850 MB, allowing 680 MB (80% of 850 MB) of buffers to reside in the 31-bit pool for normal condition, and for twice the amount (200% of 850 MB) of storage available before reaching the limit of 1728 MB. Keeping the 31-bit pool specification close to 680 MB can avoid performance slowdowns during spikes.

RLS 64-bit buffer pools use buffers above-the-bar in the SMSVSAM address space. To enable 64-bit buffering, both of the following attributes must be set:
• Set the DATACLAS attribute RLS ABOVE THE BAR to YES (the default is NO).
• Set the RlsAboveTheBarMax-PoolSize to 300 MB - 2 terabytes (the default is 0). The largest amount that IBM has tested is 65 GB, and IBM recommends using 64-bit buffer pools.

The buffer sizes that are used by RLS are 2 KB and 4 KB - 32 KB in 4 KB increments. The buffer sizes are only a subset of the possible CI sizes. Each buffer contains a single control interval (CI) of data. Data sets are assigned to a pool of buffers with the best-fit buffer sizes. When a buffer size exceeds a CI size, the excess space becomes unusable virtual storage. It is important to keep this in mind when considering CI size and buffer size.

**Coupling facility cache structure**
A key resource used by RLS is the coupling facility cache. A cache structure contains directory entries and data elements. The directory entries are used to keep track of the validity of CIs that are stored in virtual buffers on any system in the Parallel Sysplex. The data elements are for data CIs that can be quickly read into a virtual buffer.

RLS provides multiple levels of caching. The IGDSMSxx parmib member RLS_MaxCfFeatureLevel parameter can be one of two settings, A or Z (the default being Z). The feature level is a Parallel Sysplex-wide setting established when RLS is initialized on the first system.

• Feature level Z does not allow greater than 4 KB caching. This implies no CI larger than 4 KB can be retrieved from the CF cache structure. Z is a good option when data is not shared across systems.
• Feature level A allows greater than 4 KB caching and honors the DATACLAS value RLS CF Cache. When RLS CF Cache is ALL (the default), all data and index CIs are cached. When RLS CF Cache is set to NONE, only index CIs are kept in the cache. Index CIs are referenced far more frequently than data CIs. The

**UPDATEONLY value results in both index and data CIs being cached when updated. Feature level A might require larger CF cache structures but performance improves, especially when data is shared across systems.**

You can define multiple RLS cache structures. The cache structure size is another important tuning value to consider when setting up your environment. Directory entries and data elements in the CF cache structure can be reclaimed when cache space becomes constrained. The ideal cache structure size for minimizing DASD reads and maximizing performance is the sum of the local virtual buffer pool sizes from each system in the Parallel Sysplex. You can look in the SMF 42 records, under RLS BMF False Invalids, to see if the cache size is too small.

**How it all fits together**
The first retrieval of any CI in the Parallel Sysplex reads the CI from DASD into a virtual buffer. A directory entry is created in the coupling facility cache structure. Depending on the specified caching option parameter in the DATACLAS, the CI might also be stored in the CF cache. If another system references the same CI, RLS tries to retrieve the CI from the CF cache, which is faster than reading from DASD.

RLS looks for a CI in a virtual buffer first. When a CI is modified, the contents are written to DASD, the cache is updated to reflect the change, and virtual buffers on sharing systems are invalidated. If one of the sharing systems references it, the CI must be reread. An RLS BMF-read hit occurs when a referenced CI is retrieved from a valid virtual buffer. A Sysplex Cache Manager (SCM) read hit occurs when a CI is found in the CF Cache. A DASD read occurs when the CI must be read from DASD.

A false invalid happens when a CI is found in the local buffers, but the CI validity cannot be confirmed by the CF cache. This is usually because CF cache is reclaimed. The system reclaims the existing CF cache directory entry or data elements to satisfy other CIs. False invalids are more expensive because a CI must be reread from DASD even though a valid copy might exist in virtual storage. A high value in this field of the SMF 42 records might indicate that the cache structure is too small and cache structure resources are being reclaimed often.

Not all data is created equal. You can give priority access to the data that you want most. Keep high-value data in a large cache structure by using the caching option ALL. You might open high-value clusters first after RLS comes up so that permanently fixed pages are used for that buffer size. Test data might be assigned to a smaller cache by using the NONE or UPDATEONLY options.

**SMF Type 42 Records**
RLS reports statistics by using SMF type 42 subtypes 15 - 19 records. They provide valuable information on virtual buffer pools and CF cache.

Subtype 19 records give important feedback on how RLS BMF is performing. You can ascertain how frequently BMF reclaims buffers in different modes. Parallel Sysplex-wide numbers for total BMF requests, BMF read hits, and other valuable statistics are available for analysis.

Subtype 18 has information on the CF cache structures, while subtype 16 contains statistics at a data set level. Values such as percentage for BMF read hit, SCM read hit and DASD read, as well as the total number of requests, can help compare how different data sets are performing. Subtype 15 provides similar information at a STORAGE CLASS level.

RLS allows sharing of large quantities of data across many z/OS systems while providing full read and write integrity for all. Many factors affect performance. RLS gives you options to control the allocation of key resources. Now it’s time to put your best foot forward and make RLS perform like a champ.

**Acknowledgements**
Thanks to Terri Menendez, Larry Law, and Janet Barton of VSAM RLS Development and Test for their contributions to this article.
Ask Mr. Catalog

What’s new for catalog in z/OS V1R13?

BY STEPHEN BRANCH, PATRICA CHOI, AND ROHAN KURANE

Many of you have asked what’s new in z/OS V1R13 for catalogs, and Mr. Catalog is here to answer.

In the last installment of “Ask Mr. Catalog”, we told you how you could let your aliases grow using a new catalog feature in z/OS V1R13. But there’s more! Every release, we look through our marketing requirements and do our best to address your concerns. Here are a couple of new functions we added for z/OS V1R13:

Catalog parmlib member

Over time, we’ve added new parameters to the MODIFY CATALOG system command to let you tailor the Catalog Address Space (CAS) to your needs.

We added new function to the MODIFY command because:

• We ran out of room in SYS1.NUCLEUS member SYSCATxx and SYS1.PARMLIB member LOADxx where you can specify CAS settings. The system allows only one 80-character image for either member.
• Use of the LOADxx member precludes the use of the SYSCATxx member.
• Changes to SYSCATxx and LOADxx require an IPL.

Now, in z/OS V1R13, we introduce a new parmlib member, IGGCATxx, to address these limitations and let you specify CAS settings. You can specify the IGGCATxx parmlib member or members you want the system to use in new IEASYSxx system parameter CATALOG=xx. The xx suffix can be any 2 alphanumeric or national characters. The default is 00. You can specify multiple suffixes in the CATALOG= system parameter, with values enclosed in parentheses and separated by commas.

The Catalog component searches SYS1.PARMLIB and its concatenation for each member specified in the CATALOG= system parameter and processes it. If the system cannot find a specified member, the system issues a message and continues to search for the next member in the list.

When you specify multiple suffixes, the system processes IGGCATxx members in the order specified, with the last member’s contents taking precedence. If you do not specify any IGGCATxx members on the CATALOG= parameter and the system finds no IGGCAT00 member either, the system uses the default parameter values.

To display the IGGCATxx parmlib members specified on the system parameter CATALOG=, issue the following command:

DISPLAY IPLINFO, CATALOG

Unlike the MODIFY CATALOG command, which uses positional fields, IGGCATxx uses parameters. You can specify these parameters in any order. If a parameter is specified more than once, the last instance takes precedence. IGGCATxx parameters take precedence over entries specified in both the LOADxx and SYSCATxx members.

Over time, we’ve added new parameters to the MODIFY CATALOG system command to let you tailor the Catalog Address Space (CAS) to your needs.
The IGGCATxx parameters are processed both at IPL and when CAS is restarted.

For complete information on IGGCATxx, including the syntax rules, see z/OS MVS Initialization and Tuning Reference.

**VVDS Expansion**

The VSAM Volume Data Set (VVDS) is designed to contain:

- Physical information about VSAM data sets in records called VSAM volume records (VVRs)
- SMS construct information for SAM data sets in records called non-VSAM volume records (NVRs).

There is one VVDS per DASD volume which has a VSAM data set or is SMS-managed.

Until z/OS V1R13, the maximum size of a VVDS was 65535 Control Intervals (CIs), which is 364 cylinders. Because VVRs and NVRs are variable in length, it was difficult to predict how many data sets a VVDS could hold. Still, the past amount was large enough for most volumes. But with the increase in the size of volumes, especially with extended addressable volumes (EAVs), there can be more data sets defined on a volume, so this amount is no longer always sufficient.

To address this issue, in z/OS V1R13 we increased the number of CIs you can have in a VVDS by a factor of 16, to 1,048,575 CIs. For an existing VVDS, the expansion takes effect the first time the VVDS is opened on a z/OS V1R13 system. This allows your existing VVDS on a z/OS V1R13 to extend beyond 65535 CIs automatically. There is nothing else you have to do – it’s just that easy!

If you define a new VVDS explicitly on a z/OS V1 R13 system, you can define it with a size up to 5825 cylinders.

Systems lower than z/OSR13 can access VVRs and NVRs beyond 65535 CIs, with tolerance APAR OA36299, but will not be able to either define or extend VVDSs to the higher CI level, regardless of what system level created them.

The following table describes the new IGGCATxx parameters and their meaning:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVDSSPACE(pri, sec)</td>
<td>Specifies the number of primary and secondary space for VVDS implicit defines. The default is 10 tracks for both values.</td>
</tr>
<tr>
<td>TASKMAX(n)</td>
<td>Where n is the maximum number of user-initiated service tasks that can be executing at any given time. The default is 180; minimum is 24, maximum is 360. The value for TASKMAX cannot be higher than 90% of the CAS Service Task Table Size, TASKTABLESIZe currently specifiable in the SYS1. NUCleUS(SYSCATxx). (See APAr OA38822.)</td>
</tr>
<tr>
<td>NOTIFYEXTENT(n)</td>
<td>Where n is the percentage threshold to warn when a catalog is getting full. Default is 80%.</td>
</tr>
<tr>
<td>DELFORCEWNG(YES/NO)</td>
<td>Enables/Disables the ability to issue the IDC1997I or IDC1998I warning message.</td>
</tr>
<tr>
<td>DSNCHECK(YES/NO)</td>
<td>Enables/Disables syntax checking on names being added to a Catalog.</td>
</tr>
<tr>
<td>SYMREC(YES/NO)</td>
<td>Enables/Disables the ability to create a SYMREC.</td>
</tr>
<tr>
<td>UPDTFAIL(YES/NO)</td>
<td>Enables/Disables the message IEC390I when a VSAM update request against a catalog has been abnormally terminated.</td>
</tr>
<tr>
<td>VVRCHECK(YES/NO)</td>
<td>Enables/Disables enhanced VVR checking on VVDS I/O.</td>
</tr>
<tr>
<td>DELRECOVWNG(YES/NO)</td>
<td>Enables/Disables the ability to issue the IDC1999I warning message when issuing the DELETE UCAT RECOVERY command.</td>
</tr>
<tr>
<td>EXTENDEDALIAS(YES/NO)</td>
<td>Enables/Disables the ability to create extension records for user catalog aliases on the current system.</td>
</tr>
</tbody>
</table>

Table 1. IGGCATxx parameters
Have your tape allocations gone astray? Tape allocations unevenly distributed within your environment? Do you wish you could direct your scratch allocations to a particular TS7700 Virtualization Engine in a multi-cluster grid configuration? And lastly, do you wish you could direct your system-managed tape request to a specific device? If you answered yes to even one of these questions, you’ll be excited to hear about our recent enhancements!

Before we dive into the enhancements, let’s take a look at the basics of SMStape and the IBM TS7700 Virtualization Engine.

The basics of SMStape

You can use System-managed tape (SMStape) support to manage IBM tape libraries, including:

- IBM TS7700 Virtualization Engine, for virtual tape
- IBM System Storage TS3500 Tape Library, for physical tape
- Manual Tape Library, for stand-alone tape devices

With SMStape, you can establish policies through SMS and the automatic class selection (ACS) routines that enable system-managed tape allocation requests to be directed to one or more tape libraries.

All about the IBM TS7700 Virtualization Engine

The IBM TS7700 Virtualization Engine is a product family consisting of:

- TS7740 Virtualization Engine for disk cache with physical tape
- TS7720 Virtualization Engine for disk cache only with no physical tape

Using TCP/IP, you can interconnect one or more TS7700 Virtualization Engines (or a mix referred to as a hybrid configuration) to form a multicluster grid configuration for local, metro, and remote data availability requirements. That means that there can be copies of the data in one or more of the clusters in the grid, accessible from any of the clusters. The interconnected TS7700 Virtualization Engines, which are also called distributed libraries or clusters, appear to the host as a single library image; a composite library. In TS7700 Virtualization Engine terms, a library refers to a composite library because MVS allocation has no direct knowledge of the underlying clusters, or distributed libraries. See Figure 1:

Using your SMStape policies and ACS routines, the tape storage groups that are assigned to an allocation request determine which tape libraries are eligible for a scratch allocation. By default, MVS device allocation in a JES2 environment first randomizes allocation across all eligible libraries and then, after a library is selected, randomizes allocation on the eligible devices within that library. This default load-balancing algorithm is referred to as “EQUAL” because it tends to evenly distribute the workload across multiple libraries, no matter how many tape devices there are in each library.

Since z/OS V1R11, there’s a new load-balancing option that randomizes your scratch allocations across all eligible devices. This new algorithm is called “BYDEVICES”.

Figure 1. TS7700 virtualization engine
Both the EQUAL and BYDEVICES balancing options apply to any tape library in a JES2 environment, whether automated, manual, or virtual.

Choosing between EQUAL or BYDEVICES
Choose EQUAL if the libraries that are eligible for the scratch allocation contain an equal number of online devices. For example, if two libraries are eligible and each library has 128 devices, each library receives approximately half of the scratch allocations. Likewise, if one of the libraries has 128 devices and the other library has 256 devices, each of the libraries still receives approximately half of the scratch allocations regardless of the number of online devices.

Choose BYDEVICES when the libraries that are eligible for scratch allocation have a different number of online devices, because the BYDEVICES algorithm randomizes your scratch allocations across all eligible libraries. Let's look at an example. Imagine that you have two libraries, one with 128 devices and another with 256 devices. If you use the BYDEVICES algorithm, over time the library that has 128 devices receive approximately 1/3 of the scratch allocations and the library that has 256 devices receive approximately 2/3 of the scratch allocations.

Using the BYDEVICES algorithm can also help you get a better workload distribution between the clusters (distributed libraries) in a TS7700 multicloud configuration. Switch to the BYDEVICES algorithm if you have devices that are online in multiple clusters and your scratch allocations are not being distributed across the clusters in the grid.

Enabling the BYDEVICES algorithm
You can enable the BYDEVICES algorithm in one of the following ways:

- To enable BYDEVICES during IPL, specify the SYSTEM TAPELIB_PREF(BYDEVICES) parameter in the ALLOCxx parmlib member.
- For systems at z/OS V1R11 and later, you can dynamically enable BYDEVICES between IPLs by using the SETALLOC command:

```
SETALLOC SYSTEM,TAPELIB_PREF=BYDEVICES
```

You can disable the BYDEVICES algorithm by specifying EQUAL in either of these two methods.

Next we’re going to talk about some allocation assist enhancements that only apply to the TS7700 Virtualization Engine.

Device allocation assistance (DAA)—directing your scratch allocations to a particular TS7700 Virtualization Engine
For specific allocations in a JES2 environment, MVS allocation might receive an ordered list of subsystems from the library. A subsystem represents the devices behind a physical- or logical-tape control unit, and can contain up to 16 devices. Starting with TS7700 Virtualization Engine Release 1.5, the list of subsystems are returned in cluster-preference order, favoring, for example, clusters that have the logical volume in disk cache. MVS allocation attempts to allocate a device from the first subsystem that is returned in the list. If an online device is not available within the first subsystem, MVS allocation moves to the next subsystem in the list and tries again until a device is chosen. But the way MVS Allocation processes the ordered list of subsystems depends on the load balancing option you're using:

- For the default algorithm EQUAL, MVS allocation honors only the first seven subsystems returned in the ordered list. After that, if an eligible device cannot be found, the rest of the subsystems in the list are considered equal.
- The BYDEVICES algorithm removes the subsystem limitation discussed for EQUAL. Use BYDEVICES if you are using the TS7700 Virtualization Engine, especially if you notice that the system repeatedly allocates clusters that are not the best choice.
Device allocation assistance (DAA) was introduced with APAR OA24966 in the JES2 environment.

**Scratch allocation assistance (SAA)**
TS7700 Virtualization Engine Release 2.0 provides additional support for scratch allocations in a JES2 environment. If you have a hybrid configuration of interconnected TS7720 and TS7740 clusters, you can direct specific workloads by using scratch allocation assistance (SAA) as follows:

- For workloads that need fast access, you can direct allocations to the TS7720
- For archive-type workloads, you can direct allocations to the TS7740.

You can also direct workloads in this way without using SAA, by using management class copy policies at the library, but if the allocation request goes to the other cluster, the library uses TCP/IP instead of FICON to send the data between the clusters, which is not as efficient. By using SAA, you can establish policies at the library with a management class option. By using this option, you can specify the clusters to be used for the scratch allocation request, known as candidate clusters. This means that only the specified clusters will be considered for the allocation request.

Unlike DAA, which is enabled by default, you must enable SAA by using the LIBRARY REQUEST command:

```plaintext
LIBRARY REQUEST, libname, SETTING, DEVALLOC, SCRATCH, ENABLE
```

If you enable SAA and request it through the management class policies at the library, the TS7700 then provides a filtered candidate list of clusters to the host. Using this filtered list, MVS allocation then randomly chooses a device from the filtered list (and the filtered lists of other eligible grid configurations) using the load-balancing options.

SAA was introduced with APAR OA32957 in the JES2 environment.

**Demand allocation**
Lastly, we’re going to talk about an enhancement you can use with any system-managed tape library—automated, manual, or virtual. By default, the job control language (JCL) UNIT specification is ignored in any JES environment for system-managed tape requests. This enables the SMS automatic class selection (ACS) routines to direct allocation requests to a set of eligible libraries and devices and enables the system to randomize allocation across the eligible devices. This is the recommended way to use the system-managed tape support.

However, there might be times when it’s better to direct an allocation request to a specific device or set of devices, for example to gather diagnostic trace data. You can specify the JCL keyword SMSHONOR with the UNIT parameter, or through the dynamic-allocation text unit, DALSMSHR. When you specify the SMSHONOR or DALSMSHR keywords, the system will:

- Use the list of eligible devices returned by SMS and the devices that are specified on the UNIT parameter
- Allocated devices using the intersection of the two lists.

If you specify a single device, and that device is one of the eligible devices returned by SMS, the system allocates to the device specified. If you specify an esoteric on the UNIT parameter, the system allocates only the devices that are on both lists. Devices that are not in both lists are ignored.

For more information on the SMSHONOR keyword, see z/OS V1R13 MVS JCL Reference. This support was first introduced in z/OS V1R11.

**Next steps**
You can read more about the new allocation enhancements in the z/OS V1R13 DFSMS Planning, Installation, and Storage Administration Guide for Tape Libraries, SC35-0427-10.
You’ve probably experienced this scenario: You’re sitting at your workstation. It’s the end of the day. The clock is making its way, very slowly, to 5:00pm. Your final task is to start the batch jobs. Suddenly, errors flood the console! Jobs are failing left and right! And to top it off, your phone is ringing off the hook! Looks like it’s going to be a long night. At least you have your trusty take-out menu!

Before you place your dinner order, let’s take a minute to assess the situation. The firestorm before your eyes might actually be a small problem that you can easily fix. This article covers a few common problems related to the DFSMS Catalog component that might seem disastrous, but are actually easily fixed without having to burn the midnight oil.

Common errors

**Moment of Panic (MOP):** IEC331I 20-0 (2908001C)

**Sigh of Relief (SOR):** This error indicates that the catalog reached its maximum size or that it cannot be extended. Either it reached its extent limit or there is not enough space to extend it. To fix this error, complete one of the following steps:

- Define a larger catalog and copy in the entries by using the REPRO command. z/OS R12 increases the catalog size beyond the 4 GB limit.
- Split the catalog.
- If your system is at z/OS V1R11 or earlier, reclaim unused space by running the EXPORT/IMPORT command.
- If your system is at z/OS V1R12 or later, ensure that CA Reclaim is active for the catalog in question.

**MOP:** IEC070I 34

**SOR:** This error indicates that the catalog has reached the 4 GB limit. Split the catalog into two or more smaller catalogs. For systems at z/OS V1R12 and later, you can also redefine the catalog with the Extended Addressability attribute on an Extended Addressable Volume (EAV).

**MOP:** IDC3009I 228-60

**SOR:** This error indicates that Enhanced Catalog Sharing (ECS) is deactivated for a catalog. This problem occurs when the catalog is in mixed-mode sharing. Ensure that the catalog has ECS active on all shared systems.

**MOP:** IDC3009I 50-28

**SOR:** This error indicates that there is no space available to extend the VSAM volume data set (VVDS). You must complete one of the following steps:

- Increase the size of the VTOC by using the ICKDSF EXT VTOC command.
- Use Dynamic Volume Expansion.
- Delete unnecessary data.

**MOP:** IDC3009I 50-30

**SOR:** This error indicates that the VVDS reached the maximum extent of 123. Recover the volume after you define a larger VVDS.
**MOP: IDC3009I 50-32**

**SOR:** This error indicates that the VVDS is full. If you are defining an SMS-managed data set or non-SMS VSAM data set, use a different volume. If you are defining the VVDS, decrease the primary allocation. Also, verify the SMS configurations to ensure that they are best suited for your system. Often, you might have many one-track data sets going to a smaller VVDS, and this setup can cause IDC3009I 50-32 to occur unexpectedly.

**MOP: IDC3009I 50-38**

**SOR:** This error indicates that the DELETE command failed and the VVDS could not be deleted. This error occurs because the volume still contained data sets that are cataloged in the specified CATALOG parameter.

**Other options you can use to save the day**

If you have exhausted all the preceding options and are about to give up, DON’T! IBM offers the following tools to help you fix your catalog quickly.

**Use ICFRU to restore your catalog**

Integrated Catalog Forward Recovery Utility (ICFRU) uses a previously exported catalog and SMF records to restore the catalog to its current state. Using this utility, it’s simple to restore data. However, this utility is only as successful as your initial setup, so ensure that you have the following items before you use the ICFRU utility:

- An error-free BCS backup that was created by using the IDCAMS EXPORT TEMPORARY command
- SMF Type 61,65, and 66 records from all systems that share the catalog
- Consistent SMF timestamps across sharing systems.

See the IBM Redbooks® publication *ICF Catalog Backup and Recovery* for information about performing the forward recovery.

**VVDS Recovery**

VVDS damage can occur in two forms: Structure and Entry.

**Entry Damage**

If you can open the VVDS, but are still receiving errors, chances are that an entry is damaged. To find the problem, issue the IDCAMS DIAGNOSE command against the VVDS. The DIAGNOSE command not only shows what entries are damaged, but also shows how they are damaged. Armed with this information, you can fix the entry.

**Structure Damage**

If you cannot open the VVDS, it most likely has structural damage. A full-volume restore might be necessary. You can dump a non-SMS managed volume separately and then restore it after the full-volume restore to avoid data loss.

To perform the full-volume restore, you must complete the following steps:

1. Vary the volume offline.
2. Unallocate the VVDS by issuing the MODIFY CATALOG,VUNALLOCATE command.
3. Unallocate the catalog by issuing the MODIFY CATALOG,UNALLOCATE command.
4. Perform a full-volume restore operation by using the DFSMShsm RECOVER or DFSMSdss RESTORE command.

Now you possess the knowledge to fix the seemingly catastrophic failures, and now you know how to recover your system from near-fatal errors. With this, you can save the day and—more importantly—have dinner at home and get to sleep at a decent hour!

**Recommended reading**

We recommend keeping the following manuals nearby for reference purposes:

- *DFSMS Managing Catalogs*, SC26-7409
- *DFSMS AMS for Catalogs*, SC26-7394-12
- IBM Redbooks publication, *ICF Catalog Backup and Recovery*, REDP-4212.

We also suggest keeping a copy of the VVDSFIX tool handy. For more information about the VVDSFIX tool, see:

[ibm.com/support/docview.wss?uid=isg3S1000618](http://ibm.com/support/docview.wss?uid=isg3S1000618)

VVDSFIX is a zap tool, so exercise caution when using it.
The world is full of great combinations: Chips and salsa, Simon and Garfunkel, and of course chocolate and anything! With a great combination, the sum is truly greater than the parts.

Here's another great combination to add to the list:

- IBM Health Checker for z/OS and IRRXUTIL, which is the RACF REXX interface.
- IBM Health Checker for z/OS examines your z/OS environment for configuration issues that might cause an outage.
- IRRXUTIL allows you to extract information from the RACF database using the REXX programming environment.

Individually, each is a very useful tool. Is there a way that we can use them together?

Yes there is! You can write your own IBM Health Checker for z/OS checks using System REXX to examine the contents of your RACF database and your RACF system options. POW! This lets you create a check that examines your RACF configuration, including:

- The RACF profiles protecting your resources
- Your system options

Then, your check issues an exception message if it finds that your RACF profiles or RACF settings do not match your expectations.

For basics, see the following information:
- For instructions on how to use the RACF IRRXUTIL interface in the article “This used to be like pulling teeth (but not anymore!)” in z/OS Hot Topics Newsletter Issue 22, March 2010, GA-22-7501.
- For detailed instructions on how to write a System REXX check, see IBM Redpaper “Exploiting the IBM Health Checker for z/OS Infrastructure” (REDP-4590-01).

Bringing it all together

Let’s bring all of this together and show you how to write your own System REXX RACF IBM Health Checker for z/OS check!

Let’s say that we want to write a check that examines the RACF profiles z/OS uses to assign user IDs to started tasks, verifying that all of the user IDs that are assigned have the PROTECTED attribute. The PROTECTED attribute means that the user IDs can’t be logged on using a password or a PassTicket. We want the output of the check to look as follows:

What do we have to do to get this kind of output? Our check must:

1. Establish a connection to IBM Health Checker for z/OS.
2. Extract the profiles from the STARTED class, which define the association between the started task name and the user ID.
3. Extract the user profile using the user ID within the STARTED profile.
4. Look for exception conditions and issue check output messages.
5. At the end of this step, we stop the connection to IBM Health Checker for z/OS.

Let’s look at these steps in a little more detail.

CHECK(IBMSAMPLE,RACF_STARTED_CLASS)
START TIME: 04/25/2011 15:10:58.866140
CHECK DATE: 20061219 CHECK SEVERITY: LOW
VERBOSE MODE: YES

RACF Started Class Report

<table>
<thead>
<tr>
<th>Started Profile</th>
<th>Gen</th>
<th>User</th>
<th>Group</th>
<th>Prot</th>
<th>Trust</th>
<th>Priv</th>
<th>Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRRDPTAB.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>JES2.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>JCOO.*</td>
<td>Yes</td>
<td>PROTECT</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>JKOOP.*</td>
<td>Yes</td>
<td>MARKN</td>
<td>SYS1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RACF.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RSPF1.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RSPFWKSP.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>U RSPF2.*</td>
<td>Yes</td>
<td>NOSUCHID</td>
<td>SYS1</td>
<td>***</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RSPFWKSP.*</td>
<td>Yes</td>
<td>STCUSER</td>
<td>SYS1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>E **</td>
<td>Yes</td>
<td>IBMUSER</td>
<td>SYS1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Low Severity Exception *

RAC260E One or more started tasks were found which had associated user IDs which do not have the protected attribute.

...
1. Establishing a connection

We start by establishing a connection to IBM Health Checker for z/OS and defining a function code of 1 for our check, as shown in Figure 2. We’ll reference this function code later, when we define the check to the IBM Health Checker for z/OS infrastructure.

```plaintext
HZSLSTRT_RC = HZSLSTRT()
IF HZSLSTRT_RC <> 0 THEN
  DO
    SAY "HZSLSTRT RC"  HZSLSTRT_RC
    SAY "HZSLSTRT RSN" HZSLSTRT_RSN
    SAY "HZSLSTRT SYSTEMDIAG" HZSLSTRT_SYSTEMDIAG
    EXIT /* Exit, check cannot be performed */
  END
END
```

2. Extracting the profiles

Now we’re going to extract the profiles from the STARTED class. For example, let’s look at the Process_RACF_STARTED_CLASS_CHECK routine. In this routine, we start by using IRRXUTIL’s EXTRACTN function looping through the STARTED class (Figure 3):

```plaintext
class = 'STARTED'
RACF.R_PROFILE = ' '
RACF.R_GENERIC = 'FALSE'
checkException = "NO"

Do Forever
  myrc= ,
  IRRXUTIL("EXTRACTN",class,RACF.R_PROFILE,"RACF","R_",RACF.R_GENERIC)
  /*------------------------------*/
  /* Check for "end of profiles" IRRXUTIL return code. */
  /*------------------------------*/
  if (word(myrc,1)=12 & word(myrc,2)=12 & word(myrc,3)=4 & ,
      word(myrc,4)=4 & word(myrc,5)=4) then do
    Leave
  end
  else if (Word(myrc,1) <> 0) then do
    /* Any other non-zero IRRXUTIL return code is an error */
    call irrxutil_error
    Say "Class=" class "Profile=" RACF.R_PROFILE
    Say "Generics=" RACF.R_GENERIC
    Say "Started myrc=" myrc
    Leave
  end
```

Figure 2. Establishing a connection to IBM Health Checker for z/OS and defining a function code

Figure 3. Using IRRXUTIL’s EXTRACTN function to extract profiles from the STARTED class
3. Extracting the user profile

Using the user ID that we just extract from the STARTED profile, we now extract the user profile, as shown in Figure 4:

\[
\text{RACF.U_PROFILE} = \text{RACF.R_STDATA.R_USER.1} \\
\text{startedTrustedFlag} = \text{RACF.R_STDATA.R_TRUSTED.1} \\
\text{startedPrivilegedFlag} = \text{RACF.R_STDATA.R_PRIVILEGE.1} \\
\text{startedTraceFlag} = \text{RACF.R_STDATA.R_TRACE.1} \\
\text{startedGroupName} = \text{RACF.R_STDATA.R_GROUP.1} \\
\text{startedGenericFlag} = \text{RACF.R_GENERIC} \\
\text{startedProfileName} = \text{RACF.R_PROFILE} \\
\text{exceptionFlag} = "\" \\
\]

\[
\text{myrc=IRRXUTIL("EXTRACT","USER",RACF.U_PROFILE,"RACF","U")} \\
\text{RACF.R_GENERIC} = \text{startedGenericFlag} \\
\text{RACF.R_PROFILE} = \text{startedProfileName} \\
\]

Figure 4. Extracting the user profile

4. Looking for exception conditions

Now we have all of the information that we need to determine if we have found exception:

- First, we look to see if there is a user profile defined for the user ID in the STARTED profile. If there is not, we flag a “U” exception.
- Next, we look to see if the user ID is a protected user ID, which is one that can’t be logged on using a password, password phrase, or a PassTicket. If the user ID is not protected, we flag an “E” exception. See Figure 5:

\[
/*----------------------------------------*/ 
/* Check for ”no profile found” IRRXUTIL return code. This means */ 
/* that the started profile is referencing a user ID which does */ 
/* not exist. This is an exception. */ 
/*----------------------------------------*/ 
if (word(myrc,1)=12 & word(myrc,2)=12 & word(myrc,3)=4 & , 
word(myrc,4)=4 & word(myrc,5)=4) then do 
exceptionFlag="U" 
end 
else if (Word(myrc,1) <> 0) Then Do

/*----------------------------------------*/ 
/* Any other non-zero IRRXUTIL return code is an error */ 
/*----------------------------------------*/ 
	call irrxutil_error 
	exceptionFlag="U" 
End 

if RACF.U_BASE.U_PROTECTD.1=FALSE then do 
	exceptionFlag="E" 
	checkException= "YES" 
end

Figure 5. Looking for and flagging exception conditions
Now we put our exceptions, if any, in the line of output as shown in Figure 6.

```
HZSLFMSG_REQUEST = "CHECKMSG"
HZSLFMSG_MESSAGENUMBER = 258
HZSLFMSG_INSERT.0 = 9

HZSLFMSG_INSERT.1 = exceptionFlag
HZSLFMSG_INSERT.2 = RACF.R_PROFILE
HZSLFMSG_INSERT.3 = YesNo(RACF.R_GENERIC)
HZSLFMSG_INSERT.4 = RACF.U_PROFILE
HZSLFMSG_INSERT.5 = startedGroupName
HZSLFMSG_INSERT.6 = YesNo(RACF.U_BASE_U_PROTECTD.1)
HZSLFMSG_INSERT.7 = YesNo(startedTrustedFlag)
HZSLFMSG_INSERT.8 = YesNo(startedPrivilegedFlag)
HZSLFMSG_INSERT.9 = YesNo(startedTraceFlag)
HZSLFMSG_RC = HZSLFMSG()

IF HZS_PQE_DEBUG = 1 THEN
  DO
    SAY "HZSLFMSG RC" HZSLFMSG_RC
    SAY "HZSLFMSG RSN" HZSLFMSG_RSN
    SAY "SYSTEMDIAG" HZSLFMSG_SYSTEMDIAG
    IF HZSLFMSG_RC = 8 THEN
      DO
        SAY "USER RSN" HZSLFMSG_UserRsn
        SAY "USER RESULT" HZSLFMSG_AbendResult
      END
  END
END
```

Figure 6. Issuing lines of information in output message

After we process all of the STARTED class profiles, we issue our summary message, indicating whether the check found an exception. And finally, we end the connection to IBM Health Checker for z/OS (Figure 7):

```
/*********************
/* End of Check Function*/
/*********************

if checkException = "NO" then do
  HZSLFMSG_REQUEST = "CHECKMSG"
  HZSLFMSG_INSERT.0 = 0
  HZSLFMSG_MESSAGENUMBER = 259
  HZSLFMSG_RC = HZSLFMSG()
  IF HZS_PQE_DEBUG = 1 THEN
    DO
      SAY "HZSLFMSG RC" HZSLFMSG_RC
      SAY "HZSLFMSG RSN" HZSLFMSG_RSN
      SAY "SYSTEMDIAG" HZSLFMSG_SYSTEMDIAG
      IF HZSLFMSG_RC = 8 THEN
        DO
          SAY "USER RSN" HZSLFMSG_UserRsn
          SAY "USER RESULT" HZSLFMSG_AbendResult
        END
    END
  END
else do
  HZSLFMSG_REQUEST = "CHECKMSG"
  HZSLFMSG_INSERT.0 = 0
  HZSLFMSG_MESSAGENUMBER = 260
  HZSLFMSG_RC = HZSLFMSG()
  IF HZS_PQE_DEBUG = 1 THEN
    DO
      SAY "HZSLFMSG RC" HZSLFMSG_RC
      SAY "HZSLFMSG RSN" HZSLFMSG_RSN
      SAY "SYSTEMDIAG" HZSLFMSG_SYSTEMDIAG
      IF HZSLFMSG_RC = 8 THEN
        DO
          SAY "USER RSN" HZSLFMSG_UserRsn
          SAY "USER RESULT" HZSLFMSG_AbendResult
        END
    END
  END
HZSLSTOP_RC = HZSLSTOP() /* report check completion */
```

Figure 7. Issuing a summary message for the check and ending the connection to IBM Health Checker for z/OS

Finally, you must place your check source in an installation-defined library. And that's all there is to the check!

Defining your check to IBM Health Checker for z/OS

To define your check to IBM Health Checker for z/OS, define it in an entry in the HZSPRMxx parmlib member. Be sure to set the ENTRYCODE to the same value that you used in your REXX exec check (1, in our example) and set the REXXHLQ to a valid high level qualifier on your system. (Figure 8)

```
ADDREP CHECK(IBMSAMPLE,RACF_STARTED_CLASS)
EXEC(HCSTART)
REXXHLQ(MARKN)
REXXTSO(NO)
REXXIN(NO)
MSGTBL(HCSTCMSG)
ENTRYCODE(1)
US3(NO)
VERBOSE(YES)
SEVERITY(LOW)
INTERVAL(24:00)
DATE(20110703')
REASON('RACF Started Class Sample Check')
```

Figure 8. Defining a check to IBM Health Checker for z/OS

Note that we're not showing the details of message definitions for a check in this article. For details about defining checks and check messages, see the IBM Health Checker for z/OS User's Guide.

Activating your check

Now, we're ready to activate the check! Assuming that we've placed the health checker registration statements in the HZSPRMxx parmlib member, the command to execute the PARMLIB statement is:

```
F HZSPROC,ADD,PARMLIB=nn
```

In summary

As you can see, IRRXUTIL, System REXX, and the IBM Health Checker for z/OS provide a powerful combination ready to help you create your own Health Checks to perform your own checks on your RACF database!
Archiving data on offsite tape drives is a common business practice for disaster recovery. Entire databases – and even systems – are written to such transportable media as tape devices. These tapes are sent long distances to secure facilities. Tapes that contain sensitive personal information have been lost during this process.

It seems as though the number of laws and regulations intended to safeguard electronic-based data from the threat of unauthorized access grows every day. Many customers seem interested in data encryption as one means of compliance.

Central to any encryption-based strategy is the management of encryption key material. Although the data storage devices that are involved perform the cryptographic operations, the encryption keys are generated and securely stored away from those devices.

The IBM Security Key Lifecycle Manager for z/OS (ISKLM) provides this centralized key management for device-based encryption on 3592 and Linear Tape-Open (LTO) tape drives and the DS8000® Storage Controller. This article shows you how to implement ISKLM on z/OS.

Installing ISKLM

Devices contact the key manager on a TCP/IP port to request keys. The key manager services that request by using a key from a keystore file or RACF keyring. The Integrated Cryptographic Service Facility (ICSF) can be used to store keys behind the CryptoExpress3 features. Keys are added to the keystore using commands that are native to the keystore type. Administrators communicate with ISKLM using z/OS operator commands.

ISKLM is a Java program that runs as a started task. The Java virtual machine (JVM) starts a special Java class called ibm.jzosekm.ISKLMConsoleWrapper. This class manages communication between the z/OS operator console and the ISKLM application. Commands that are sent to the ISKLM and the responses to these commands are passed through this Java class.

Several data sets and files make up the ISKLM implementation. Some of these files are configurable and are managed by the system administrator:

- A started procedure in a library that is in the JES2 or JES3 concatenation for started tasks
- A data set that contains the IBM z/OS UNIX System Services environment variables
- A configuration file in a UNIX System Services file system

Some files are created by the application during initial startup:

- An audit log file
- A debugging file
- A device metadata file
- A keygroups metadata file

Stay tuned for this important broadcast message

Implementing ISKLM on z/OS

BY WILLIAM C. JOHNSTON
Customizing ISKLM

You need to move several files in order for ISKLM to operate correctly on z/OS. The following steps show you how to copy the JVM load module to the z/OS link list concatenation, place security files in the correct Java directory, and make the ISKLM for z/OS Java archive (JAR) file accessible to the JVM.

1. Copy the JVM load module (JVMLDMnn) into the link list concatenation:

   ```bash
   cp -X $JAVA_HOME/mvstools/JVMLDM60 "//SYS1.SIEALNKE(JVMLDM60)"
   ```

   This example uses SYS1.SIEALNKE, but any partitioned data set extended (PDSE) can be used. In this way, the JVM can be run as a started procedure.

2. Copy the unrestricted policy files into the Java security directory:

   ```bash
   cp $JAVA_HOME/demo/jce/policy-files/unrestricted/* $JAVA_HOME/lib/security
   ```

   ISKLM requires these files so that it can serve Advanced Encryption Standard (AES) keys.

3. Copy the IBMSKLM.jar file into the Java library extensions directory:

   ```bash
   cp /usr/lpp/ISKLM/IBMSKLM.jar $JAVA_HOME/lib/ext
   ```

   The $JAVA_HOME variable points to the Java installation directory.

Customizing the started task

ISKLM comes with sample job control language (JCL) code for the started task. This JCL code must be tailored for the installation. The LIBRARY variable points to the data set where the JVMLDMnn load module was stored.

```bash
// LIBRARY='SYS1.SIEALNKE', < STEPLIB FOR JVMLDM module
```

The VERSION variable denotes the version of Java that is being used. The &VERSION suffix on the JVMLDM string (JVMLDM&VERSION) is used to start the JVM on the EXEC statement of the started task control (STC) procedure.

```bash
// VERSION='60', < JVMLDM version: 14, 50, 56
```

The STDENV DD statement must point to the data set that contains the environment variables:

```bash
//STDENV DD DSN=USER. PLX4.PROCLIB(ISKLMENV), DISP=SHR
```
Regardless of which keystore type is implemented, it is predefined and used by ISKLM to serve keys. ISKLM does not create keys to be stored in the keystore.

ISKLM uses several keystores. These keystores must all be the same type. They can either be the same physical keystore or they can use different instances of the type. The ISKLM keystores are:

• config.keystore, which holds the keys that are used for encryption
• Admin.ssl.keystore, which holds client certificates for secure communication between ISKLM instances for synchronization operations
• Admin.ssl.truststore, which holds server certificates for secure communication between ISKLM instances for synchronization operations
• TransportListener.ssl.keystore, which holds client certificates for devices that use the Secure Sockets Layer (SSL) to contact the key manager
• TransportListener.ssl.truststore, which holds server certificates for devices that use the SSL to contact the key manager.

For best results, ensure that config.keystore is separate from the other keystores. However, the other keystores can all be the same keystore.

For each keystore, you must set three properties that define the keystore's name, type, and password.

For example, to define config.keystore as a Java file:

```text
config.keystore.name     = full_path_of_the_file
config.keystore.type     = JCEKS
config.keystore.password = file_password_value
```

To define config.keystore as a RACF keyring:

```text
config.keystore.name     = safkeyring://SKLMSRV/ISKLMRing
config.keystore.type     = JCERACFKS
config.keystore.password = password
```

To define config.keystore as a Java file using ICSF:

```text
config.keystore.name     = full_path_of_the_file
config.keystore.type     = JCECCAKS
config.keystore.password = file_password_value
```

To define config.keystore as a RACF keyring using ICSF:

```text
config.keystore.name     = safkeyring://SKLMSRV/ISKLMRing
config.keystore.type     = JCECCARACFKS
config.keystore.password = password
```

The same constructs hold true for the SSL keystores and truststores. The value of the password variable for RACF-based keystores is always password. RACF manages security on the keyrings using classes and profiles, not passwords, but Java requires the password property to be defined. After ISKLM is initialized, the password property is renamed and changed to a hashed value. ISKLM is designed not to process passwords in the clear after that.

Keys are added to the ISKLM keystore using the external commands based on keystore type:

- **JCEKS** uses the Java command keytool:
  ```text
  keytool –v –genkeypair –alias keyLabel ...
  ```

- **JCERACFKS** uses RACDCERT:
  ```text
  RACDCERT GENCERT ID(ISKLM_user_ID)
  WITHLABEL('keyLabel') ...
  ```

- **JCECCAKS** uses hwkeytool:
  ```text
  hwkeytool –v –gencert –alias keyLabel ...
  ```

- **JCECCARACFKS** uses RACDCERT with the PCICC(*) option:
  ```text
  RACDCERT GENCERT ID(ISKLM user_ID)
  WITHLABEL('keyLabel')PCICC(*)...
  ```

See the appropriate command manual for more information.

### Additional authorizations

The ISKLM STC user ID must have the proper authority to read the keyrings. Ensure that the user ID owns the keyrings that the key manager uses.

You can configure ISKLM to write audit data as System Management Facility (SMF) records by using the following property:

```text
Audit.handler.class = com.ibm.ltklm.audit.smf.
SMFSecurityEventHandler
```

ISKLM requests SMF Type 83, subtype 6 records. Ensure that the system is capturing these types of SMF records.

Writing SMF records requires UPDATE authority to the R_auditx callable service:

```text
PERMIT IRR.RAUDITX CLASS(FACILITY) ID(user_ID)
ACCESS(READ)
```

The security administrator must have UPDATE authority to the ISKLM keyrings:

```text
PERMIT ISKLM_STC_user_ID.ring_name.UPD ID(security_admin_user_ID) CL(RDATALIB) ACCESS(READ)
```

### Disaster recovery considerations

Ensure that the file systems that contain ISKLM code and configuration files and the keystore (file or RACF database) are not stored on media encrypted with keys stored on encrypted media.

See the ISKLM product documentation for more information.
IBM Common Cryptographic Architecture (CCA) has been extended to include support for controlled and typed exchange of Data Encryption Standard (DES) / Triple DES (TDES) secure keys. These keys are designed to use the methods described in American National Standards Institute (ANSI) Technical Report 31 (TR-31). CCA defines a set of cryptographic methods and data structures, including key token formats, that are common across IBM products. Many experts in the industry believe TR-31 defines a method for secure key wrapping that complies with American National Standard (ANS) X9.24 Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques.

Key exchange: an overview
CCA has an existing, well-established system of key usage control that is intended to securely bind key usage information to the key material, including during key transfer, by way of a control vector (CV). Before TR-31, however, there was no generally agreed-upon method for secure, interoperable symmetric key exchange.

A common method of transferring keys between different cryptographic systems was to strip the key of its attributes and send just the untyped key and then attach the attributes again at the receiving end. Not only is this method error prone, but it might also be a security risk: an insider might obtain the untyped key and attach a different set of attributes to it that could then be used to attack the system. TR-31 can help prevent this security exposure by allowing differing cryptographic systems to securely exchange typed key material in an interoperable manner.

As an example of key exchange using TR-31, this article focuses on the keys that are used when customers access a bank’s ATM using payment cards and personal identification numbers (PINs). To simply our example, let’s just say that the ATM encrypts the customer’s PIN using a PIN encryption key, and the bank’s computer must have that same key in order to decrypt and verify the PIN.

So how does this work? We’re glad you asked!

How is TR-31 implemented?
The TR-31 function is implemented in the Crypto Express Hardware Security Module (HSM) and accessed on z/OS through a set of services provided by the Integrated Cryptographic Services Facility (ICSF). Services are provided that allow you to import (CSNBT31I) and export (CSNBT31X) a key to or from TR-31 format. You can also parse (CSNBT31P), read (CSNBT31R), and build (CSNBT31O) optional data to bind to a TR-31 key.

Bootstrapping the process
Before we can begin using the TR-31 services to transfer keys, both parties must share a symmetric key that is suitable for protecting a TR-31 key block (also known as a key block protection key). Such a key must be used, at a minimum, to protect the first TR-31 key block that’s transmitted. This first TR-31 key block could contain the key block protection key that is to be used to protect future TR-31 key blocks. If such a key does not yet exist between both parties, they need to use existing mechanisms to accomplish its transfer.

Transferring keys
Now that we’ve established a common key block protection key between both parties, let’s see how we can put these services together to transfer keys using TR-31. Going back to our example, we have an ATM and a bank that need to share a common key for encrypting and decrypting PINs. Let’s assume that the bank is using ICSF to transfer the key to the ATM and that the ATM uses a non-CCA cryptographic provider.

At the bank, we use the TR-31 Export service to take the PIN-encrypting key in CCA format and export it to the ATM in TR-31 format. This service accepts a number of parameters, but here we will only focus on those parameters that control the usage of the key. By including the keyword parameters PINENC and ENG-ONLY, we define the key as one that should be used only as a PIN-encrypting key. The resulting TR-31 key block is then a bundling of the key and all of its attributes, which are
cryptographically bound together with a message authentication code (MAC) and protected under the key block protection key for transfer to our ATM.

Let’s now take a look at the other side of the exchange: receiving a TR-31 key block. In addition to processing PINs from its own customers, the bank often needs to forward PINs belonging to other banks for processing. A separate PIN encrypting/decrypting key is shared between the bank and the external network for this purpose.

CCA has an existing, well-established system of key usage control . . .

Let’s assume that this other PIN encrypting/decrypting key pair was created outside of the bank and the bank receives the PIN-encrypting key in TR-31 format. The bank now uses the TR-31 Import service to import the incoming TR-31 key block to a CCA key that can be used by ICSF. When importing the key, you would include the keyword parameters PINENC and ENC-ONLY to import the key as a PIN-encrypting key only.

HSM’s built-in protections

Because the TR-31 Import and Export services allow the user to specify keywords that indicate the intended use of the key, it might seem as if an attacker could change the type of a key to something entirely inconsistent with its intended use. Fortunately, the HSM design explicitly prohibits this. For example, the bank is prohibited from importing the PIN-encrypting key as a data-encrypting key. What the keywords do provide is the ability for the user to perform the conversion between CCA key types and TR-31 key types. The reason for this is that there is not always a one-to-one mapping between the key types defined by TR-31 and those defined by CCA. For instance, CCA defines one type of PIN-verifying key (NO-SPEC) that can be used in any PIN algorithm, but TR-31 does not. This is one reason why, when exchanging keys between CCA systems, it is best to use the CV within the CCA key token to convey the key’s attributes, either by using existing ICSF export/import services, which transfer keys in CCA format, or by using the ATTR-CV keyword with the TR-31 Export service, to embed the CV within the TR-31 key block.

The HSM implements an access control system through access control points (ACPs), which must be enabled to allow specific behaviors. For TR-31, separate ACPs are defined for most CCA-to-TR-31 or TR-31-to-CCA key export or import translations, so as a user, you can prohibit many usages specifically and independently. On z/OS systems, the ACP settings are universal and apply to all users of the system. Using the ACPs, you can

Figure 1: Exporting a CCA key to TR-31 format
disable export/import translations that should never occur on the system. On 
z/OS systems, use the Trusted Key Entry (TKE) workstation to modify the 
ACPs.

The CV of a CCA DES/TDES 
key has an added bit for TR-31 support 
that is used to indicate whether the key 
can be exported to a TR-31 key block. 
This bit works in conjunction with the 
general export bit that already exists in 
the CV. In order to export a CCA key 
to TR-31 format, general exporting and 
TR-31 exporting must be allowed.

**SAF protections**
We’ve seen how the ACPs and the 
TR-31 export bit can be used prevent 
certain translations and the exporting 
of some keys altogether. However, 
the ACPs and TR-31 export bit by 
themselves do not allow us to make 
decisions about who should be able to 
perform TR-31 operations and who 
should not. This is where the Secure 
Access Facility (SAF) of z/OS comes in. 
The TR-31 Import and Export services 
are protected by a unique SAF profile 
in the CSFSERV class: CSFT31X for 
export and CSFT31I for import. Using 
these profiles, you can choose to grant 
access only to trusted applications or 
individuals.

In addition to the TR-31 services, SAF 
also controls authority to use the keys 
managed by ICSF, which are included 
in the Cryptographic Key Data set 
(CKDS). To export a key in the CKDS 
to a TR-31 key block requires you as a 
user to have at least READ access to the 
CSFKEYS profile covering that key. 
This makes it all the more important 
for you to properly configure access to 
the CSFSERV profiles.

**Conclusion**
To aid in understanding, this article 
limited the focus to a basic key 
exchange flow involving a specific set 
of keys in a specific circumstance. The 
scope of TR-31, however, is not limited 
in this fashion. The TR-31 support 
allows the exchange of most key types, 
including, but not limited, to key-
encrypting keys, data-encrypting keys, 
mandatory access control (MAC) keys, 
and EMV/Chip Issuer master keys, 
which have a variety of applications.

Exploiting TR-31 on z/OS requires 
z/OS V1R11 (or later), an IBM 
zEnterprise server (z114, z1196, or 
later) with a Crypto Express3 (or 
later) Coprocessor using September 
2011 (or later) Licensed Internal Code 
(LIC), the level of ICSF provided in 
the Cryptographic Support for z/OS 
V1R11, V1R12, V1R13 (or later) web 
deliverable, and of course, a partner 
who also supports TR-31.

**References**
For more information about the TR-
31 support, see the z/OS Cryptographic 
Services ICSF: Application Programmer’s 

For a list of ICSF publications, see the 
following website: 

**Acknowledgements**
The authors would like to thank 
Todd Arnold and Carsten Dahl Frehr 
for their many contributions to this 
article.
Easy does it with DB tooling and training

BY JEFF M. SULLIVAN AND JAVIER CHAVEZ

Being part of the database administration (DBA) staff carries many responsibilities. The role requires having that special ability to perform many unrelated tasks such as installation and maintenance of software, performance tuning applications, and being that occasional sounding board or a shoulder to cry on for applications programmers and database users.

A DBA provides the glue between application programming teams and system programming. This position occupies a unique role that doesn’t fall into the area of the operating system administrators, z/OS system programmers, DASD administrators, or network or security administrators. As a result the database administrator position is often the first line of contact between the IT organization and the rest of the company.

Economic realities
We live in a world dictated by economic terms. Gone are the days when you had the time to perform root cause analysis for many IT problems. Project plans are no longer measured in years, but what can be finished this quarter before the budget runs out. And training budgets are so lean that we are told to look up the information on the Internet.

Given all of these factors, the DBA still must abide by the fundamental aspects of the job guided by the following principal goals of the database support and administration team for the company’s data assets:

• Protection
• Availability
• Fast delivery
• Analysis of data asset problems as required

Tools and training for the DBA
Let’s look at having adequate tools that can give you more time to balance the needs of keeping up with both corporate agendas and the major goals that guide the database team. In additionally, let’s look at what you can do when faced with a project where the work requires a skill set not available in the IT organization.

To discuss what is meant by adequate tooling, we need to understand the work that we do. One principle aspect is staying proactively ahead of database problems. The idea of working smarter is not only a catch phrase—it’s a necessity.

Commodity tasks
We start working smarter by streamlining all those commodity-based tasks—the things you do so often that they require no special analysis to perform and can be time-based or happen periodically. An example of a commodity task is periodically checking if any of the database management system (DBMS) objects are in danger of running out-of-space, which, left ignored, can cause an outage. Another example is checking performance or, more appropriately, being asked to “performance tune” some piece of sequential query language (SQL). Yet another example is documenting the current physical database environment and how it interacts between applications.

Obviously, classifying work is a lot of darn work; but it is these repeatable-and-required time-sapping things that a DBA performs, either daily or weekly, that should be reduced in the time it takes to perform. This is where tooling plays a major role in reducing commodity-type tasks and, specifically, will help you in minimizing the drudgery of these types of repeatable, required work items. For example:

• Data diagrams
• Object manipulation
• SQL tuning.
Did Someone say Tooling?

How many times have you been approached by an application programmer asking you for a system diagram? Do you remember back in the day breaking out your plastic flowchart template and attempting to draw the DB2 or IMS structures? Yeah, that was fun being an artist for a few days!

Now, I am not talking about a full-fledged data architect role. Architecture people live and breathe metadata, and can quote rules on exactly how to transform data into “4th normal form”. What I am talking about is the informal “what-are-we-running-and-what-does-it-look-like” conversation. This is where a good tool comes into play and one that is free to use with the operable word being “free”.

IMS Explorer and IBM Data Studio

IBM IMS Enterprise Suite Explorer is one such product that is in my arsenal of tools along with IBM Data Studio. With these tools, I can create application and system diagrams to quickly fulfill the need of any programmer looking for instant application knowledge. With IMS Explorer, part of the IBM IMS Enterprise Suite, not only can you create the database definition (DBD) diagram, but you can also query the database with the SQL plug-in. IBM Data Studio does these same visualization features with DB2 for z/OS and DB2 for Linux, UNIX, and Windows.

IBM Infosphere Data Architect

Now, there are more sophisticated tools for answering the informal question we asked earlier — “what are we running and what does it look like”—but they come with a price. Of these, I really like IBM Infosphere Data Architect because it can do so much more than create pretty diagrams. This is a very good tool to perform deep dive physical and logical modeling, along with artifact and metadata collection. It also interfaces with many of the available eclipse-based tools.

Change happens

While we are on that subject, both IMS Explorer and IBM Data Studio allow DBMS object manipulation. When have you NOT been interrupted for some emergency such as adding a column to a table in test when you were busy with an outage problem in production? The production problem takes precedence, but the test issue is delaying a project deadline important to management. These tools can get that column added quick without taking you away from an outage.

For IMS, we never really had anything that was cool and exciting like the many DB2 tools until now. You can finally do away with that PDS containing all of your DFSDDLT0 control statements because you can use SQL to manipulate the IMS data. As for DB2, we had tools like IBM DB2 Administration Tool but with IBM Data Studio, you get many of the same things but in a graphical interface. Did someone say “multitasking on steroids”?

The ease-of-use in adding columns can be quickly accomplished as shown in Figure 1.

What is best, though, is making the changes with these tools without necessarily interrupting any work being done with a TN3270 session. Most shops usually do not allow these types of tools to run against the production mainframe. Rightfully so, after living through a 2:00 AM call where some “cowboy” put in an untested change!

SQL tuning tools

The last tool that is on my tools list is IBM Optim™ Query Tuner for DB2 for z/OS and its big brother IBM Optim Query Workload Tuner for DB2 for z/OS. These are not free products, but you will absolutely love these tools.

The first reason to consider SQL tuning tools is if you are doing data warehousing/business intelligence and you have been given the task to tune this SQL. Have you ever seen data warehouse/business intelligence SQL? This is some of the most complicated and lengthy SQL you will ever see. If you tune SQL “old school” using a PLAN TABLE, you will spend days tuning this stuff.

Ease of use is what’s really nice with using Optim Query Tuner for DB2 for z/OS With a click of a button along with a mainframe connection, the tool advisors provide advice on a tuning strategy and build the DML required to tune the queries.
Obviously, the second reason is that there is so much SQL and so much of it can be problematic. A tool that can quickly tune this SQL has the potential of saving the company machine CPU cycles but, more to the point, it saves you time.

Internet training for the DBA
We’ve seen how tooling can reduce the time it takes to do the easy-and-repeatable tasks. What about those situations where you have been asked to install and support a new database environment — and you’ve never done it before? The boss tells you to just go look on the internet and learn how to do it.

The internet is exploding with how-to videos that can explain everything from how DB2 data sharing works to the best ways to give your dog a bath.

There is just no way to quickly learn the things you need to know or, more importantly, where the landmines are without taking a structured learning class. Without the knowledge, even the most experienced person cannot know the right questions to ask, which things are important, and what you can defer. The lost productivity is much more costly than the cost of the class tuition.

Bottom line
Tools greatly simplify the responsibilities of the DBA group and can allow you to spend more time furthering the corporate agenda and making the business profitable. When you get down to it, that’s the primary objective for all of us — the bottom line.

For more information on the tools and training mentioned in this article, see the following web sites:

- **DB2 and IMS Tools**
  ibm.com/software/data/db2imstools
- **IBM Data Studio**
  ibm.com/software/data/optim/data-studio/
- **IBM InfoSphere Data Architect**
  ibm.com/software/data/optim/data-architect/
- **Optim Query Tuner for DB2 for z/OS**
  ibm.com/software/data/optim/query-tuner-z/
- **IMS Enterprise Suite**
  ibm.com/software/data/ims/soa-integration-suite/enterprise-suite/
- **IBM Training worldwide**
  ibm.com/training

Driver 93 on IBM zEnterprise 196 (z196) or IBM zEnterprise 114 (z114), and subsequent rollbacks, introduces the Problem Management Viewable (PMV) record. A PMV is a problem record that you can open using the hardware management console (HMC), or support element (SE), both equipped with phone home capabilities. The PMV goes directly to IBM Level 2 hardware support. The good news is that while a traditional PMH is not visible to you, if you are using the HMC or SE interface, you can update and view the contents of a PMV.

Advantages
A PMV provides you with the ability to exchange information directly with IBM hardware support. You can take advantage of the new PMV option to describe to IBM hardware support precisely what you are seeing, ask a question, or express your concern about an issue. IBM hardware support will optimize on the PMV interface to maximize first failure data capture success and expedite problem resolution. What this really means to you is problem data like logs and screen shots can get to IBM support right away and potentially minimize the likelihood of losing or truncating meaningful data. Getting the right documentation to the right people, as fast as possible, can lead to faster resolution of your problem.

Setup is a snap
When you open the PMV, you first supply some initial information including:

- Serial number of the machine,
- Driver microcode change level (MCL)
- First error data collect (FEDC)

To expedite problem resolution it’s critical to report the problem from the correct machine.

Creating the PMV is easy
Log on to the HMC or SE, under “defined CPC” and select the central processor complex (CPC) for which you want to create a PMV.
Select the report a problem icon under the “service task.” When reporting the problem select “Type V Viewable PMH (PMV)” to ensure the ability to view the record on the HMC. In the text box on this screen, enter a problem description and optionally provide an email address. Providing your email creates another option that allows you to exchange information with IBM Level 2 hardware support (Figure 1).

Select the “Request Service” button to continue with the PMV creation. The next panel will prompt you for your name, contact phone number, and electronic or voice transmission preference. Providing the name of the correct individual to contact and the best contact number enables IBM Level 2 hardware support to reach you quickly and advances the problem resolution swiftly.

After filling in the desired contact means and contact information, select the Request Service button a second time to create the PMV (Figure 2). You receive the problem management hardware number from a hardware message and can reference the number whenever you want to view that PMV (Figure 3).

You can view the PMV under the service task view PMV option. You can also add a comment directly into the record.

**Next?**

After the PMV is opened by the IBM Level 2 hardware support center, they contact you if appropriate and route the call to the next level of support with the correct severity level assigned. IBM Level 2 hardware support receives the PMV and continues to assist by reviewing the PMV, then calling you back directly (using the contact method you provided in Step 3). Whenever the PMV is updated, you are notified. You can then view the PMV on the HMC or SE and take the next action.

A PMV works best for issues related to the user experience at the time the problem is occurring. For example, if a user is on the operating system console and the console generates an error message or gives an unexpected response a PMV is a great way for users to explain directly to IBM hardware support exactly what they are experiencing. In this situation, it’s ideal to take screen captures to share with IBM hardware support.

**Finding a proper home**

On occasion, an issue is reported to the IBM software support center and is ultimately resolved by IBM hardware support. When this happens, IBM software support transfers the software PMR to the IBM hardware support where it becomes a PMV, and the record can then be viewed by you on the HMC.

**At home with hardware**

The components most likely to transfer from IBM software support to IBM hardware support are those that interact most directly with the hardware or firmware. For instance, cross-system extended services (XES) is a component of z/OS that provides services applications and subsystems can utilize to communicate with the coupling facility in a
Parallel Sysplex. Performance problems associated with a coupling facility are equally likely to be reported to IBM software support as they are IBM hardware support. The issue can be reported to software and ultimately determined to be resolved by the coupling facility control code firmware. In this type of situation, it’s appropriate to transfer the PMR from software support to hardware support and enable you to continue to view the record as a PMV.

**At home with software**

Similarly, it’s possible that a problem is reported to the IBM hardware support center and is resolved by IBM software support. The PMV is transferred from IBM hardware support to IBM software support to become a PMR. The PMR can be accessed by the Service Request (SR) tool. For instance, a server time protocol (STP) for a PMV might be opened to IBM hardware support who ultimately determines that the resolution lies with the z/OS STP component. In this situation, it’s appropriate to transfer the PMV from hardware support to software support.

**PMV in action**

The PMV is a good way to interface with IBM hardware support and is not intended to replace the traditional PMH. There are always situations where a PMH is more appropriate than a PMV. Let’s consider some examples.

**Scenario one**

Joe, a system programmer for the XYZ Company, is using the user ID wizard and receives a failure message from the hardware that reads licensed internal code (LIC) and has detected a problem as in this example:

**Problem Description:** The support element licensed internal code has detected a problem with Coupling Facility that may have caused sysplex impact.

When a licensed internal code error occurs, it automatically generates a “phone home.” There’s no need for a PMV in this situation because the “phone home” call has already occurred. IBM Level 2 hardware support receives the PMH and calls Joe. Depending on the details of the situation, the system services representative (SSR) might be engaged to further assist with the situation to aid Joe and IBM Level 2 hardware support.

**Problem Description: The support element licensed internal code has detected a problem with Coupling Facility that may have caused sysplex impact.**

**Scenario two**

Another example, seen recently, involved the Coupling Facility Control Code (CFCC), which provides the capability to non-disruptively dump a coupling facility image. When a non-disruptive CFCC dump is generated, the CPC automatically phones home. As in the first scenario, because the CPC itself already phoned home, IBM hardware support is engaged automatically, so there is no reason to open a PMV.

**Now you have it!**

Using the new PMV gives you greater access to the highly skilled IBM Level 2 support representatives throughout the lab. With IBM hardware support leveraging PMV to maximize first failure data capture, IBM support can expedite problem resolution and provide the best possible solution for your systems.
Want to simplify your TCP/IP profile? This article will show you how to do just that, using the IPv4 INTERFACE statement for Open Systems Adapter-Express (OSA-Express) queued direct input-output (QDIO) definitions. The IPv4 INTERFACE statement provides many benefits. Making changes to your configuration just got a whole lot easier.

When you use the INTERFACE statement, you can also take advantage of the multiple virtual LAN (VLAN) support for OSA-Express QDIO. This function allows you to configure multiple interfaces from the same TCP/IP stack onto the same OSA port for IPv4 and IPv6.

A little background: traditional IPv4 definition statements

Before z/OS V1R10, the only way you could configure IPv4 OSA-Express QDIO definitions was by using the DEVICE, LINK, and HOME statements. Here’s an example:

```
DEVICE OSAQ1 MPCIPA
LINK OSAQLINK1 IPAQENET OSAQ1 VMAC
HOME 9.67.1.1 OSAQLINK1
```

In this definition, the device name (OSAQ1) must match the port name from the corresponding transport resource list element (TRLE) definition.

A better way: the IPv4 INTERFACE statement

The IPv4 INTERFACE statement combines the functions of the DEVICE, LINK, and HOME statements into a single statement. This example shows an equivalent definition using one statement:

```
INTERFACE OSAQINT1 DEFINE IPAQENET PORTNAME OSAQ1 VMAC
   IPADDR 9.67.1.1/24
```

Here, the PORTNAME parameter (OSAQ1) ties this statement to the associated TRLE definition. Start the OSA using the interface name (OSAQINT1). The stack uses this name when it issues messages pertaining to this interface.

Now let’s look at some other benefits of the IPv4 INTERFACE statement.

Changing your configuration

When making changes to your configuration with the VARY TCPIP,OBEYFILE command, the task becomes much simpler when you use the INTERFACE statement.

To define a new OSA using DEVICE and LINK, you need to specify:

- A new DEVICE statement
- A new LINK statement
- A complete replacement for the HOME statement that adds the new IP address to the entire set of home IP addresses

To delete a DEVICE and LINK definition for an OSA, you need to specify:

- A DELETE DEVICE statement
- A DELETE LINK statement
- A complete replacement for the HOME statement that lists all of the home IP addresses, except the one you are removing
In these procedures, updating the HOME statement is awkward and error prone.

However, if you use the INTERFACE statement, these tasks are much easier. You can add a new interface by defining a new INTERFACE statement in the OBEYFILE data set:

```
INTERFACE OSAQINT2 DEFINE IPAQENET PORTNAME OSAQ2 VMAC IPADDR 9.67.2.1/24
```

Similarly, you can delete an interface using a single statement:

```
INTERFACE OSAQINT1 DELETE
```

Enabling source VIPA

If you enable the source virtual IP address (VIPA) function using the SOURCEVIPA parameter on the IPCONFIG statement, the TCP/IP stack sends outbound packets using a specified static VIPA as the source IP address.

When source VIPA is in effect and you define an OSA using DEVICE and LINK, the source VIPA used by the stack for that OSA depends upon the order of the IP addresses in the HOME statement in the profile.

With the IPv4 INTERFACE statement, source VIPA selection is much more straightforward. To designate a source VIPA for an interface, use the SOURCEVIPAINTERFACE parameter on the INTERFACE statement to specify the link name of an IPv4 static VIPA:

```
INTERFACE OSAQINT2 DEFINE IPAQENET PORTNAME OSAQ2 VMAC IPADDR 9.67.2.1/24 SOURCEVIPAINTERFACE VIPA1
```

Other INTERFACE benefits

OSA-Express QDIO offloads address resolution protocol (ARP) processing from the TCP/IP stack. With a DEVICE and LINK definition, the stack instructs OSA to perform VIPA ARP processing for all VIPAs, which can result in a number of unnecessary ARPs. With the INTERFACE statement, the VIPA ARP processing is limited to VIPAs in the same subnet as the OSA based on the subnet mask specification on the IPADDR parameter (/24 in the examples in Figure 1). Using the INTERFACE statement eliminates any unnecessary ARPs.

With the INTERFACE statement, you can specify a maximum transmission unit (MTU) for the interface using the MTU parameter.

Accessing multiple VLANs

A VLAN is a configured logical grouping of nodes. You can subdivide a physical LAN into separate VLANs. This technology has become an important aspect of network configuration planning.

The multiple VLAN function allows you to access multiple VLANs from one stack through the same OSA-Express port. To do this, configure separate INTERFACE statements for each VLAN, point these to the same OSA port using the PORTNAME parameter, and put each VLAN into a separate subnet.

Figure 1 shows a stack accessing two VLANs over OSA port OSAQ1. Here are some matching sample definitions:

```
INTERFACE OSAQVL11 DEFINE IPAQENET PORTNAME OSAQ1 VMAC VLANID 11 IPADDR 9.67.11.1/24
INTERFACE OSAQVL12 DEFINE IPAQENET PORTNAME OSAQ1 VMAC VLANID 12 IPADDR 9.67.12.1/24
```

A TCP/IP stack can have up to 32 IPv4 interfaces and up to 32 IPv6 interfaces on the same OSA port.

Even more benefits

The multiple VLAN support provides much greater network configuration flexibility. Before z/OS V1R10, the only way to access multiple VLANs from one stack was to use separate OSA ports. The only way to access multiple VLANs from the same OSA port was to share that port across multiple stacks.

Using the multiple VLAN function, you can consolidate multiple OSAs on one stack across different VLANs onto a single OSA port. Similarly, you can consolidate multiple application servers across multiple stacks into a single z/OS image where the traffic related to these servers is on unique VLANs.

```
INTERFACE OSAQVL11 DEFINE IPAQENET PORTNAME OSAQ1 VMAC VLANID 11 IPADDR 9.67.11.1/24
INTERFACE OSAQVL12 DEFINE IPAQENET PORTNAME OSAQ1 VMAC VLANID 12 IPADDR 9.67.12.1/24
```

Figure 1. Multiple VLANs on the same OSA port
Follow the rules

To configure multiple VLANs to an OSA port, you need to adhere to these rules:

• Use the INTERFACE statement for each definition.
• Within each IP version, configure a unique VLAN identifier for each statement using the VLANID parameter.
• Use a virtual MAC address by specifying the VMAC parameter with the default ROUTEALL attribute. You can configure a virtual MAC address or request that OSA generate one.
• Configure a unique subnet for each IPv4 interface for this OSA-Express port using the subnet mask specification on the IPADDR parameter of the INTERFACE statement.

If one of your definitions does not conform to these rules, the stack issues message EZD0044I to identify which rule you did not follow:

```
EZD0044I INTERFACE interface_name NOT ALLOWED - reason
```

Understanding the resource requirements

Here are some considerations when planning your network configuration.

DATAPATH devices

In order to use multiple VLANs for an OSA, you need to configure a separate interface to the OSA for each VLAN. Each of these interfaces requires a separate DATAPATH device in the TRLE definition. Therefore, you need to make sure the DATAPATH parameter on your TRLE definition specifies a sufficient number of devices for your configuration.

If you have an existing DEVICE and LINK definition and an IPv6 INTERFACE definition for the same OSA port, these definitions share a DATAPATH device. If you convert the DEVICE and LINK definition to the IPv4 INTERFACE statement, this combination uses an additional DATAPATH device.

Fixed storage

• Each DATAPATH device uses a certain amount of fixed storage for inbound data processing. For OSA-Express QDIO, you can globally configure the amount of fixed storage for each DATAPATH device using the QDIOSTG VTAM start option. You can override this value for a specific interface by using the READSTORAGE parameter of the INTERFACE statement.
• For information about the TRLE definition and the fixed storage required for each DATAPATH device, see the z/OS Communications Server: SNA Resource Definition Reference, SC31-8778.

Converting to the INTERFACE statement

The z/OS Communications Server: IP Configuration Guide, SC31-8775, contains instructions for converting your existing DEVICE, HOME, and LINK statements to the INTERFACE statement.

Profile conversion tips

For help with updating your TCP/IP profile to use IPv4 INTERFACE statements, try these steps:

1. Obtain a dump of the TCP/IP address space.
2. Use the Interactive Problem Control System (IPCS) against that dump.
3. Issue the IPCS subcommand TCPIPCS PROFILE(CONVERT).

The resulting output displays your profile configuration with the OSA-Express QDIO definitions converted from DEVICE, LINK, and HOME statements into INTERFACE statement format. The IPCS subcommand also converts your existing static route definitions from the GATEWAY statement into BEGINROUTES format. When you use the INTERFACE statement, you need to use the BEGINROUTES statement to configure static routes.

Using the IPCS can be a big help as you migrate to the IPv4 INTERFACE statement. For more information about the IPCS subcommand, see the z/OS Communications Server: IP Diagnosis Guide, GC31-8782.

Other references

• For the complete syntax of the IPv4 INTERFACE statement, see the z/OS Communications Server: IP Configuration Reference, SC31-8776.
• For more information about the multiple VLAN function, see the z/OS Communications Server: IP Configuration Guide, SC31-8775.
Endless opportunities: z/OS LDAP plug-ins

BY JON FURMINGER AND SAHEEM GRANADOS

LDAP plug-ins allow administrators to customize their LDAP server. Because LDAP is a widely used, mature technology, the opportunity to customize the server can allow administrators to develop new interactions between otherwise incompatible IT components. This capability can benefit an IT organization in many ways, such as enhancing IT security, making propriety data models accessible by using a standard protocol, and automating unrelated tasks.

Are LDAP plug-ins difficult to create? How do I configure them? Can I access or modify Directory information? What about system information? This article answers these questions and more.

What is a plug-in?
A plug-in is a user-defined extension to an LDAP server. It consists of a dynamic link library (DLL) created by an application writer or independent software vendor. IBM also has made available several LDAP server plug-ins. Plug-ins are typically written in C using a publicized interface called SLAPI. The SLAPI interface is supported by various LDAP server implementations on many platforms, such as IBM Tivoli Directory Server (which runs on platforms other than z/OS). With multiple LDAP platforms supporting the SLAPI interface, a plug-in writer could easily deploy their plug-ins on multiple platforms.

Ultimately, a plug-in is designed to enhance how an existing LDAP server performs the LDAP requests, which are: ADD, BIND, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, RENAME, SEARCH, and UNBIND.

When is a plug-in called?
LDAP servers typically define three distinct times when a plug-in can be invoked during its processing of LDAP requests.

- Pre-operation,
- Post-operation,
- In place of the operation, which is referred to by z/OS LDAP as a client-operation.

A pre-operation plug-in is called before the server processes a request and can prevent or redirect an operation. For example, a pre-operation plug-in can do extra validity checking on input data. Upon invocation, this plug-in can examine the LDAP request and check new values for an LDAP ADD or MODIFY request. If the values were out of range, the plug-in can respond to the client with an error message and indicate to the LDAP server that the request should not be processed.

A post-operation plug-in is called after the server processes an LDAP request and is useful for performing extra cleanup after operations and auditing. For example, a post-operation plug-in could augment the server’s standard processing of an LDAP DELETE request. Upon invocation, this plug-in, using the SLAPI interface, could search the directory and cleanup any references to the deleted entry. Another common post-operation example involves an audit.

Source code for an example plug-in can be found in the following z/OS file system:

```
/usr/lpp/ldap/examples/plugin_sample.c
```

See Figure 1 for contents.
A client-operation plug-in is called instead of the implementation-provided processing for the LDAP requests. This can be useful for providing interfaces to local data, proprietary databases, or other product directories. For example, a client-operation plug-in could translate an LDAP request as a request to a local data store or proprietary database. Therefore, if the request was an add operation, the plug-in can choose to store the data in a different format, in a different place. If the request is a search operation, the plug-in can search the proprietary database and return the data as a standard LDAP entry.

Ultimately, client-operation plug-ins allow implementers to create systems to manage any type of data by using a standard, open protocol. If Tivoli Directory Servers for z/OS back-end options do not meet your needs, a client-operation plug-in could be the answer. If local data must be enhanced to allow remote access, a client-operation plug-in could be the answer.

How do I deploy plug-ins in Tivoli Directory Server for z/OS?

Plug-ins require external configuration by using the Tivoli Directory Server configuration file. The configuration information includes:

- The type of plug-in, for example, preOperation, postOperation, or clientOperation
- The name of the plug-in DLL
- The initialization function name
- Any parameters you would like to pass to the plug-in when it is initialized.

The DLL must exist in the module search path of the address space in the Tivoli Directory Server, and must be in an APF authorized data set. When starting, Tivoli Directory Server for z/OS will attempt to load all plug-in DLLs and call the defined initialization function for each. From within its initialization function, the plug-in must register for server events. In addition, client-operation plug-ins can register for suffixes. The combination of registered server events and suffixes determines when a client-operation plug-in is invoked.

What events can a plug-in register for?

A plug-in can register for ABANDON, ADD, BIND, CALLBACK, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFIED
DN (=rename), SEARCH, or UNBIND. Note when registering for an event, a client-operation plug-in registers a suffix. For example, a plug-in can register for the search event with the o=IBM,c=US suffix. Then any searches under that suffix result in the client-operation plug-in invocation. Extended operations have no associated suffix, but only one entity (Tivoli Directory Server or plug-in) can register for a specific extended operation.
Callback is an interesting event that is called during bind processing and sometimes during ADD, DELETE, EXTENDED OPERATION and MODIFY requests. Only client-operation plug-ins can register for this event. Essentially, this event is intended to allow plug-ins to participate in the authentication and authorization calculations for a LDAP user. This includes providing a user’s password, groups, and alternate names.

For example, if a user’s groups are requested by using the callback event, the client-operation plug-in would return the user’s group list. Note the implications. The group list is used during the server’s authorization checking. Providing a new list by using the plug-in could completely alter the security constraints defined within the directory. As a result, a plug-in could play a key role in enforcing security with Tivoli Directory Server and care should only be installed if the vendor is trusted. A plug-in runs in the LDAP server’s address space, if the plug-in ABENDs it takes down the LDAP server.

Also recall the implications of deploying client-operation callback plug-ins. These plug-ins could alter the security constraints defined within the directory.

**How do I get started?**

There is a post-operation plug-in sample shipped with z/OS LDAP called plugin_sample.c and is installed in the /usr/lpp/ldap/examples directory. (See Figure 1.) This plug-in registers as a post-operation plug-in for bind operations. For a description of how to build it, see IBM Tivoli Directory Server Plug-in Reference for z/OS, “Appendix A,” SA76-0148.

Build the plug-in, and bring up a test LDAP server. Do a few binds then look at the audit file it creates. You should see a line for each bind attempt. This plug-in can be modified to try different things such as registering for unbinds too. Here is a general guideline:

1. Edit the plugin_sample.c.
2. Copy the prototype for plugin_bind_fn().
3. Change plugin_find_fn to plugin_unbind fn.
4. Copy:

```c
/*
 * Register the BIND function *
 */
rc = slapi_pblock_set(pb, SLAPI_PLUGIN_BIND_FN, (void *)&plugin_bind_fn);
if (rc != 0) {
    slapi_log_error(SLAPI_MSG_HIGH, "PLUGSAMP",
"Unable to register BIND function: error %d\n", errno);
    return -1;
}
```

5. Change SLAPI_PLUGIN_BIND_FN to SLAPI_PLUGIN_UNBIND_FN in the copy.
6. Change plugin_bind_fn to plugin_unbind_fn.
7. Change “Unable to register BIND function: error %d\n", errno); to "Unable to register UNBIND function: error %d\n", errno); in the plug-in.
8. Add the code in Figure 1 to the file.

Build sample_plugin.c and bring up the server again. Do a few binds. Now look at the audit file, and you should see Binds and UnBinds in the resulting file now.

**Closing remarks and a heads up**

Many LDAP servers, including Tivoli Directory Server for z/OS support plug-in extensions for installation specific customization of standard LDAP processing. Enterprises that leverage LDAP technology and want greater control over the LDAP server processing can deploy plug-ins for immediate results.

As an administrator, take advantage of LDAP’s robust technology. As remote access to data becomes more important to enterprises, you can use plug-ins to facilitate remote access to data or to components that lack remote capabilities. For example, you might consider plug-ins that can allow LDAP access to data stored in a proprietary format on a UNIX System Services file system.

Now a note of caution: In this article, we wanted to introduce key details that you need to consider when implementing an LDAP plug-in. Given the typical role of an LDAP server within the IT security of an enterprise, be sure to exercise care to ensure that a plug-in does not circumvent any security requirements. If you do that you’ll be able to take advantage of the endless opportunities and flexibility that plug-ins can provide.

---

**A plug-in is a user-defined extension to an LDAP server.**

**Can I have more than one plug-in?**

Yes, you can have as many plug-ins that fit into the computer’s memory and CPU. In fact, you can have a pre-operation plug-in that range checks requests for a suffix, another that performs the operation for the suffix, and yet another that audits the requests that occurred on that same suffix. The only limitations on suffixes are that only one entity (back-end or client operation plug-in) can register for a specific suffix. Remember that pre-operation and post-operation plug-ins do not register for a suffix that sees all events that they register for and choose which they handle.
Beginning with z/OS Version 2 Release 1 (V2R1), because of more and more problems caused by the shared nature of the default z/OS UNIX System Services identity, plans are in place for the V2R1 system to ignore BPX.DEFAULT.USER. The default identity is implemented with the BPX.DEFAULT.USER profile in the RACF FACILITY class. If a user isn’t defined to z/OS UNIX (that is, does not have an OMVS segment on their user profile), they run under the shared z/OS UNIX identity established by BPX.DEFAULT.USER.

Over the hills and far away
IBM introduced BPX.DEFAULT.USER very early in the evolution of z/OS UNIX as a fast path to get people accustomed to this new model, under a limited set of acceptable situations. For security and integrity reasons, the z/OS UNIX kernel restricted the use of certain functions for users running with the default identity. Over time, as z/OS UNIX became more and more integrated with OS/390, and then z/OS, even the installations that originally used the default for its intended purpose saw its users start exploiting the capabilities of z/OS UNIX in a more meaningful way. As they did, certain services did not work as expected because of the built-in restrictions; when creating files, those files were accessible by other users. The result: confusion, and several PMRs.

In V2R1, when a user has no UID or the user’s group has no GID and the user tries to perform a z/OS UNIX action, it will fail. Period.

Hey, hey, what can a user do?
Beginning with V2R1, every user who uses z/OS UNIX must have their own OMVS segment. There are two paths you can go by:

1. Assign the user segment manually: The BPX.UNIQUE.USER function, introduced in z/OS V1R11, actually assigns a permanent OMVS segment to users who lack them the first time they invoke a z/OS UNIX service. After the initial setup, the assignment happens with no action by that user or by an administrator. Further, the system is designed to assign a UID to the users that are unique when it’s assigned, and into the future.

2. Use the BPX.UNIQUE.USER function: Setting up BPX.UNIQUE.USER involves setting up other functions on which it depends. Before we describe BPX.UNIQUE.USER in detail, let’s address the prerequisites in turn. In summary, BPX.UNIQUE.USER depends on the AUTOUID and AUTOGID functions (that controls the assignment of unique IDs), which depend on shared ID control (that protects the uniqueness of the assigned IDs), which depends on the Application Identity Mapping (AIM) function (that allows fast z/OS UNIX identity mappings), which you might find out requires a bit of planning.

BPXCHECK: That’s the way
You want to know which, if any, of these prerequisites you already have implemented. To save you the trouble of figuring out how to gather all this information, we’ve created BPXCHECK to do this for you! Based on its output, you can figure out which of the functions you need to implement. See the link to BPXCHECK at the end of this article.
Ramble on: through an AIM conversion

An AIM conversion creates an alternate index structure in the RACF database that allows rapid mappings from UIDs to user IDs and from GIDs to group names. This is essential for the RACF functions that enforce uniqueness and that dole out new z/OS UNIX identity values automatically. The conversion entails running the IRRIRA00 utility in three stages that represent a progression where:

1. The alternate indices are created while the system still uses and maintains the UNIXMAP mapping profiles. This stage takes the longest to execute.
2. The system uses the alternate indices and keeps maintaining the UNIXMAP profiles, but using them only as a backup if the index lookup fails.
3. The UNIXMAP profiles are deleted.

If you have not yet performed the AIM conversion, the major planning consideration is the number of users with UID 0 (or any UID or GID value assigned in multiple OMVS segments, for that matter). If you have more than a certain amount, the AIM conversion fails. That certain amount isn’t fixed, as it depends on the lengths of the user IDs. BPXCHECK figures out if you are within the limit.

If you are not within the limit, you must replace UID 0 in some subset of your users with something else before proceeding. See Thank you for more information about this process and an informational APAR for running the IRRIRA00 utility.

The JCL used to run the IRRIRA00 utility is incredibly simple:

```
//STEP EXEC PGM=IRRIRA00,FARM=STAGE(n)
//SYSPRINT DD SYSOUT=*`
```

where n is the stage to which you are converting.

What is and what should never be: preventing UID sharing

It might seem counter-intuitive, but RACF doesn’t, by default, prevent the sharing of UIDs among users. However, after you define the SHARED.IDS profile in the UNIXPRIV class, it acts as a switch to tell RACF, “Don’t let anyone assign a UID to a user if that UID is assigned to another user.” This also applies to groups and GIDs.

If you do try to assign a UID that is in use, you will get the following error:

```
ALTUSER SHANNON OMVS(UID(452))
IRR52174I Incorrect UID 452. This value is already in use by THOMAS.
```

But what happens if you really, really need to assign a shared UID, like for a started task that does not support BPX.SUPERUSER and requires UID 0? Then a user with the SPECIAL attribute, or a user with READ access to the SHARED.IDS profile, can use the SHARED operand. For example:

```
ALTUSER BIGGUY OMVS(UID(0) SHARED)
```

This also gives you tighter control over who is allowed to create superusers!

Oh, it makes me wonder: automatic assignment of UID and GID

Think for a minute about how you assign your UID values and GID values today.

• Do you keep UID and GID values consistent with your users’ accounts on other z/OS UNIX systems?
• Do you use a serial number, or some other piece of personal data, as a UID?
• Do you use a department number as a GID?

With this example, when the AUTOUID keyword is used on an ALTUSER command, UID 500 is assigned as the UID (assuming it isn’t already in use). Then RACF increments the lower range and updates the APPLDATA to now be ‘501-50000/1-2000’.

```
ALTUSER ANDREW OMVS(AUTOUID HOMEB(/u/andrew))
IRR52177I User ANDREW was assigned an OMVS UID value of 500.
```

Keep such considerations in mind when deciding if automatic ID assignment is right for you. The RACF method of assigning an automatic UID or GID involves retrieving the last-used number and incrementing by one. If that ID isn’t used, it’s assigned. You define the range of numbers RACF can use, but you cannot choose the particular ID number using automatic assignment.

If you’re willing to use the RACF method of automatically assigning UIDs or GIDs, then keep reading.

Hot dog! Automatic unique ID assignment

Next you need to define the BPX.NEXT.USER profile in the FACILITY class. This is where you specify some details in the APPLDATA field about the range of numbers you want to assign as UIDs and GIDs. After you define this profile, you can use the AUTOUID and AUTOGID keywords to have RACF choose and assign unique IDs in user and group OMVS segments. This profile is also used by the BPX.UNIQUE.USER function.

The APPLDATA field of BPX.NEXT.USER consists of two qualifiers separated by a forward slash (/). The left qualifier specifies a starting UID value, or range of UID values. The right qualifier specifies a starting GID value, or range of GID values. You can specify “NOAUTO” in either qualifier to block RACF from assigning that type of ID, while allowing automatic assignment of the other type.

```
RDEFINE FACILITY BPX.NEXT.USER
APPLDATA('500-50000/1-2000')
```

You can change the range value at any time. Before you are tempted to lower the starting value to “fill in the gaps” left by deleted users and groups, ensure you first clean up residual references in the z/OS UNIX files (file ownership and access list entries). This prevents unintended access by the new user or group. The IRRHFS utility can help with this task (see Thank you for the URL).
Good times, better times: enabling automatic segment assignment

Finally, it’s time to talk about BPX. UNIQUE.USER. This FACILITY class profile replaces BPX.DEFAULT.USER by providing ‘on the fly’ assignment of an OMVS segment containing a unique UID or GID. Let’s see how to set it up.

First, you can optionally set up a model user profile. The OMVS fields from this user profile (except the UID field) are copied to the OMVS segment, which is created automatically for a user. If you want to set up a default HOME and PROGRAM field, here is an example:

```plaintext
ADDUSER BPXUNIQ NAME('BPX.UNIQUE.USER MODL') NOPASSWORD RESTRICTED OMVS(HOME(/tmp) PROGRAM(/bin/sh))
```

We chose the user profile name of BPXUNIQ and the description ‘BPX. UNIQUE.USER MODL’, but you can use what makes the most sense to you.

Next, define the BPX.UNIQUE.USER profile in the FACILITY class with the model user name in the APPLDATA field.

```plaintext
RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('BPXUNIQ')
```

After you define the profile, the system ignores BPX.DEFAULT.USER.

As shown in Figure 1, when a user without an OMVS segment accesses a z/OS UNIX service, an OMVS segment is created for the user with a UID derived from the BPX.NEXT.USER profile and other OMVS fields are copied from the model user profile. If the user’s group does not have a GID, an OMVS segment is also created for the group with a GID derived from the BPX.NEXT.USER profile.

When this function is creating OMVS segments for a “normal” z/OS UNIX user, after allocating the directory, you need an additional step to update the home directory field.

Bring it on home unless...

Even though plans are in place for the system to ignore BPX.DEFAULT.USER completely, you’ll probably get a “whole lotta love” from your auditors if you delete it. First, make sure you aren’t sharing the RACF database with a system that still needs it, or you might be taken to “the gallows pole”.

Thank you: for the references

For additional guidance, check out the following references. On the RACF downloads page find BPXCHECK and the IRRHFSU utility:

[ibm.com/systems/z/os/zos/features/racf/goodies.html](http://ibm.com/systems/z/os/zos/features/racf/goodies.html)

Find guidance about running the IRRIRA00 utility in the informational APAR II12972:

[ibm.com/support/docview.wss?uid=isg1II12972](http://ibm.com/support/docview.wss?uid=isg1II12972)

In the z/OS Library:
- z/OS Security Server (RACF) System Programmer’s Guide, SA23-2287 for specific advice about the AIM conversion
- z/OS UNIX System Services Planning, SA22-7800

![Figure 1. How BPX.UNIQUE.USER works](image-url)
Jennifer Ash is a Software Engineer at IBM in Poughkeepsie, New York. She is the IBM Academic Initiative System z curriculum owner, and organizes the annual Professor Summer Seminar series.

Norman Aaronson, after 38 years at IBM, retired in March as Senior Software Engineer at the IBM STG, System z Product Portfolio Organization in Poughkeepsie, New York. Involved in System z Java since 1998, he was the System z Java Project/Release Manager since 2001.

Rita Beisel is a Technical Writer at IBM. She is currently writing for z/OS MVS and has written for RACF, z/VM, DB2, and DFSMS. She also likes to do interface design, usability testing, and surveys.

Debbie Benjamin is one of the core developers for VSAM RLS. Her expertise includes record management, buffer management for VSAM and VSAM RLS.

Stephen Branch is an IBM Senior Software Engineer at Silicon Valley Lab whose 25-year career has dealt with all aspects of DFSMS. Stephen specializes in ICF Catalog, IDCAMS, and VSAM and holds several patents for these components. He is the author of the Catalog Search Interface (CSI) as well as many ICF Catalog and VSAM functions. Stephen has a Master of Science Degree from Louisiana State University.

Chris Brooker is an Advisory Software Engineer for z/OS Global Resource Serialization (GRS) Development. Chris holds an M.S. degree in Software Development from Marist College in Poughkeepsie, New York, and has been with IBM since 2001.

Javier Chavez is a DB2 on z/OS specialist working in the IBM STG Lab Services and Training organization. He has extensive experience with DB2 release migration, DB2 maintenance level migration, DB2 performance and tuning, and DB2 analytics accelerators and has helped customers with their investment in DB2 technology and features.

Bryan Childs is a Senior Software Engineer in z/OS Global Resource Serialization (GRS) Development and is the GRS Development Team Lead. Bryan holds an M.S. degree in Software Development from Marist College and has been with IBM for over 20 years.

Patricia Choi is an IBM Senior Software Engineer with over 30 years of software engineering experience, 25 of those in DFSMS. Pat has been on the DFSMS Catalog team for 14 years. She holds a Master of Science Degree in Computer Science from Purdue University.

Tariq Choudhry is an Advisory Software Engineer in z/OS Print organization. He has been most recently involved in Infoprint and transform products for Infoprint Server for z/OS test and development work and has worked on JES3 test and development. Tariq has been with IBM for 19 years and obtained his MSCS from Polytechnic University in 1990.

Michael Crump is an Advisory Software Engineer in the Tivoli z/OS software group and has worked on OMEGAMON since 1989.

Roan Dawkins works in IBM’s ICSF group in Poughkeepsie, New York, developing software for performing key management and for interfacing with IBM’s Crypto Express HSM. He holds a Masters Degree in Computer Science from Rensselaer Polytechnic Institute.

Erika Dawson is a Senior Technical Staff Member (STSM) at IBM in Tucson, Arizona. In her 25 years with IBM, Erika has spent her entire career in DFSMS Development. Most of her career has been associated with the support of DFSMSdfp OAM and the software support for IBM tape products.

David Ellis is a software developer in the OMEGAMON XE for CICS on z/OS product team. He has over 30 years experience developing IBM mainframe application and system software.

Nicole Fagen is the Parallel Sysplex Support Lead, debugger for XES, XCF and GRS. She also aids customers in configuring for high availability and leveraging resiliency options.

Mike Fox is a senior software engineer for IBM’s Enterprise Network Solutions Design and Strategy group in Research Triangle Park, working on routing and connectivity software technologies for z/OS and zEnterprise. Mike holds a B.S. degree in Electrical Engineering from Virginia Tech and has worked on mainframe networking with IBM for 24 years.

Jon Furminger is a Software Engineer for z/OS LDAP at IBM Poughkeepsie, New York. His expertise includes password policy, the LDBM back-end, and replication. Jon holds a Bachelor and Masters Degree from Binghamton University. He has been working for IBM for 23 years.

Joseph Gentile is a Software Engineer in z/OS Global Resource Serialization (GRS) Function Test. He holds a B.S. degree in Computer Science from Binghamton University and is currently pursuing his M.S. in Computer Science at SUNY College at New Paltz.

Clark Goodrich is a Senior Software Engineer at the IBM STG, System z Performance Organization in Poughkeepsie, New York. Clark’s focus is performance analysis of Java on System z.

Nina Goradia is a Senior Software Engineer for z/OSMF Development at IBM Rochester. She has been with IBM for 14 years. Her focus is on User Interface development and z/OS simplification.

Saheem Granados, CISSP*, is an Advisory Software Engineer at the IBM Poughkeepsie Development Lab and currently a member of the ITDS for z/OS development team. His expertise includes Encryption and Java Security. Saheem is with IBM for 11 years and holds a Master’s Degree from Binghamton University.

Jeff Haggar works on the design and development of z/OS Communications Server in Research Triangle Park, North Carolina, and has expertise with networking hardware and software on the System z platform. Jeff holds a B.S. degree from Rensselaer Polytechnic Institute and is an IBM Master Inventor.

* Certified Information Systems Security Professional
** Certified Secure Software Lifecycle Professional
Shigeki Kimura has been with IBM for 26 years. Migration, GA22-7499, since 2006. has participated in key reviews of z/OS release-to-release migration and has expertise in and deep knowledge of for z/OS migration at IBM Japan. He is the technical lead for z/OS migration at IBM Japan. He has expertise in and deep knowledge of z/OS release-to-release migration and has participated in key reviews of z/OS Migration, GA22-7499, since 2006. Shigeki Kimura has been with IBM for 12 years.

Mike Kasper is an Advisory Software Engineer for TSO/E Development at IBM Poughkeepsie with a BS in Computer Engineering from the University of Illinois at Urbana-Champaign. He has been with IBM for 12 years.

Shigeki Kimura is the technical lead for z/OS migration at IBM Japan. He has expertise in and deep knowledge of z/OS release-to-release migration and has participated in key reviews of z/OS Migration, GA22-7499, since 2006. Shigeki Kimura has been with IBM for 12 years.

Richard Kisley is a Senior Software Engineer for IBM System z firmware at IBM Charlotte. His expertise includes HSM application software and firmware and embedded operating systems. Richard holds a Masters in Computer Science from Duke University and has been with IBM 10 years.

Juergen Kuhn has been with IBM at the Boeblingen Laboratory for 23 years. During his IBM career, he has worked for development projects in z/OS, z/VSE, and Cloud Computing. He is currently leading the z/OS Resource Measurement Facility (RMF) development team.

Rohan Kurane is an IBM Software Engineer at Silicon Valley Lab with over 4 years of experience in the field of DFSMS. Rohan specializes in ICF Catalog. Rohan has a Master of Science Degree in Computer Science from Northern Illinois University.

Michael Lucks is the Hardware Management Console, HMC, Product Engineering team leader. Mike's team is responsible for the entire HMC hardware and firmware implementation. HMC PE Team also supports all microcode maintenance and network-related issues. Mike is a member of the System z HMC Design Review Board.

Ed Miller is an Advisory Software Engineer doing Functional Component Test for JES2 since 2007. Ed's 21 years with IBM has included experience working as a software developer, Level 3 developer and software tester in the System i® database and data communications components.

Anthony Mingo is an Advisory Software Engineer in z/OS Print at Rochester, Minnesota. His expertise includes Print Transforms from AFP for Infoprint Server for z/OS and Infoprint Server development. Anthony has been with IBM for 13 years. He likes long walks on the beach.

Marcel Mitran is a Senior Technical Staff Member with IBM Java Technology Center. He has worked on compilers and Java for System z for over a decade and is currently the lead technical architect for Java on System z.

Raymond Morse is a Senior Software Engineer in z/OS Print at Rochester, Minnesota and has recently worked on the AFP Print transforms, enhancing the Infoprint Server LPD to handle files larger than 2GB, and providing Infoprint Server 1.12 migration health checks. He has been with IBM for 29 years, enjoys working on the Infoprint Server team, and also enjoys traveling all over the United States. His goal is to visit every state.

Phil Muller is System z Parallel Sysplex & Capacity on Demand Technical Development Team Leader.

Bill Neiman is a Senior Software Engineer at IBM Poughkeepsie working in the Parallel Sysplex area of the BCP. His expertise includes couple data set management and the various interactions between z/OS and coupling facilities, and he is responsible for the computational portion of the CFsizer tool. It is this responsibility that has repeatedly forced him to confront the issues described in his article.

Mark Nelson, CISSP®, CSSLP**, is a Senior Software Engineer with IBM's z/OS Security Server Design and Development Team in Poughkeepsie New York, where he has spent the past 25 years working on RACF. Mark’s focus with RACF has been on auditing and data analysis (IRRDBU00, IRRADU00, RACFICE), RACF’s Health Checks, RACF/DB2, and RACF’s support for digital certificates.

George Ng is a Senior Certified I/T Managing Consultant for IBM Systems and Technology Group Lab Services at IBM Poughkeepsie. His areas of focus include z/OS Parallel Sysplex, High Availability, and Performance.

Margaret Patterson is a Staff Software Engineer in OMEGAMON XE on z/OS Development. Margaret holds a B.S. degree in Computer Science from North Carolina State University and has been with IBM for over 25 years.

Kathy Pfeiffer is a Senior Software Engineer at IBM in Poughkeepsie, New York. She is the Academic Initiative System z program manager.

Roity Prieto is a developer for VSAM RLS and Transactional VSAM. He specializes in data caching in the Coupling Facility for VSAM RLS.

Regine Reichert is a member of the z/OS Performance Management development team. Before that she was engaged in z/OS HCD/HCM development. She has been with IBM since 1991 at the Boeblingen Lab.

Joan Ries is an Advisory Software Engineer in Rochester, Minnesota. She recently moved from z/OS Shells and Utilities to Power Hypervisor development. While on z/OS, Joan added SMF record support to OpenSSH and a migration health check for Infoprint Server 1.12. She has been with IBM for 24 years. One of the many hobbies Joan enjoys is raising chickens.

* Certified Information Systems Security Professional
** Certified Secure Software Lifecycle Professional
**Daniel Rinck** is a zSeries® Product Engineer specializing in Channel Subsystem, Coupling Facility and Sysplex support.

**Steve Simonson** is a Senior Software Engineer working in JES2 as a developer since 2007. His experience includes substantial time working in IBM System i database, communications, work management and system high availability as a software engineer. Steve has been with IBM for 27 years.

**Jeff Sullivan** is a Senior IT Consultant in Systems and Technology Group Lab Services and Training. For over 30 years, Jeff has provided guidance and advice to scores of companies across many industries, including healthcare, retail, manufacturing, and aerospace. His expertise focuses on large-scale DB2 installations, enterprise-wide performance tuning, BI/data warehousing, and large application migrations.

**Bruce Talbott** is an Advisory Software Engineer performing development for JES2 since 2007. Bruce’s 25 years with IBM have included development and Level3 experience on System i in areas such as work management, local and remote workstations, and data communications.

**Tracee Tao** is an IBM developer for VSAM RLS in San Jose, California.

**Mike Todd** is a member of the IBM Academic Initiative System z team. He serves as team leader for the North American Master the Mainframe Contest and as the academic liaison to the Destination z community.

**Marna Walle** is an IBM Senior Technical Staff Member in the z/OS Build and Installation department at IBM Poughkeepsie. Her current responsibilities include z/OS release migration and new installation technologies of the z/OS platform. Marna is a frequent speaker on z/OS installation topics at EXPO, SHARE, and other customer conferences.

**Bruce Wells**, CISSP, is a senior software engineer for the RACF design and development group in Poughkeepsie, New York. He has extensive experience in both the z/OS and z/VM versions of the product. Bruce has been working in the RACF area for the past 20 or so years.

**Ryan Wisniewski** is an advisory software engineer in DFSMS Level 2 Support at the Tucson, Arizona, lab specializing in Catalog and IDCAMS. Ryan holds a Computer Science and Accountancy Degree from Northern Illinois University.

**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.

**Steve Warren** is a Senior Software Designer and Developer for the BCPii component of z/OS in IBM Poughkeepsie. Steve has been with IBM for 23 years.
Need a place to go for all your z/OS publications? How about trying the z/OS Information Center Library: ibm.com/systems/z/os/zos/bkserv/

You’ve probably used the z/OS Information Center on the web, but we’d like to do a little advertising. The Information Center is one-stop shopping and not just for information about the z/OS operating system or the products that run on it, but for system-level topics like planning z/OS installation and migration, zArchitecture, and even a portal to the z/OS Hot Topics newsletters.

Here’s a screen shot of the z/OS V1R13 Information Center as updated April 2012. Select **Search scope** to try out the search capability to define the scope. For example, if you only need to search for JES2 documentation, you can choose JES2 as your search scope.