HOT TOPICS

A z/OS NEWSLETTER

IBM

AUGUST 2007—ISSUE 17

Meet zIIP and zAAP

IBM's specialty processors

Also in this issue:

- Simplify your systems management and security:
  - SMF and System Logger
  - Encryption Facility for OpenPGP
  - z/OS Communications Server
  - z/OS Health Checker
  - XL C Compiler for Language Environment
- Hot Topics "How to" articles

Gita Berg
IBM Senior Technical Staff Member

Bob Rogers
IBM Distinguished Engineer

Yikes — Growth is a good thing, but this is getting scary! This issue of *z/OS Hot Topics Newsletter* is nearly 30% larger than the last one. Admittedly, we attained this bulky size by including as many articles about the new functions in z/OS V1R9 as we could lay our virtual hands on, plus hints and tips from the experts. When you see the wide range of coverage in this issue, you'll want to read every word.

Speaking of rampant growth, how do we newsletter editors deal with a sudden surge in our workload? (Do editors *scale*, by any chance?) Rather than limiting our healthy expansion, we added some *processing power* where we needed it most: The editorial staff! In this issue, we welcome two talented new managing editors, **Sue Van Parys** and **Tim Flaherty**, both of the z/OS User Technologies organization in IBM Poughkeepsie. And, keeping up the pace with the terrific art direction for this big issue is **Natasha Fray,** with graphic contributions from **Chris Olson.**

The subject of adding power where it is needed most does segue nicely into IBM specialty processors — the System z9 Integrated Information Processor (zIIP) and the System z Application Assist Processor (zAAP) are certainly generating a lot of buZZ! In our feature article, "Would you like to hear about our specials?", Bob Rogers, IBM Distinguished Engineer, traces the development of specialty processors from the early days of the S/370 Model 3033 through the rise of the RISC and CMOS processors in the 1990s, to the present day with zIIP and zAAP. These specialty processors are designed to offer price performance through existing IBM technology and provide a cost effective solution for eligible workloads. In a companion feature, "Moving right along with zIIP and zAAP applications," Gita Berg describes the exploiters of the zIIP and zAAP and points to future application enhancements that promise to help lower the overall total cost of ownership (TCO).

Beside the potential for driving down costs, this issue focuses on simplifying your work, with articles on using System Logger with SMF recording, z/OS Health Checker, controlling group capacity limits, invoking REXX routines with System REXX, and configuring z/OS Communications Server. We also look at ways of simplifying your application development through z/OS UNIX System Services, Java enhancements, and new options for XML parsers on z/OS. And, we hope our articles on encryption, RACF, SAF, and disaster recovery planning with TKE will help you feel more secure.

New to this issue, we highlight the articles that offer practical information for accomplishing common tasks on System z. This issue, our experts provide step-by-step instructions for simplifying your WLM administration, migrating to zFS file systems, viewing your z/OS file systems on Windows, and invoking SDSF through REXX commands. To make it easy for you to spot these pearls of hands-on wisdom in this hefty issue, we identify them with our Hot Topics *How-To* logo.

While you're digesting all this new material, be sure to save some room for our updates to the academic initiative geared toward educating a whole new generation of systems programmers, and the story of one IT specialist who tells us about the System z Mastery Test (back cover).

Finally, test your own skill with our very own z-Doku puzzle. (How many ways can you spell the word M-A-I-N-F-R-a-m-E?)

Is that enough for one issue of *z/OS Hot Topics Newsletter*? We are eager to know what you think of the newsletter and what you would like us to cover next. Drop us a line at **newsletr@us.ibm.com.**

A big newsletter for a big platform. Experienced users and beginners, there's room for us all — jump right in and explore the world's greatest computing platform!

The Editors


Jim Guilianelli


Paula Cross


Jodi Everdon


Tim Flaherty


Susan Van Parys


Wayne O'Brien


Natasha Fray

HOT TOPICS

Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400 USA

Executive Editor
Jim Guilianelli

Managing Editors
Paula Cross
Jodi Everdon
Tim Flaherty
Wayne O'Brien
Susan Van Parys

Art Director
Natasha Fray

Graphic Contributor
Christopher Olson

Technical Editors
John Eells
David Raften

Cover Photo
Stephen Letus

For information about the availability of future issues or to obtain hardcopy, go to: ibm.com/zseries/zos/bkserv/hot_topics.html

Contents: *Featured articles*

Back cover:

Would you like to hear about our specials?

Prelude to the zIIP and zAAP specialty processors

BY BOB ROGERS

Interested in giving your processing power some zip? Want to zap costs? In the last few years, IBM has introduced two new specialty processors called the System z™ Application Assist Processor, or zAAP, and the System z9 Integrated Information Processor, or zIIP. These specialty processors are designed to improve a system's price-performance efficiency for applicable workload. An underlying theme in the development of these types of specialty processors is the application of existing technology to solve specific problems and provide a cost-effective solution. Much of the cost reduction can be attributed to "reusing" an existing design and existing parts.

This article provides some historical perspective on specialty processors by reviewing previous types of specialty processors and demonstrating how their development led to the new zAAP and zIIP processors.

A history from the 3033 to the 9672 processors

Take the case of the IBM System/370™ (S/370™) Model 3033 in the 1970s. Before the advent of the 3033, what we used to call the central processing unit (CPU) spent a good portion of the time chit-chatting with I/O channels. As systems got larger and included more I/O channels to provide greater data bandwidth, the amount of the time the expensive CPU spent communicating with the I/O channels became a problem. To address this problem, IBM engineers created an "I/O Director" that was designed to communicate with the I/O channels on behalf of the CPU. This freed up the CPU to do more productive business computing, and improved the price-performance of the system as a whole. In order to provide the I/O Director at reduced cost, our engineers cleverly reused the CPU of a previous processor model (S/370 158) rather than design the new unit from scratch. As a result, there was a savings in the design cost, but the parts for an I/O Director were still expensive.

At IBM® research, Dr. John Cocke developed the Reduced Instruction Set Computer (RISC) concept. By greatly reducing and simplifying the instruction set and relying on compilers to knit the simple instructions together to do complex operations, one can build a much simpler, and thus cheaper, computer. One of the first commercial uses of a RISC processor was the I/O Processor (IOP) of the IBM ES/9000® Model 9021. Like the I/O Director, the IOP managed the communication between the CPUs and the I/O channels but was fabricated from considerably fewer parts because of its RISC design. Although the IOP did require some new design work, the use of a RISC processor also reduced the environmental costs of space, power, and cooling.

Once the IOP was in place, it could be enhanced to do more than just communicate with I/O channels. One of these enhancements was to provide the capability of offloading the movement of data between central storage and expanded storage. This capability, called the Asynchronous Data Mover Facility (ADMF), relieved the expensive bipolar processors from the task of simple data movement, freeing them to do other, more complex work. Again, the use of a "specialty processor" improved the price-performance of the system.

After the introduction of the ADMF, there was a lot of prognostication about offloading other specific tasks to RISC processors similar to the IOP. The idea was to offload as much work as possible from the expensive bipolar processors (the CPUs) to less expensive RISC bipolar processors. The offloaded work might have included things like sorting and certain database operations. But, this idea did not come to pass because a bigger thing happened.

In 1994, IBM introduced the 9672 processors, which were CMOS-based systems intended to run the MVS™ operating system. Because the main processors on a 9672 were fabricated in lower-cost CMOS, it didn't seem to make much sense to offload a function to a special RISC processor. Now such a special processor would be no cheaper than the main CPUs. The extra design work and additional complexity would not provide a solution any more cost effective than by just running everything on the CMOS main microprocessor. (This statement is true only if software license charges are not taken into account — but more on that later). So, the "sort engine" and the "database engine" weren't built at that time.

The next step: The IBM 9672 processor

The 9672, instead, took the next evolutionary step toward the new specialty engines of today. Where the IBM 3090™ used a previous generation processor for I/O control, the 9672 used the very same CPU chip for its System Assist Processor (SAP). The SAP, like the IOP, performed I/O channel communication, ADMF functions, and some other system communication tasks. The SAP was inexpensive because it was fabricated in CMOS and required little extra development expense because it involved no unique parts. The same technology and economics that enabled the creation of the 9672 SAP opened up other interesting possibilities.

When Parallel Sysplex® was first introduced, the coupling facility (CF) was developed using the same hardware components as the 9672. The only unique hardware was not in the coupling facility itself, but in the coupling links used to connect MVS systems to the CF. Other technologies had been proposed for the coupling facility, but for obvious reasons, using the same 9672 CPU chip, won out. This approach made possible a number of different kinds of coupling facilities — stand-alone, integrated and internal. In time, the Internal Coupling Facility (ICF) could satisfy all needs.

At first, a high availability configuration required two external coupling facilities to avoid having a single point of failure — a primary and a backup. Coupling facilities needed to be on physical hardware that was distinct from the hardware systems running the z/OS® systems serviced by the coupling facilities. Then, because the coupling facility used the same hardware as a 9672, it was possible to reduce cost using an ICF engine to place a coupling facility on hardware already running z/OS without affecting availability. Now, with CF structure duplexing, it is possible to place both the primary and the backup coupling facility on ICF engines. Being able to run both z/OS and coupling facilities on the same hardware image reduces both hardware and environment costs.

So far, the specialty processors have been "closed," that is, capable of specialized function, but not capable of running business work for customers. This changed with the introduction of the Integrated Facility for Linux (IFL). Although IFLs cannot run the z/OS operating system, it can run Linux on z and z/VM®. With Processor Resource/Systems Manager™ (PR/SM™), customers run many Linux on System z images on a single hardware system. With the help of z/VM, you can run hundreds of Linux on System z images. The fact that the IFL was based on the same 9672 microprocessor chip immediately gave Linux on System z the virtualization capabilities of PR/SM and z/VM with little extra development cost. This powerful virtualization is at the heart of the excellent cost of ownership provided by Linux on System z.

But what about those licensing fees? IBM does not include the capacity provided to a system by IFLs when considering software license fees that are based on the total capacity of a system (Monthly License Charges or MLC). This allows the customer to enjoy the lower hardware and environment costs driven by system consolidation without additional software license charges from IBM.

zAAP and zIIP specialty processors

This now brings us to the new specialty processors, zAAPs and zIIP, introduced on zSeries® and System z systems. They are based on the same processor chip as the standard processors, SAPs, ICFs and IFLs. Although zAAPs and zIIPs cannot run z/OS, z/OS can manage them and use them to run certain types of work.

See "Moving right along with specialty processors" on page 6, a companion to this article. It provides details on what types of work can be run on zIIPs and zAAPs and what that means to you and your customers, and to your IBM software license charges.

Always moving toward the future

This article has shown how IBM's specialty processors have evolved, both in their effectiveness and in their affinity with z/OS. The deployment of specialty processors like the zIIPs and zAAPs that are based on common technology and parts has made a great change in the mainframe price-performance landscape. And that evolution has been a good thing. ■

Put some zIIP and zAAP in your z/VM

Are you interested in exploring specialty engines? z/VM V5.3 can help.

The latest release of z/VM includes the following support for virtual specialty engines:

- **Simulation** - z/VM provides specialty engines for guest virtual machines but dispatches them on real Central Processors (CPs). This lets you try out zIIPs and zAAPs before buying them.
- **Virtualization** - z/VM provides specialty engines for guest virtual machines and dispatches them on the real specialty engines that match their type, if the specialty engines are available in the z/VM LPAR processor configuration.

Using this support is easy — just complete the following steps:

1. Enter the DEFINE CPU command to define virtual central processors of type zIIP or zAAP.
2. IPL z/OS in the virtual machine.

The virtual specialty engine is then available for z/OS use. For more details about this support, see *z/VM V5R3.0 Running Guest Operating Systems*, SC24-6115, online at **ibm.com/vm/library**. For more information about z/VM, see **ibm.com/vm/zos**.

Moving right along with specialty processors

What's on the menu for zIIP and zAAP applications

BY GITA BERG

Specialty processors have a long, reliable, and cost-effective history on the mainframe. The availability of the System z9™ Integrated Information Processor (zIIP) and System z Application Assist Processor (zAAP) can help make processor-intensive applications cost competitive on System z.

Overview

As the article "Would you like to hear about our specials?" on page 4 suggests, the IBM zIIP and zAAP are conceptually the same. z/OS redirects portions of eligible workloads from general-purpose processors to the zIIP and zAAP specialty processors. Because today's applications often require a lot of CP-intensive power on the general purpose processor, shifting some workloads to the zIIP and zAAP might help make processing these applications more cost effective.

Here's what we mean: Specialty processors like the zIIP and zAAP give you a "technical dividend." You can buy a zIIP or a zAAP specialty processor once and upgrade it at no additional charge. Let's say you have a z990 and you purchase a zAAP for it. When you upgrade the z990 to a z9™ EC, you don't repurchase the zAAP or pay for any net incremental zAAP capacity because of the increased processor size of the z9 EC. This continues to be true when you upgrade the z9 EC to some future machine. In essence, you can invest in the specialty processor technology and reap the benefits for a long time.

Table 1 looks at the similarities between the zIIP and zAAP.

zIIP and zAAP similarities	zIIP	zAAP
Are processors that run asynchronously with other general purpose processors	Yes	Yes
Are available on z/OS V1R6 and above	Yes	Yes
Can receive eligible work from z/OS but cannot IPL with z/OS, or any other operating system	Yes	Yes
Support On/Off Capacity on Demand (On/Off CoD), Capacity Upgrade on Demand (CUoD) for the non-disruptive addition of one or more engines, and Capacity BackUp (CBU) for emergency situations	Yes	Yes
Can be managed by z/OS workload manager (WLM)	Yes	Yes
Do not require additional IBM software charges (as of the date of this publication there is also no known instance where an ISV has imposed software charges)	Yes	Yes
Uses the PROJECTCPU tool to measure eligible workload	Yes	Yes
Moderately priced	Yes	Yes

Table 1. zIIP and zAAP: Similar

Yet, as shown in Table 2, zIIP and zAAP do differ in important ways.

zIIP	zAAP
Introduced 2006.	Introduced 2003.
Available on IBM System z9 Enterprise Class and Business Class (z9 EC and z9 BC) – available on z9 servers only, hence the name System z9 Integrated Information Processor.	Available on IBM System z9 Enterprise Class and Business Class (z9 EC, z9 BC) and IBM eServer™ zSeries 990 and 890 (z990, z890) – available on multiple machine generations, hence the name System z Application Assist Processor.
Intended to help integrate data and transaction processing from across the enterprise to System z9.	Intended to help implement new application technologies on System z.
Most data and transactions are being accessed remotely to z/OS; the underlying technology is the z/OS ability to redirect portions of enclave SRB work to the zIIP.	Most applications are executing locally on z/OS; the underlying technology is the z/OS ability to redirect portions of TCB mode work to the zAAP.

Exploiters include:
• DB2® for z/OS
• Data serving
• Data Warehousing
• z/OS Communications Server
• Network encryption
• z/OS XML System Services.
	Exploiters include:
	• Anything that uses Java® through the IBM SDK, that is IBM Java Virtual Machine (JVM), such as:
	  • WebSphere® Application Server
	  • IMS™
	  • DB2
	  • CICS®
	• z/OS XML System Services.

Table 2. zIIP and zAAP: Only different

zIIP exploiters: Choices, choices, choices!

The zIIP is intended to help integrate data serving and transaction processing across the enterprise on the System z9, starting with z/OS V1R6 (with Web deliverable and PTFs), DB2 Universal Database™ (UDB) for z/OS Version 8 (with PTFs), DB2 Version 9.1 for z/OS, and System z9 servers with appropriate maintenance levels. DB2 gives z/OS the necessary information to have portions of eligible workloads directed to the zIIP. zIIP can help reduce System z9 utilization, increase "white space," and lower overall total cost of ownership (TCO) for eligible DB2 workloads.

DB2 data serving

Applications that run on UNIX®, Windows®, z/OS, or Linux on System z and access DB2 through the z/OS DB2 Distributed Database Relational Architecture™ (DRDA®), the way DB2 Connect™ does, for example, can have portions of their workload directed to the zIIP. About 40 percent of the DRDA workload is eligible to run on the specialty processor. This might vary if DB2 stored procedures, user-defined functions, or VTAM®, CICS, IMS, and other workloads associated with the Database Management System (DBMS) are not run in enclave service request blocks (SRBs). Using zIIP can help:

- *Grow applications:* Existing enterprise, business intelligence, and Web-based applications that access DB2 remotely, which were restricted by DB2 utilization, can now grow more easily.
- *Reduce the amount of siloed data:* With the reduction in TCO for remote access to DB2, more applications can access the same DB2 data store and need not pull down a separate copy of the data.

Data warehousing with DB2

Applications that utilize DB2 for z/OS for long-running parallel queries can have most of their SQL (and SQL/XML) requests directed to the zIIP. The results can be significant with up to 80 percent of the long-running parallel query workload eligible for zIIP. Results will vary depending on the application and the amount of parallel processing. You can run workloads locally or remotely.

Use zIIP to provide real-time business intelligence (BI) and data warehousing (DW). For example, zIIP can significantly reduce the amount of processing capacity on general-purpose processors, while maintaining consistent overall execution times of a parallel query workload; thus, using zIIP might make it more cost effective to run BI and DW against your real-time online transaction processing (OLTP) data. You can run workloads locally or remotely over DRDA.

Securing network traffic with zIIP-assisted IPSec

z/OS Communications Server can use zIIP for portions of IPSec processing. This capability is available on z/OS V1R8 (with PTFs) and later when the zIIP is present. Use the TCP/IP configuration statement IPCONFIG IPSECURITY or IPCONFIG6 IPSECURITY and then a single configuration statement (GLOBALCONFIG ZIIP IPSECURITY) within the TCP/IP profile. That will trigger z/OS Communications Server to request z/OS to direct this IPSec enclave SRB processing to the available zIIP.

Using zIIP-assisted IPSec can help reduce the total cost of ownership of IPSec encryption, which can help you to establish virtual private networks for your mainframe network traffic. The amount of workload eligible for zIIP depends on the network operation. All IPSec processing from incoming network bulk data transfers runs in enclave SRBs and, therefore, is eligible for zIIP, as is most IPSec processing from outgoing network bulk data transfers. The z/OS outbound bulk data transfer starts in TCB mode, but responses from thereceiving server result in the remainder of the transfer being completed in any enclave SRB. Only a portion of the IPSec processing from interactive work is eligible for zIIP. Interactive workloads are — well — interactive, and have a significant amount of outbound communication from z/OS that runs in TCB mode and are therefore ineligible for zIIP.

zAAP exploiters: Choices, choices, choices!

The zAAPs can help implement new application technologies on System z. Java programming under control of the IBM Java Virtual Machine (JVM) was the first to use zAAP. Let's see what uses zAAP and the effect on application workloads for System z.

Java

The JVM function became available with z/OS V1R6 and the IBM Software Developer's Kit (SDK) for z/OS, Java 2 Technology Edition, V1.4 (with PTF) and SDK for z/OS, Java 2 Technology Edition, V5. Modifications to the JVM for z/OS detect when an application is calling Java code or when Java calls out to non-Java code. These JVM modifications signal the z/OS dispatcher that the unit of work can move between a CP and a zAAP.

Here's a sample of the products or applications whose eligible Java workloads can benefit from the use of zAAP:

- WebSphere Application Server V5.1 and V6.0.1
- Java applications running with CICS for z/OS V2.3 and later
- CICS support for any programs written in Java in the CICS environment
- IMS V7 (and later) transactions using Java programming
- DB2 V7 stored procedures using Java
- Any application that is written to the IBM SDK
- Java batch.

Use zAAP to integrate Java Web application mission-critical data for high performance, reliability, availability, and security. Also, you can lower TCO for WebSphere Application Server and other Java technology-based applications with the zAAP.

How much work is eligible for zAAP? It's highly variable because it depends on the amount of Java application code in your applications.

Because today's applications often require a lot of CP-intensive power on the general purpose processor, shifting some workloads to the zIIP and zAAP might help make processing these applications more cost effective.

Java based XML

You can run Java based XML parsing on zAAP today. There's a Java XML Parser and a Java XSLT Processor included in the IBM SDK for z/OS, Java 2 Technology Edition, V1.4 and higher. For information, see **ibm.com/servers/eserver/zseries/software/xml/**.

zIIP and zAAP specialty processors and XML parsing

On April 18, 2007, IBM announced its intent to enhance z/OS XML System Services to exploit zIIP and zAAP specialty processors. (What is z/OS XML System Services? See "Meet the parsers z/OS and XML — a different approach" in *z/OS Hot Topics Newsletter Issue 15*, August 2006, GA22-7501-11.) z/OS XML System Services exploitation of zIIP and zAAP processors is expected to be rolled out over time.

Today, XML processing running under DB2 9 can already be partially directed to zIIPs when part of a distributed request (like DB2 DRDA). But in the future, IBM intends to direct all eligible z/OS XML System Services processing to zIIPs when it forms part of any zIIP eligible workload. Specifically, all z/OS XML System Services parsing workload run in any enclave SRB is planned to be redirected to the zIIP, not just the 40% associated with DRDA.

In addition, z/OS XML System Services is also enabled to exploit the zAAP specialty processor. Specifically, all z/OS XML System Services parsing performed in TCB mode is eligible to be executed on the zAAP processor. The immediate exploiter and benefactor of this enhancement is DB2 9 for z/OS. This means that locally attached applications (ISV or homegrown) can store XML data in DB2 with the TCO benefit of the zAAP processor without any anticipated changes to the application. Eligble for zAAP processing are z/OS XML System Services parsing from individual XML document inserts, XML updates requested from local thread or stored procedures, and bulk table loads. The function is available with z/OS V1R9 and will be rolled back to z/OS V1R7 and V1R8 with PTFs.

The new z/OS XML System Services capability means that XML processing on DB2 9 for z/OS becomes more cost effective. For more information on z/OS XML System Services workloads, see **ibm.com/systems/z/ziip/** and **ibm.com/systems/z/zaap/**.

Figuring it all out

If you suspect you are currently running one of the workloads described earlier, the z/OS PROJECTCPU option in IEAOPTxx (available since z/OS V1R6) might help determine if a zIIP or a zAAP is right for you.

Setting the IEAOPTxx SYS1.PARMLIB member option PROJECTCPU=YES directs z/OS to record the amount of work eligible for zAAP or zIIP processors. SMF Record Type 72 subtype 3 is input to the RMF™ post-processor. The Workload Activity Report lists workloads by WLM service class. In this report, the field APPL% AAPCP indicates which percentage of a processor is zAAP eligible, while the field APPL% IIPCP indicates which percentage of a processor is zIIP eligible. SMF Record Type 30 provides more detail on specific address spaces.

Because the price of zIIP and zAAP specialty processors is less than that of general purpose processors, values as little as 25% for APPL% AAPCP or AAPL% IIPCP can make a new zAAP or zIIP processor worth your while. Here are the requirements:

- z/OS V1R6 for measuring Java workloads on zAAP
- z/OS DB2 data serving and long running parallel queries workloads on zIIP
- z/OS V1R8 (with PTFs) for zIIP Assisted IPSec workloads
- z/OS V1R9 for z/OS XML System Services and zAAP.

If you are not running one of these workloads, you cannot use PROJECTCPU. Instead, see the sizing guidance documents at **ibm.com/systems/z/ziip/resources.html** and **ibm.com/systems/z/zaap/resources.html**.

Use these guidelines for any sizing considerations you might have. Your IBM representative can also assist you.

Addressing the future

As you can see, zIIP and zAAP are moving beyond Java and DB2 to branch out into z/OS XML System Services and IPSec network encryption. The zIIP and zAAP specialty processors might help make these processor intensive technologies more attractive to host on the mainframe. Another way to look at it: you want to implement the latest technology, IBM System z is helping you get there with a great menu of application choices! ■

Kicking down the costs of XML

BY JUN OGATA

We know that the cost of XML processing is high, so anything that helps reduce that cost is good news, right? With z/OS V1R9, and through a PTF for V1R7 and V1R8, z/OS XML System Services is now able to take advantage of the lower-cost zAAPs, if one is available on your system, to help reduce the cost of computing when parsing an XML document.

The system does this zAAP enablement under the covers when any program calls the main XML parser module, GXL1PRS (GXL4PRS), or its new C/C++ equivalent service for z/OS V1R9, glxpParse(). After the parser completes its processing, the system turns off zAAP enablement.

Most XML functions can take advantage of zAAPs, although some specific processing does not. Exit routines associated with the parser do not run on the zAAP; the system turns off zAAP enablement before a call to any of these exit routines, and turns it back on when the exit routine returns control to the parser. Even though the z/OS XML parser can run in SRB mode, system limitations prevent the parser from running on a zAAP while in SRB mode.

IBM plans to roll zAAP enablement support down to z/OS V1R7 and V1R8 in PTFs for APAR OA20308 so that all supported releases of the XML System Services parser can take advantage of a zAAP.

In the future, IBM intends to add support for zIIPs to the XML System Services (z/OS XML) parser. With this support, the parser will be redirected to the lower-cost zIIPs when running in enclave SRB mode. With respect to DB2, processing that can be partially directed to zIIPs like DB2 Distributed Relational Database Architecture (DRDA) will further benefit by directing the full amount of the z/OS XML System Services processing to the zIIPs.

In addition, IBM intends to add parser validation support to z/OS XML System Services, and to enhance the XML Toolkit for z/OS so that eligible workloads can exploit z/OS XML System Services. Both of these additions will exploit the ability of the z/OS XML parser to use zAAPs. ■

zAAPs: They're not just for Java any more

XML parsing enhancements

The Statement of Direction (SOD) in the April 18, 2007 IBM z9 EC and z9 BC announcement alludes to future XML processing support on z/OS. You can view Announcement 107-190 at **ibm.com/vrm/newsletter_10577_1179_29223_email_DYN_1IN/qunt28633311**. Of particular note is the planned ability to direct XML parsing operations performed by XML System Services (also called z/OS XML) to zAAP and zIIP specialty engines. The described zAAP support has been announced, and is available with z/OS V1R9, along with z/OS XML C/C++ API support, satisfying another previous SOD. The zAAP support is of particular interest to DB2 Version 9.1 for z/OS users because the DB2 Version 9.1 for z/OS PureXML support already utilizes z/OS XML in certain contexts.

Among other items, the April SOD also indicates that a future enhancement to the XML Toolkit for z/OS will allow eligible programs using the included XML4C parser to optionally request that XML4C utilize z/OS XML, thus providing additional opportunity to take advantage of specialty engines.

SMF's got a new set of wheels!

Using System Logger with SMF

BY STEVEN B. JONES

Truth be told, SMF recording has not seen much development in recent years. While z/OS images have been steadily growing because of consolidation trends and the development of larger processors, the only recent change in SMF was to allow more storage for buffering, which, while worthwhile, did nothing to improve the SMF recording rate.

Of course, one thing you could always say about SMF is that, when it's not overloaded, it's stable and dependable — which is why installations build their billing and auditing on SMF. In fact, where data movement is concerned, you might even think of SMF as the reliable z/OS "family station wagon," safely transporting the family to its destination, albeit in the slow lane (yawn).

That was then. Time for dependable old SMF to visit that special garage at the end of the street in z/OS Development. In z/OS V1R9, we handed over the "keys" to our mechanics. While the new SMF still retains all that dependability that so impresses parents, in z/OS V1R9, SMF has moved into the fast lane!

Let's take a closer look at SMF, the new hot rod in z/OS Town.

SMF and System Logger

In z/OS V1R9, SMF now has the option of recording data using the System Logger instead of writing to a set of SYS1.MAN data sets. System Logger has the advantage of buffering large amounts of data in memory or in a coupling facility (CF) structure, which means that it can do much larger write operations at greater efficiency than can the basic SMF recording method.

Another "speed bump" in using SMF MAN-style data sets has been the "switch" process, whereby SMF finishes using one MAN data set, marks it "DUMP REQUIRED," and starts filling another data set. This switch process can cause large installations to lose SMF data as SMF buffers are overrun during the switch. With System Logger, there is no physical need to do a switch. A log stream is continuous, with System Logger maintaining the underlying data sets internally, so that the exploiting program, SMF, sees no break in data. (You might still have a need for the SWITCH SMF command; we will return to that idea in a moment.)

Super highway, super SMF

Besides the change to use System Logger, we further streamlined SMF throughput by adding multiple lanes of traffic. In z/OS V1R9, SMF allows you to make use of multiple log streams, selected by record type. You can easily increase SMF's capacity, subject to the limits of the underlying System Logger infrastructure, just by separating records into different log streams. And, if required, you can send the record type to several different log streams.

This enables some interesting new capabilities for SMF. You can separate the records strictly by number, with records 0-63 going to one log stream, records 64-128 going to a second log stream, and all user records (types 129-255) going to a third.

Or, you can separate the records by purpose. For example, you can send record types 30, 70, 71, 72 and 99, typically used for performance studies, to one log stream, while you can send records needed for auditing purposes, such as types 30, 80, 81 and 83, to a different log stream. Because Type 30 is needed for both uses, it is sent to both log streams.

Figure 1 shows the typical flow of SMF data.

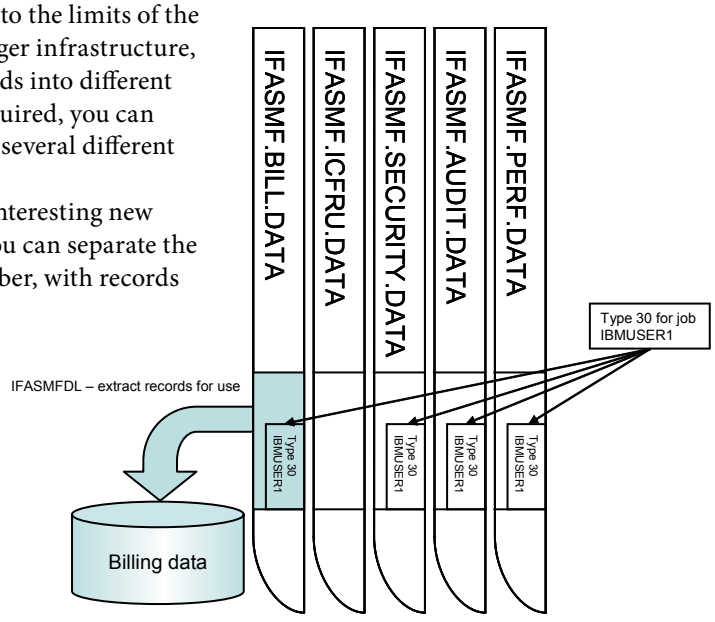


Figure 1. A single record can be written to several log streams and extracted as needed

By separating the data by purpose, you have segmented the data so that the application that uses it does not have any records to filter out or ignore. The savings in back-end processing for these applications can be significant.

Using the data from the log stream, or "Will it take longer to unpack the car?"

z/OS installations have built significant procedures around the recording, dumping and archiving of SMF data, usually with significant automation involved. Changing where SMF records its data can be simple or painful, depending on how your installation implements dumping and archiving.

In z/OS V1R9, we added a new dump program *IFASMFDL*, which is designed to dump data from the SMF log streams. Using syntax similar to IFASMFDP, you can dump and archive data from a log stream and write it to the same archive data sets, such as any GDG.

As with IFASMFDP dump jobs, you can drive IFASMFDL dump job with an automated SWITCH SMF command or through a new user exit, similar to IFAU29, called *IFAU29L*. Alternatively, you can just submit it through your usual batch JCL procedures.

In the past you had to run IFASMFDP against a SYS1.MAN data set several times to extract, archive, and disseminate the data to the different groups for processing. With the data segmented to different log streams at the time it is written, much of this reprocessing is eliminated. In addition, we have enhanced IFASMFDL to allow you to specify more filter keywords (DATE, START, END, SID) on the OUTDD statements. The result is that an IFASMFDL job can read a log stream once and create as many data sets as needed from that data, based on date, time, SMF ID (SID), and record type. This reduces, if not eliminates, the reprocessing required and makes the task of maintaining SMF data much less onerous.

Figure 2 shows how IFASMFDL extracts SMF data for use or for archiving.

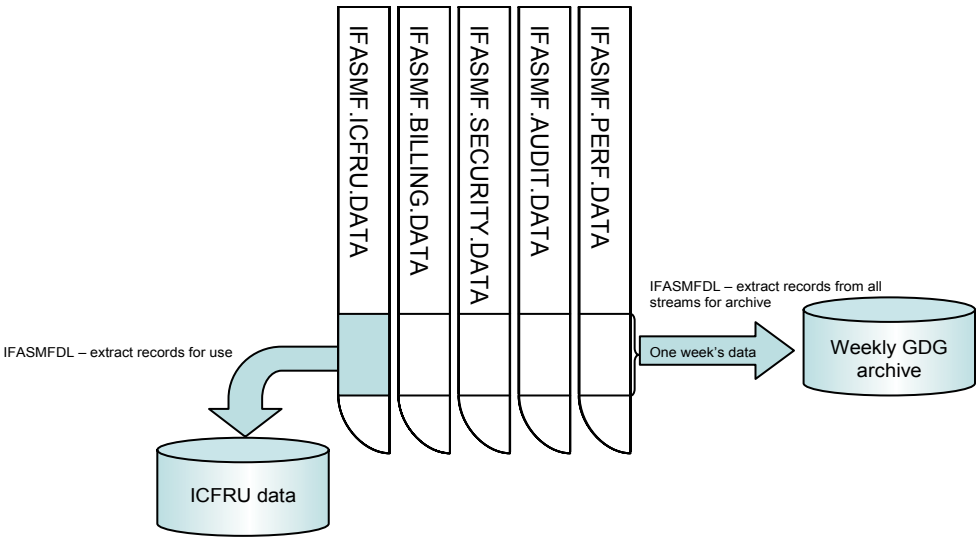


Figure 2. When you use log streams, SMF data is captured and immediately available for use or archiving

Where the rubber meets the road

If you are interested in using the new SMF recording to System Logger function, the first thing to do is plan, plan, plan!

First, map out your current SMF record type usage and recording rates, as these numbers are essential when setting up your SMF log stream environment and tuning your log stream parameters.

Next, be sure to note the required retention periods for different record types. Some types, such as Type 101 for DB2 accounting, can simply be re-obtained if discarded, while others, such as audit records, must be retained for many years.

Finally, read *Systems Programmer's Guide to: z/OS System Logger*, which is available at **www.redbooks.ibm.com/abstracts/sg246898.html**.

While not updated with SMF information as of this writing, the information on basic System Logger functions will help you understand such concepts as CF-based versus DASDONLY log streams, staging data sets, data class attributes, and so on. If you decide that one or more of your SMF log streams will be CF-based, the CFSIZER application can help you decide on the correct structure size and attributes for an optimal balance between coupling facility (CF) structure storage and SMF data throughput.

After you have defined the log streams, you can create a new SMFPRMxx parmlib member and new jobs to dump those log streams.

Here's a quick overview of the new SMFPRMxx keywords.

LSNAME(IFASMF.q1.q2,TYPE(aa,bb:cc,dd))

This keyword specifies that you want to direct the named record types to the specified log stream. Note that all SMF log streams must begin with IFASMF, or the LSNAME keyword will be rejected. You can specify this keyword multiple times, but you cannot specify the same log stream name more than once. The specified record types use syntax similar to TYPE in the SYS/SUBSYS keywords but do not allow subtypes to be specified. Be aware that the record types specified in the SYS/SUBSYS keywords are the types that the system will record. Any "extra" record types specified on the LSNAME parameter are ignored unless the SYS/SUBSYS parameters are changed to record that type.

DEFAULTLSNAME(IFASMF.q2.q3)
This keyword specifies that any record type that does not match the LSNAME keywords to be written to this log stream. This optional keyword can simplify the specification of record types in the LSNAME keywords. If it is not specified and the SYS or SUBSYS parameters include types that are not specified in the LSNAME parameters, error messages can result.

Take note!

Do not delete your SMF MAN data sets or remove them from the SMFPRMxx parmlib member! A problem can arise that causes SMF to stop recording to the log stream. As with any new function, errors in programming or operations can cause SMF to stop recording to the log stream. By determining and fixing the root cause (such as increasing the size of a CF structure) and resuming SMF recording through the SETSMF RECORDING(LOGSTREAM) command, you can resolve many problems.

However, SETSMF RECORDING(DATASET) allows the operator to switch back to SYS1.MAN-style recording dynamically, minimizing the possibility of an outage and the effects of lost data. Be sure to plan for extracting data recording to MAN-style data sets in the event of a fallback to data set use after an error.

…System Logger now gives SMF the horsepower to record more of the data you want….

RECORDING(DATASET|LOGSTREAM)
This keyword specifies which recording technique SMF uses. DATASET is the default, allowing you to continue to use your old SMFPRMxx parmlib members without taking any migration actions. The RECORDING parameter value can be changed dynamically through the SETSMF command to start recording to log streams or to fall back to MAN-style data sets in the event of an error.

Finally, you'll have to determine the best way to get the switch command and dump program invoked in your installation. This depends greatly on your existing strategies for invoking IFASMFDP. The infrastructure built around dumping data sets (switch command, messages such as IEE366I, and exits such as IFAU29 have parallel functions in the log stream recording area so that you can use similar techniques for dumping the data).

With System Logger, you'll arrive safe and sound!

System Logger has been developed for reliability as well as speed. Exploiting System Logger now gives SMF the horsepower to record more of the data you want, while retaining the reliability of the family station wagon.

Now's the time to start thinking about trying out a new set of wheels for your SMF auditing! ■

Cruising the limits of z/OS

Using group capacity limit

BY ROBERT VAUPEL AND GUENTER VATER

How do you keep your car traveling at the speed limit and avoid a speeding ticket? You use cruise control! How do you control processor consumption of your computer system? You apply a defined capacity limit for z/OS on a System z logical partition (LPAR).

A defined capacity limit (also called a softcap) is one of the most efficient ways to control million service units (MSU) consumption on your z/OS LPARs. You define an MSU limit for the partition, and z/OS Workload Manager (WLM), together with the Processor Resource/ Systems Manager (PR/SM) hypervisor, ensures that the defined limit is enforced. WLM checks this by measuring a four-hour rolling average of MSU consumption for the z/OS system and tells the PR/SM hypervisor to cap the partition when that average exceeds the defined capacity limit. As long as the average stays below the defined limit, you can temporarily exceed the limit. Going back to our analogy, this means you can temporarily exceed the speed limit and not get a ticket!

Using this technique allows you to control the MSU consumption of individual z/OS partitions, but does not help you manage the MSU consumption of multiple z/OS systems against one common defined capacity limit. Yet, in many installations, workloads peak at different periods during the day. While one system is running heavy batch work during the night, another partition might use most of its capacity during online hours, and yet a third partition for testing purposes might use its capacity only sporadically. Today, if you want to limit the MSU consumption of these partitions, you must apply individual softcaps for each partition and ensure that each individual softcap provides enough capacity for the workload peaks so that work can run in each partition without too much degradation. This leads to periods when only a small part of the defined capacity is used.

The new way to drive

This is where group capacity enters the highway. Group capacity, available on System z9 servers, provides an effective way to apply one softcap against a group of partitions while ensuring that each partition can use its share of the defined capacity limit. When some partitions do not use their share, the unused capacity is distributed across the partitions that require more capacity, thus reducing the overall amount of unused capacity.

Defining a capacity group with HMC

You can define a capacity group on the Hardware Management Console (HMC). You can also define multiple groups, but a logical partition can be a member of only one group at any point in time. To define a capacity group and add members to it, you must perform the following steps:

1. Define a group name for the capacity group.
2. Add to the group the logical partitions that form the capacity group.
3. Define a group capacity limit.
4. Understand the effect of the weight definitions of the partitions and their actual demand to decide how much CPU can be consumed by each partition in the group.

Define the group name and the group limit by adding a new group profile, and then you can add the partitions to the group on the HMC. Additionally, it's necessary to understand which share of the group capacity each partition in the group is entitled to use. The partition weight decides the share or target MSU consumption of each partition.

Group		Partition				
Name	Capacity Limit	Name	Weight	Softcap	Target MSUs	Max MSUs
MyGrp1	810	S51	200	n/a	270	810
		S5D	100	300	135	300
		S5H	300	n/a	405	810

Table 1. Capacity group example

Table 1 shows an example of a capacity group consisting of three partitions: S51, S5D, and S5H. The target MSU consumption is calculated by applying the share of the partition weight (200 for S51) based on the sum of partition weights (600 for S51, S5D, and S5H) to the capacity limit (810). In other words, when all the partitions are using as much MSU as possible, S51 is entitled to use 270 MSU (or 33.33% of 810 MSU).

Assuming these are the only partitions on the CPC, the target MSU values are 270, 135, and 405. The target MSU value is the smaller value based on the partition softcap and its group weight. Partition S5D has a softcap of 300 in addition to the group limit MSU value of 135. If S51 and S5H do not use the full amount of MSU, it's possible for S5D to get up to 300 MSU. It's also possible for S51 and S5H to get the maximum MSU value (810) if it not being used by another partition. This means that a partition will never violate an individual capacity limit, regardless of whether that partition is part of a group.

Enforcing the limit

WLM on z/OS enforces the group capacity limit. Each partition must run z/OS V1R8 or later to be managed within a group limit.

The z/OS system manages itself by collecting hardware-consumption data from the other members of the group — just as if it were managing within a defined capacity limit. However, there are two differences:

- The individual value that the system is managed against is recalculated continually based on the target MSU value and the amount of capacity that is not used by other partitions.
- The total consumption across all partitions of the group is collected to calculate the four-hour rolling average of the group. Using this value ensures that capping is applied at the right moment.

A partition is capped when the four-hour rolling average of the group exceeds the group limit and the partition exceeds its target MSU value plus the additional capacity it can borrow from partitions that don't need it. The latter is the more complex part of the control mechanism. The target MSU value is the amount of capacity a partition is always entitled to use.

Looking at the previous example, let's assume partitions S51 and S5H want to use more capacity than the target MSU value allows for and S5D only uses 15 MSU. The remaining 120 MSU for S5D are apportioned between S51 and S5H based on their weights. S51 now can get two fifths of the 120 MSU (or 48 MSU), making its new limit 318 MSU; and S5H can use the remaining three fifths, for a new limit of 477 MSU. The three partitions still are not using more than 810 MSU, assuming the four-hour rolling average already exceeds the 810 MSU limit.

Group capacity example

Let's look at a complete example. Figure 1 shows three partitions forming a capacity group with a group limit of 50 MSU. No individual capacity limits are defined for the members of the group. The three partitions IRD3, IRD4, and IRD5 have different weights of 52, 102, and 152, which entitle them to use different amounts of the group capacity.

Partition	Limit	Weight	Target MSU
IRD3	n/a	52	~8.5
IRD4	n/a	102	~16.7
IRD5	n/a	152	~24.8
Group	50	306	50

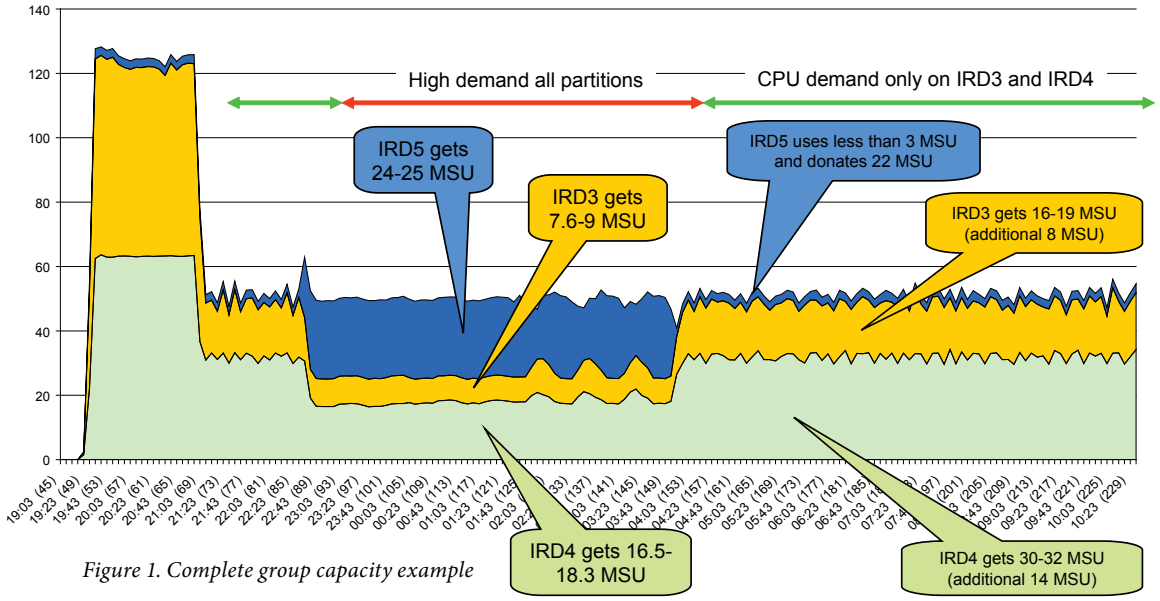


Figure 1. Complete group capacity example

Directly after the IPL, a capacity bonus of up to four hours exists (similar to defined capacity limit). This allows the partitions to use more capacity during the IPL phase. All three partitions are IPLed at the same time. IRD3 and IRD4 start work immediately, which consumes about 60 MSU on each partition. IRD5 runs only a small amount of work.

After about two hours, the IPL bonus is exhausted and the four-hour rolling group average now exceeds the 50 MSU limit. Partitions IRD3 and IRD4 are now capped. The three partitions are entitled to use around 8.5, 16.7 and 24.8 MSU each. Because IRD5 uses only around three MSU, the remaining 22 MSU are distributed between IRD3 and IRD4. IRD4 has the higher weight and therefore is able to obtain two-thirds of the available 22 MSU (around 14 MSU), which allows it to use between 30 and 32 MSU. IRD3 therefore gets around eight additional MSU from IRD5 and can use between 16 and 19 MSU.

After two more hours, work starts on IRD5, causing a noticeable spike. After that spike, each partition can use only its target MSU limit. Because the workload varies, the consumption is not fixed at the target MSU limit. The workload on IRD5 runs for about six hours and abruptly stops, causing a negative spike. Then, IRD3 and IRD4 once again divide the unused MSU from IRD5 and run in a similar manner as before.

Much more than cruise control

Group capacity limit allows you to use MSU more efficiently. In addition, group capacity limit works with WLM functions such as LPAR CPU Management (part of Intelligent Resource Director, or IRD), which manages the weight and logical CPUs of a partition. The partitions that form the group do not have to run in the same sysplex, because each partition collects its information directly from the hardware and manages itself independently from all other partitions. In addition, it's possible to dynamically configure partitions into and out of a group.

As you can see, group capacity limit is much more than the individual cruise control for your z/OS systems. It entitles you to allow z/OS systems to use more processing resource when other systems in the same group do not use their share. It also ensures that you can stretch your capacity plan across multiple images running on the same CPC. ■

Share and share alike

Virtualize your tape libraries with IBM Integrated Removable Media Manager

BY SUSAN GREENLEE AND WOLFGANG MUELLER

In today's world, the demand for storage capacity is voracious. Tape is still one of your primary options for economical long-term storage of archive and backup data for both the mainframe and distributed systems. It's relatively inexpensive and allows you to store a very large quantity of data in a very small area. Given that it can be withdrawn from direct access, you can sleep at night without worrying about waking up to deal with viruses and worms (not to be confused with the *Dipylidium caninum*, more commonly known as the tape worm) having corrupted data. But there might be one or two problems still nagging you.

How can I get higher utilization of hardware resources by sharing libraries, drives, and media between heterogeneous systems?

Sharing resources between several applications connected to the storage network requires corresponding mechanisms for access control, access synchronization, and access prioritization. These mechanisms need to control who can access which hardware when, so that potentially all applications can access all resources available in the storage network.

How can I effectively manage removable media?

In large environments, removable media management must be able to catalog hundreds of thousands of media, storing not only the media and their attributes, but also a history of access, duration of use, corresponding data about errors, and other information. In addition, it is possible to store the media separately from drives, which means that the system must additionally know the location of a medium at all times, whether the location is a manually managed store or an automatic library.

So, how can you share the physical resources across your storage network and effectively manage removable media? Enter IBM Integrated Removable Media Manager (IRMM), just announced for Linux on System z, a management middleware product that provides tape library virtualization and tape storage management.

What is IRMM?

IRMM is middleware that implements the IEEE Standard 1244 for removable media-management systems. It resides between existing applications and hardware that is connected through the storage network and shields the applications from the details of the tape hardware. It receives commands from the applications, evaluates them, and instructs the hardware to work accordingly.

How does IRMM provide higher utilization of tape hardware?

IRMM allows you to virtualize tape libraries. It combines the capacity from multiple heterogeneous libraries into a single reservoir of tape storage that can be managed from a central point. This allows all applications to share a common pool of tape drives and eliminates the need to exclusively assign drives to applications. Instead, IRMM distributes all application workload among all working tape drives. For example, empty cartridges are gathered in a common pool of scratch cartridges that can include cartridges from multiple physical

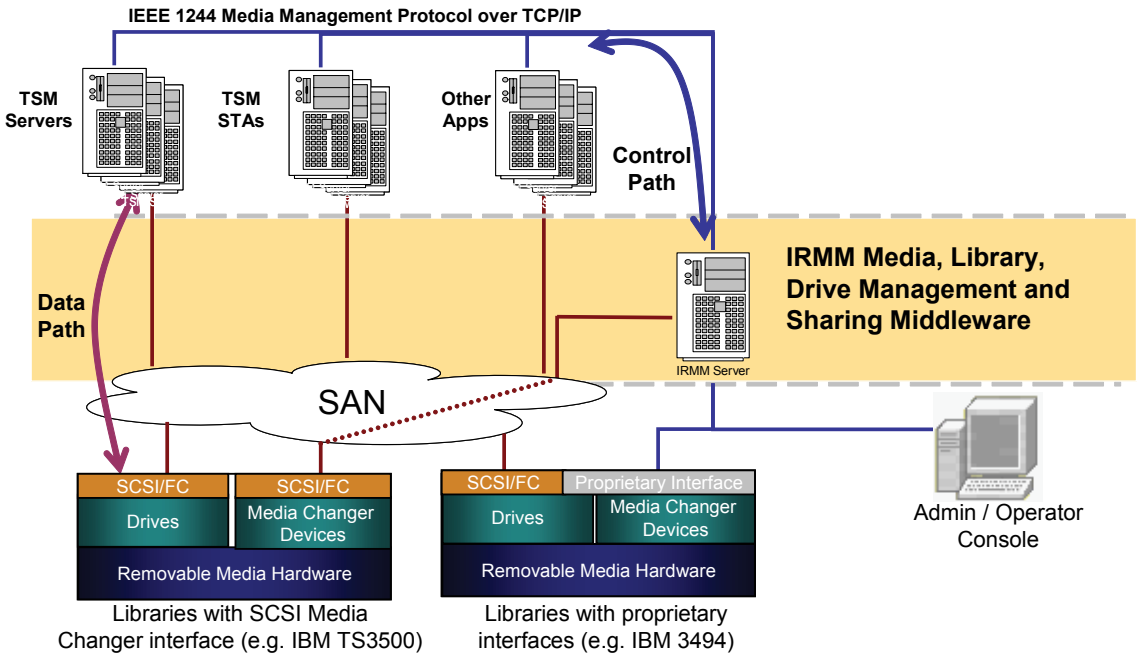


Figure 1. The IRMM architecture

libraries. While a common scratch pool can be shared among all applications, don't worry! IRMM still provides a private pool for every application to keep its private cartridges. The built-in security mechanisms of IRMM prevent applications from reading the private cartridges of other applications.

How does IRMM manage removable media?

For tape storage and resource management, IRMM provides centralized management and administration through policy-based drive allocation, cartridge allocation, and media life-cycle management of open-systems removable media. It provides monitoring and reporting capabilities such as drive utilization, data throughput, and compression rates. It also provides reporting capabilities for error statistics for root causal analysis of performance bottlenecks, and for the implementation of preventative and predictive maintenance. See Figure 1.

How does IRMM work?

IRMM is composed of 4 core components:

- The Media Manager acts as a central repository for the metadata (the data about the data) that describes the storage resources and policies. It also provides mechanisms to control and coordinate the allocation and use of libraries, drives, and media between multiple client applications. The Media Manager uses a DB2 database. (DB2 V8.2 or later, running on any platform, is a supported product for IRMM.)
- The Library Manager manages library hardware, specifically, the library media changers or cartridge accessors, on behalf of the Media Manager. The Library Manager depicts the contents and capabilities of libraries to the Media Manager. One Library Manager is installed for each tape library to be managed by IRMM. Library Managers are available for all IBM tape libraries and for StorageTek ACSLS-controlled libraries.
- The Host Drive Manager (HDM) manages storage device hardware on behalf of the Media Manager and depicts the contents and capabilities of a storage device to the Media Manager. Each client (for example, the host server running the TSM server) that needs to interact with IRMM must have an HDM installed.
- An External Library Manager (ELM) agent. The ELM agent interacts with Tivoli® Storage Manager (TSM) to manage removable media (TSM V5.3 or higher, running on any platform, is a supported product for IRMM). TSM server tracks and manages client data. The ELM agent, operating entirely outside the I/O data stream, chooses cartridges and drives, labels and tracks physical volumes, and controls the drive, slots, and doors. The ELM agent separates the complexity of media management tasks from the data management tasks performed by the TSM server. It provides a virtualization layer that hides the physical details of the hardware and presents logical library images to the TSM server. The ELM agent must be installed on every TSM Server that is to connect to IRMM managed libraries.

Figure 2 depicts a configuration of IRMM and TSM with libraries that include an SCSI Media Changer interface.

What does IRMM do for me?

When an architected solution, such as IRMM, separates the data from the metadata, and thus the data flow from the control flow, it is called out-of-band virtualization. (For an explanation of the differences and advantages of in-band and out-of-band virtualization, see Technote **www.redbooks.ibm.com/abstracts/tips0203.html**.)

Through its virtualization capability IRMM:

- Has the capability to map a library image across multiple physical libraries.
- Eliminates definition, setup, configuration, and maintenance of drives and drive paths used by TSM Servers. For example, assuming a setup with three TSM servers, 30 storage agents, and 30 drives. Without IRMM, nearly 1000 drive paths have to be defined and maintained.
- Keeps track of the right path from the TSM Server/STA to the drive even if drive paths change because of connection problems, drive maintenance, or failures.

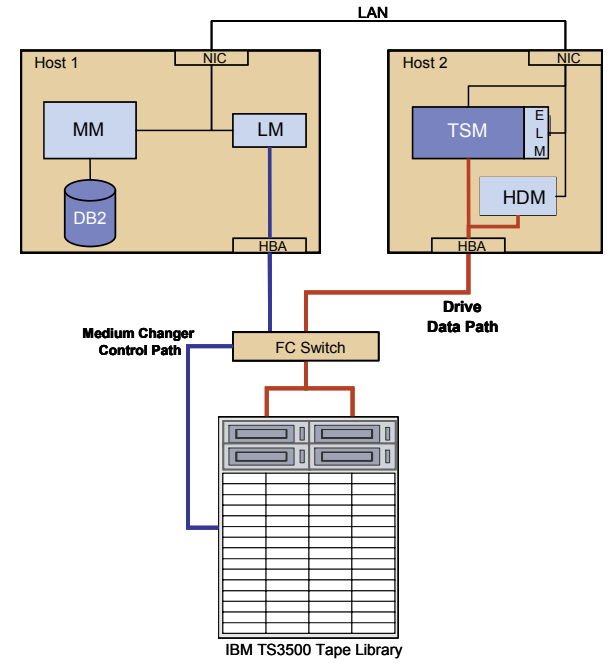


Figure 2. Configuration of IRMM and TSM with an IBM TS3500 tape library

- Distributes workload among all drives or libraries an application has access to.
- Queues mount requests in case no empty drive is available and reschedules mount requests when a drive becomes available again.

Through its media management functionality, IRMM provides the following:

- Extended monitoring and reporting. IRMM keeps track of all cartridges, usage patterns, recording drive, cartridge errors and supplies a complete history of mount requests for auditing and reporting.
- A common scratch pool that can be shared by all applications and a private pool of allocated volumes for each application.
- A security model for tape media including access control.

Can I integrate IRMM management with DFSMSrmm?

Sure you can. IRMM provides a basic integration with DFMSMSrmm™ that enables management of all removable media resources in the enterprise from one central control point. DFSMSrmm introduced a Web Service API in z/OS V1R7 that IRMM utilizes to make the cartridges used by distributed systems available to DFSMSrmm.

The DFSMSrmm repository is continuously updated by IRMM. It ensures that:

- Existing volumes in IRMM are added to the RMM repository
- When a new volume is allocated by IRMM, it is reflected in the RMM repository
- Changes in volume usage that IRMM manages are reflected in the RMM repository.

You get a unified treatment of all removable media resources in the enterprise. All cartridges are contained in the RMM database and all DFSMSrmm reporting and monitoring tools and procedures that formerly only applied to z/OS can now be applied to the distributed environment as well.

Are there more benefits for z/OS users?

Yes! IRMM provides a command line interface for z/OS that allows a z/OS administrator to fully administer, control, and monitor the distributed tape infrastructure from z/OS! All IRMM functions are available through this interface so both environments can be managed from a central z/OS system.

We can't help you with back problems, back taxes, or back payments, but we hope your back-up solution is simplified with the new IRMM! ■

So, how *can* you share the physical resources across your storage network and effectively manage removable media?

What's under the hood of System REXX?

BY JIM MCGURL, COLIN MOSCHEN, AND RICH PREWITT

Implementing an efficient set of operational tasks is critical to many aspects of today's z/OS system operations. That's why a new z/OS system component named System REXX has been developed to allow system programmers and authorized programs to invoke REXX routines outside of traditional z/OS TSO/E and batch job environments. With this new support, REXX routines can receive control through a z/OS operator command or through an authorized programming interface.

System REXX allows you to take full advantage of the strengths of REXX routines. For example, System REXX allows authorized programs to interface with traditional REXX capabilities, such as parsing.

In this article, we outline the potential uses of System REXX and provide you with tips for using this powerful new function in z/OS.

Start up System REXX!

How hard is it to get started? No worries! This support is available on both z/OS V1R8 and V1R9. With a couple of brief configuration steps, you can drive away with this new capability.

First, you need to set up the new SYS1.PARMLIB member named AXR00, which contains two key settings:

- System REXX command prefix facility (CPF)
- User ID under which the security environment is established for AXREXX invocations with SECURITY=BYAXRUSER for REXX exec processing.

Next, be sure to catalog SYS1.SAXREXEC on your system. This is where your System REXX execs will reside. Make sure you monitor control of this z/OS data set and you know who is modifying routines within it to avoid a disruptive situation.

The AXR address space starts up at IPL time on z/OS. After it's active, you can begin to run System REXX execs in the environment.

Enter the highway with REXX execs

After the AXR address space is up and ready to use, those execs can start real work. z/OS provides two types of interfaces for initiating the execution of System REXX execs. The traditional interfaces are through batch jobs, the TSO/E environment, and the z/OS UNIX environment. In z/OS V1R8 and above, you can invoke REXX execs through a programming interface, AXREXX and initiate them using a CPF on the z/OS system console.

AXREXX is an authorized programming interface that allows the invoking program to run REXX execs in a TSO/E-like environment (option TSO=YES) or non-TSO-like environment (option TSO=NO). Each has its own applicability. AXREXX provides new TSO Server address spaces (AXR01-AXR08), which are created for TSO=YES invocations.

Additionally, the interface allows the invoking program to have the System REXX routine run synchronously or asynchronously. If an asynchronous request takes too long to run, AXREXX provides a way to cancel it through a request token.

The TSO=YES environment provides the exec with isolation, the ability to allocate data sets, and the use of a subset of TSO host commands. The TSO=NO option provides an environment for execution that uses fewer resources with 64 worker tasks. Each option gives the user a choice for their System REXX execs.

Another handy way to drive these system execs is with the CPF established in AXR00 at IPL. A simple command from the console or SDSF, such as 'MYCPF SYSEXECA' provides you with a quick mechanism for kicking off diagnostic or action-oriented REXX routines. All invocations can provide a time limit. You can invoke System REXX execs from TSO/E REXX execs using the CPF capability using the ADDRESS CONSOLE construct. These always default to run in the TSO=YES environment supported by System REXX.

Standard features included

With the introduction of this new environment, you benefit from features that make the use of System REXX even more desirable. Similar to other REXX environments, System REXX allows the use of compiled REXX execs using a runtime processor, which is available at no extra cost. However, keep in mind that there is no performance boost using the IBM Alternate Library for REXX.

System REXX also delivers a set of functions to use in REXX routines that give it some **real** horsepower:

- AXRCMD() allows the exec routine to issue system commands through MGCRE and responses to the command can be returned in stemmed variables.
- AXRWTO() and AXRMLWTO() allow the System REXX routines to issue single line and multiple line Write-to-Operator (WTO) messages to z/OS.

Caution: Possible speed bumps

Whenever you take to the open road, you need to be able to handle a number of possible situations. System REXX arrives at the showroom with a full set of tools in its toolbox, which can help you understand what is going on inside the System REXX engine.

You can monitor, in general and detailed ways, how System REXX exec routines are running within the System REXX address spaces using the console Modify command:

```
F AXR,SYSREXX STATUS
```

This provides you with a quick overview of the number of active System REXX tasks in the AXR and AXR01-AXR08 address spaces.

To get more information about individual System REXX routines, specify:

```
F AXR,SYSREXX STATUS,D
```

This gives you more information about your individual routine's execution.

Note that System REXX establishes some thresholds:

- A maximum of 64 non-TSO System REXX routines can run at one time.
- A maximum of eight TSO server address spaces can run System REXX routines at one time.
- If the number of requests surpasses 5000 requests, System REXX rejects the new requests until the system falls below 4000 queued requests.

If you need to gather detailed tracing information about System REXX on z/OS, you'll be interested in the set of z/OS component trace options for SYSAXR. There's also SYS1.PARMLIB support for member CTIAXR00, which is used during initialization of the AXR address space.

If, for some reason, you need to stop the System REXX support and restart it, you can use the following commands respectively:

```
FORCE AXR,ARM
S AXRPSTRT
```

Take advantage of the horsepower

System REXX enhances the power of message processing facility (MPF) exits with the inherent capabilities of REXX. MPF exits have been in use in z/OS environments for years and now this support can extend that.

An MPF exit can be active for more than one specific message or message prefix. Our team built a single MPF exit that could handle many different message IDs.

Here's an example of the exploitation power of this new support. Our single MPF exit used the AXREXX authorized service to invoke System REXX execs passing the message text. The name of the called exec is the message identifier itself. For example, if the message specified in the MPFLSTxx for the MPF USEREXIT is IRA100E, the name of the called exec will be IRA100E. This capability allows this single MPF exit to invoke many different REXX execs. Each REXX exec must then process its own message appropriately.

A primary strength of REXX is its function-rich parsing capability. What could be better than exploiting that parsing strength out of an MPF exit? Taking that parsing chore out of assembler and placing it into a language that handles it without effort, and reduces overall complexity. With this approach, a System REXX exec can parse the message. Then, based on what it finds, the exec can perform certain functions. For example, it can, issue other commands to gain more data, shut down or start other programs, or write out information to data storage.

The use of a generic MPF exit combined with System REXX has many uses in test environments. When doing overnight testing, your installation can set MPF exits for messages of different components or applications. The component or application owners can code up their own execs to do specific functions based on the messages that they are monitoring with MPF. This provides two advantages. First, not all component and application testers would have to cover all test shots. Second, during overnight runs, if there are messages that show possible problems, they can be trapped and a System REXX exec can run to attempt to determine the actions to take or what data to collect for diagnostic purposes.

Accelerate

You are now officially licensed to drive the new System REXX functions on z/OS. As with any drive, start out cautiously until you know the road. Then, you can begin to make your operations more efficient and effective.

As always, a set of detailed publications can help guide you down the road to System REXX. Start with *z/OS MVS Programming: Authorized Assembler Services Guide,* SA22-7608, which is included in the z/OS Internet Library: **ibm.com/servers/eserver/zseries/zos/bkserv/.** ■

This time around

Moving to Server Time Protocol (STP)

BY NOSHIR DHONDY

In a Parallel Sysplex, time synchronization is of *unparalleled* importance. Your applications, subsystems, and z/OS systems need to check the time frequently to do essential things like determining the sequence of serialized events for updating a database or log file. Moreover, with the potential for processes to be running concurrently on different servers, the Parallel Sysplex must be able to maintain time consistency across multiple servers.

Since 1990, the Parallel Sysplex has used the IBM Sysplex Timer® to synchronize time among servers. Now, IBM has introduced the Server Time Protocol (STP) feature, which is designed to provide the capability for multiple servers and coupling facilities to maintain time synchronization with each other, without requiring a Sysplex Timer.

With availability of STP, IBM has withdrawn from marketing the IBM Sysplex Timer Model 2 and its features; see Hardware Announcement 906-137, dated June 27, 2006.

Let's look at what STP will mean to you.

STP overview

STP is a server-wide facility implemented in the Licensed Internal Code (LIC) of the IBM System z9 Enterprise Class (z9 EC), System z9 Business Class (z9 BC), zSeries z990, and z890 servers. (In this article, the term *server* also includes standalone coupling facilities.)

STP uses a message-based protocol in which timekeeping information is passed over the externally defined coupling links between servers. These coupling links include InterSystem Channel-3 (ISC-3) links configured in peer mode, Integrated Cluster Bus-3 (ICB-3) links, and Integrated Cluster Bus-4 (ICB-4) links. These can be the same links that you already use for coupling facility (CF) message communication.

By using the same links to exchange timekeeping information and CF messages, STP is capable of meeting the key requirement that the time of day (TOD) clocks of servers be synchronized within the fastest messaging time between those servers. Servers exchanging messages over ICB-3 and ICB-4 links are now able to meet more stringent synchronization requirements, compared to servers exchanging messages over ISC-3 links, which are not only slower but also typically used to connect servers over longer distances. STP supports a multi-site timing network of up to 100 kilometers (62 miles) over fiber optic cabling, thus allowing a sysplex to span these distances and overcome the previous 40-kilometer limitation of Sysplex Timer links.

What is a CTN?

STP introduces a new concept, *Coordinated Timing Network (CTN)*, which is a collection of servers and coupling facilities that are synchronized to a time value called *Coordinated Server Time (CST)*. The servers that make up a CTN share a common identifier, which is called a CTN ID. Only servers with the same CTN ID can join the same CTN.

Using an external time standard

If your installation has specific requirements to provide accurate time relative to an external time standard, you can use STP to set Coordinated Server Time to within 100 milliseconds (ms) of an international time standard, such as Coordinated Universal Time (UTC), by dialing out to a time service from the Hardware Management Console (HMC). STP can also track to UTC by scheduling periodic dial-outs.

Configuring a CTN

You can configure a CTN in either of two ways:

- *Mixed CTN*, which requires a Sysplex Timer that provides the timekeeping information. A Mixed CTN allows concurrent migration from an ETR network to a timing network using STP.
- *STP-only CTN*, which does not require a Sysplex Timer.

ETR to STP-only CTN migration

With proper planning for hardware, software, and connectivity, your installation can migrate concurrently from an ETR network to an STP-only CTN, and maintain continuous operations throughout the migration process.

To do so, your installation must do the migration in two phases:

- First, migrate the ETR network to a Mixed CTN.
- Then, migrate the Mixed CTN to an STP-only CTN.

The sections that follow describe the tasks involved in each phase of the migration.

ETR to Mixed CTN migration

Begin by migrating the ETR network to a Mixed CTN. The tasks involved with this phase of the migration are categorized as follows.

Preparation tasks

To make a server STP-capable, you must:

1. Install the STP LIC (MCLs) concurrently on every server that is to be configured in the CTN.
2. Migrate your z/OS systems to z/OS V1R7 or later.

Installation tasks

Before configuring a Mixed CTN, you must:

1. Install STP feature number 1021 on every server that is to be STP-enabled. The HMC now includes panels that you can use to configure and manage a CTN.
2. Apply the software maintenance for STP to all z/OS V1R7 and later systems.
3. Determine whether you need to update the CLOCKxx parmlib member.

Activation tasks

To allow your servers to exchange STP messages, you must define the CTN ID to each server.

The CTN ID is of the format [STP network ID] - [ETR network ID]. (Do not change the ETR network ID already assigned to the ETR network.)

A Mixed CTN requires at least one STP-configured server to be synchronized to the Sysplex Timer. We refer to this server as the *Stratum 1* server. For high availability, it is recommended that you have at least two Stratum 1 servers.

Servers synchronized to the Stratum 1 are called *Stratum 2* servers and servers synchronized to the Stratum 2 are called (you guessed it) *Stratum 3* servers. STP supports up to Stratum 3 servers.

Example

Assume that your installation has the following:

- Three STP-capable servers, z9 EC, z9 BC, and z990, which are configured in an ETR network.
- Logical partitions P1, P2, P3, and P4, which are z/OS V1R7 systems in a Parallel Sysplex, using internal coupling facilities on servers z990 and z9 EC.

Currently, this Parallel Sysplex uses coupling link connectivity for STP messaging.

In this example, our objective is to migrate the configuration to the Mixed CTN shown in Figure 1, with the following attributes:

- The z9 EC, z990, and z9 BC participate in the Mixed CTN.
- The z990 and z9 EC remain in ETR timing mode, synchronized to the Sysplex Timers.
- The z9 BC is migrated to STP timing mode, synchronized to either the z990 or the z9 EC using STP message exchanges.

Preparation steps

In our example, we installed the required LIC (concurrent MCLs) on the z9 EC, z9 BC, and z990 servers.

We did not need to do any software preparation tasks because all of systems are z/OS V1R7.

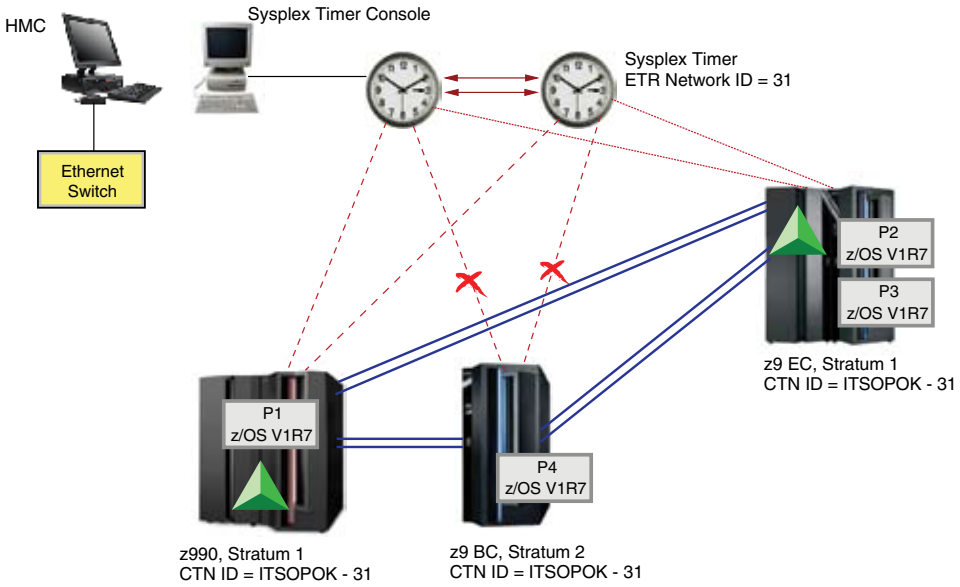


Figure 1. Migrating to a Mixed CTN

Installation steps

In our example, we did the following:

1. Installed the STP feature concurrently on each server.
2. Installed the z/OS maintenance on all logical partitions (P1, P2, P3, and P4). We will need to reIPL the systems for them to recognize the installed STP feature.
3. Updated the CLOCKxx parmlib member, if needed.

Activation steps

In our example, we did the following:

1. Assigned the same CTN ID to the three servers: [ITSOPOK] - [31]. They remain synchronized to the Sysplex Timer and continue running in ETR timing mode.
2. Disabled the ETR ports of the z9 BC, which recognizes the loss of the Sysplex Timer signals and automatically switches to STP timing mode. Note that the server requires coupling link connectivity to at least one Stratum 1 server. The server now synchronizes to either the z990 or the z9 EC using STP messages.

Mixed CTN to STP-only CTN migration

To prepare for concurrent migration to an STP-only CTN, use the HMC to assign the following roles to the server:

- *Current Time Server* (CTS). This is the active Stratum 1 server.
- *Preferred Time Server* (PTS). This is the preferred Stratum 1 server.
- *Backup Time Server* (BTS) (optional). This server can take over as the Stratum 1 server.
- *Arbiter* (optional). This server can assist during recovery.

In our example, the z9 EC is assigned the role of PTS and CTS, the z990 the role of BTS, and the z9 BC the role of arbiter.

When you assign the roles at the HMC, STP also performs the following actions:

- Disables the ETR ports of the z990 and z9 EC servers in the Mixed CTN
- Removes the ETR network ID from the CTN ID for all servers in the CTN.

The Mixed CTN is now migrated to an STP-only CTN, as shown in Figure 2.

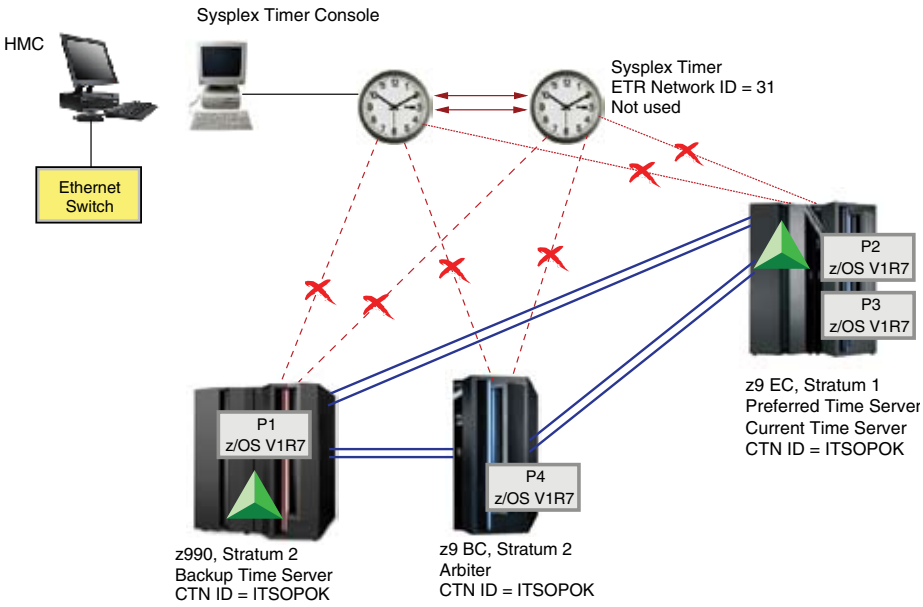


Figure 2. Migrating to an STP-only CTN

Find out more

STP is designed to meet the time synchronization requirements of current and future generations of System z servers.

For more information, see:

- STP Web site: **ibm.com/systems/z/pso/stp.html**
- IBM Hardware Announcement letter 106-715, Server Time Protocol for IBM System z9, zSeries 990 and 890; non-raised-floor support for System z9 BC, October 10, 2006.
- *Server Time Protocol Planning Guide, SG24-7280, and Server Time Protocol Implementation Guide*, SG24-7281, available at the IBM Redbooks™ Web site: **www.redbooks.ibm.com**
- Introduction to Server Time Protocol (STP) course available on the Education page of IBM Resource Link™ web site: **ibm.com/servers/resourcelink**. ■

Don't shell yourself short

The z/OS UNIX Directory List Utility

BY E. OZAN BARAN, ALFRED LEASE JR., AND FREDERICK LATES

When it's time to work with MVS data sets, most of us experienced ISPF users are accustomed to reaching for ISPF option 3.4 — the ISPF Data Set List Utility (DSLIST). Accustomed? Heck, we like using it. In fact, some of us (maybe you) would prefer a DSLIST-like utility for working with z/OS UNIX System Services files, too, so we can get out of our (ahem) shells for a while.

Now we can.

The z/OS UNIX Directory List Utility, available since z/OS V1R8, provides a convenient, panel-based method for working with z/OS UNIX directories and files from the TSO/ISPF environment. It's like the ISPF DSLIST utility but for z/OS UNIX directories and files.

File under "handy"

Depending on your particular work preferences, the z/OS UNIX Directory List Utility provides a handy alternative to having to start up the shell when all you want to do is work with some z/OS UNIX files. The z/OS UNIX Directory List Utility complements the z/OS UNIX shell and the ISPF shell command, ISHELL.

Shelling new from old

You still need the z/OS UNIX shell and the ISHELL command to do system administrative tasks, like working with file systems (mounting and unmounting them, adjusting their sizes), and for working with the system itself rather than files or directories.

Here are some specific examples of when you would use the z/OS UNIX shell or the ISHELL command instead of the z/OS UNIX Directory List Utility.

ISHELL command: Use for managing z/OS UNIX file systems — for example, mounting or unmounting file systems and modifying certain attributes. You also need the ISHELL command to enter most shell commands. With ISHELL, you use ISPF dialogs instead of shell commands to perform these tasks.

UNIX shell: Use for manipulating the execution environment — for example, choosing between different shell types such as /bin/sh or /bin/tcsh. You also need the z/OS UNIX shell for setting environment variables, creating and maintaining archive libraries, setting security labels, and running the dbx debugger.

ISPF option 3.17 gets you there

To access the z/OS UNIX Directory List Utility, select option 17 of the ISPF Utilities menu (which you reach through option 3 of the ISPF main menu). When you select this option, you will see the z/OS UNIX Directory List Utility panel (ISRUULP). The layout and options of the Directory List Utility are similar to those of the ISPF DSLIST utility. You can either display the directory list under the specified path name for further processing or print the directory list to the ISPF list data set.

DSLIST-like … or just DSLIST-ish?

How DSLIST-like is this user interface, you ask? Have a look at Figure 1, which shows the main menu of the z/OS UNIX Directory List Utility.

```
 Menu  Utilities  Options  Help
                  z/OS UNIX Directory List Utility

   blank Display directory list            P Print directory list

   Pathname . . . /u/oz2

Enter "/" to select option
 /  Confirm File Delete
 /  Confirm Non-empty Directory Delete

When the directory list is displayed, enter either:
  "/" on the directory list line command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
  "=" to execute the previous command.


Option ===>
 F1=Help      F2=Split     F3=Exit      F4=Expand    F7=Backward  F8=Forward
 F9=Swap      F10=Actions  F12=Cancel
```

Figure 1. z/OS UNIX Directory List Utility main panel

The z/OS UNIX Directory List Utility helps you create, edit, delete, browse, rename, copy, and replace z/OS UNIX files from within ISPF.

More importantly, the functions are very similar to that of the ISPF DSLIST utility. The z/OS UNIX Directory List Utility helps you create, edit, delete, browse, rename, copy, and replace z/OS UNIX files from within ISPF.

Figure 2 shows a sample input panel from the z/OS UNIX Directory List Utility.

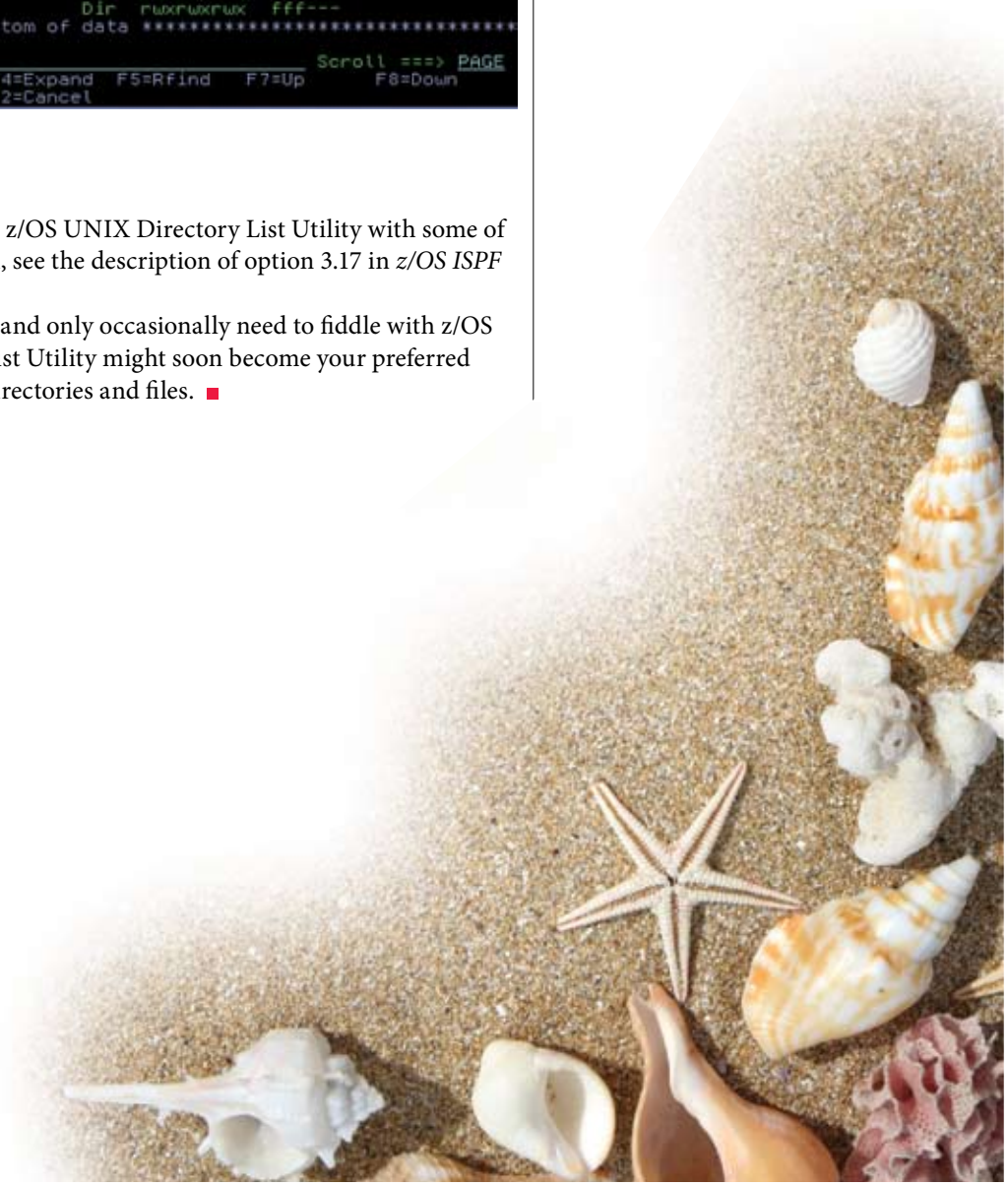
```
 Menu  Utilities  View  Options  Help
                        z/OS UNIX Directory List          Row 1 to 11 of 11
Pathname . : /u/oz2
Command  Filename        Message          Type Permission Audit  Ext  Fmat
---------------------------------------------------------------------------
________ .                                 Dir  rwxrwxrwx  fff---
________ ..                                Dir  r-xr-xr-x  ------
________ .ishell-reflist                   File r--------  fff--- --s- ----
________ .profile                          File rw-r--r--  fff--- --s- ----
________ .sh_history                       File rw-------  fff--- --s- ----
________ .ssh                              Dir  rwxr-xr-x  fff---
________ murielTest                        Dir  rwxrwxrwx  fff---
________ LDAP                              Dir  rwxrwxrwx  fff---
________ OA18012                           Dir  rwxr-xr-x  fff---
________ Tools                             Dir  rwxrwxrwx  fff---
________ USS                               Dir  rwxrwxrwx  fff---
****************************** Bottom of data ********************************
Command ===> _____________________________________________ Scroll ===> PAGE
 F1=Help    F2=Split   F3=Exit    F4=Expand  F5=Rfind   F7=Up      F8=Down
 F9=Swap    F10=Left   F11=Right  F12=Cancel
```

Figure 2. z/OS UNIX Directory List panel

What are you waiting for?

Now that you know that it exists, try the z/OS UNIX Directory List Utility with some of your daily to-dos. For more information, see the description of option 3.17 in *z/OS ISPF User's Guide Volume II*, SC34-4823.

If you do most of your work in ISPF, and only occasionally need to fiddle with z/OS UNIX files, the z/OS UNIX Directory List Utility might soon become your preferred method for manipulating z/OS UNIX directories and files. ■



HOW TO view your z/OS file systems on Windows

BY RENATA RAND MCFADDEN AND JODI EVERDON

Do you have a favorite Windows or Linux application that you want to use with z/OS files? Are you tired of using FTP to access your z/OS data sets on your workstation?

z/OS Distributed File Service Server Message Block (SMB) server eliminates the FTP nightmare and makes it easier to work with z/OS files and printers using Windows and Linux (print support requires z/OS Infoprint® Server).

What SMB supports

SMB supports sequential data sets, partitioned data sets, partitioned data sets extended (PDSE), and Virtual Storage Access Method (VSAM) data sets. Yes, that's right, you can view the Record File System (RFS). SMB even supports Windows Terminal Server on Windows 2003 and Linux. Here's what you need to know to get started.

What the administrator needs to know

SMB has a configuration file with environment variables to tune its settings. The configuration file is read during process initialization. This file is located in /opt/dfslocal/home/dfskern/envar. There's also a sample environment variable file located in /opt/dfsglobal/examples called dfskern.envar.

Some of the environment variables are:

- _IOE_PROTOCOL_RPC=OFF, which specifies whether the DCE DFS™ protocol is supported
- _IOE_PROTOCOL_SMB=ON, which specifies that SMB processing is on
- _IOE_DYNAMIC_EXPORT=ON, which allows dynamic export
- _IOE_SMB_IDMAP=/opt/dfslocal/home/dfskern/smbidmap, which specifies the location of the smbidmap file, which maps SMB user ID to the z/OS user ID.

To access specific files on z/OS, you must configure the file systems for export and create shares where users connect. An SMB share is a shared resource and is defined as a local resource on a server that is accessible to SMB clients (in our case, Windows and Linux) on the network. Three /opt/dfslocal/var/dfs/ files define the static export and the shares. The examples shown in this article are for illustrative purposes only.

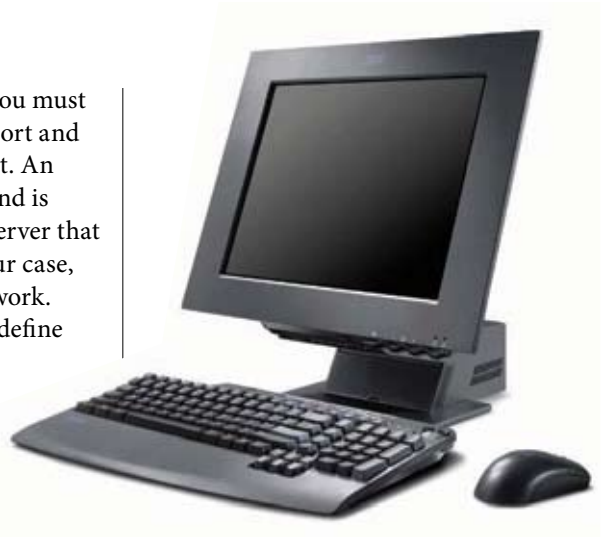


Figure 1 shows a typical view of your z/OS files on Windows.

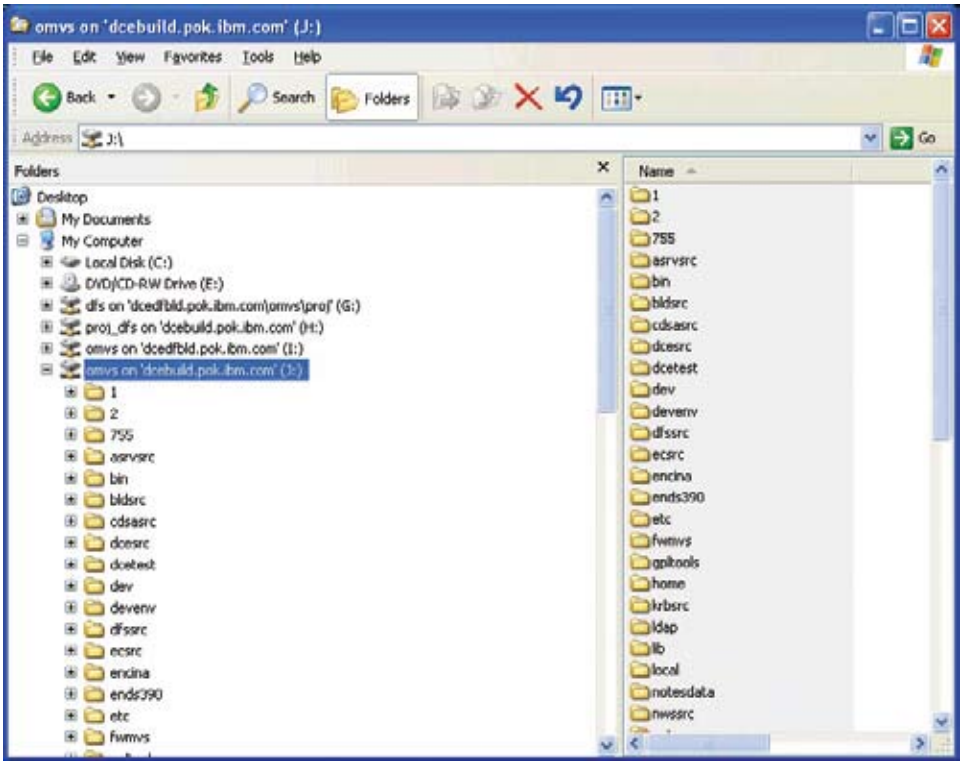


Figure 1. z/OS files on Windows

dfstab file

The **dfstab** file specifies file systems to export. For example:

```
/dev/ufs2    hfs2    ufs     101
0,,1715
```

Where:
- /dev/ufs2 is the device name of the file system to be exported
- hfs2 is the file system name
- ufs is the file system type (ufs used for all file systems in SMB)
- 101 is the file system ID
- 0,,1715 is the fileset ID.

devtab file

The **devtab** file stores the identifying information for all file systems to export. For example:

```
define _ ufs 2 omvs.user.abc text
```

Where:
- define_ufs 2 matches /dev/ufs2
- omvs/user.abc is the file system name
- text means that the data in the file system is text and should be converted

smbtab file

The **smbtab** file specifies the shared directories and printers to make available to SMB clients. For example:

```
/dev/ufs2  myshare    ufs  "My
share description"    r/w  100
/ghi
```

Where:
- /dev/ufs2 is the device name that matches dfstab
- myshare is the share name to which user will connect
- ufs is the file system type that matches dfstab
- "My share description" is the share description
- r/w are the permissions
- 100 is the maximum number of users connected to the share
- /ghi is the pathname that the share represents that user will connect to.

You can configure dynamic export through the dfstab, devtab, and smbtab entries, but you specify only the top file system as shared because any file systems mounted in the hierarchy below get exported dynamically.

When the setup is complete, use the **START, DFS** command to start the SMB server. You can also use the same files for a new release when the same IP address, RACF® user definitions, and exported file definitions apply to the target image for the new release.

Authentication

SMB checks the authorization of a user by matching the SMB user ID to the z/OS user ID. That means you'll need to set up the smbidmap file with the mapping of SMB user on the first line, the z/OS user ID on the second line, followed by blank line. There are also wildcard settings when the IDs are the same for SMB and z/OS:

```
smbusera
zosusera

smbuserb
zosuserb

*
=
```

Three methods exist for handling passwords: clear text password, encrypted passwords, and passthrough authentication. Let's take a look at each method.

Passthrough authentication

Passthrough authentication allows the SMB server to receive a logon request and forward the logon request to a Windows 2003 domain controller for authentication. That means the user ID and password must be the same as the Windows Domain user ID and password. If the domain controller successfully authenticates the user, the SMB server maps the SMB user ID to a local z/OS user ID using the smbidmap file.

To enable passthrough authentication, use the following environment variables in the /opt/dfslocal/home/dfskern/envar file:

- _IOE_SMB_CLEAR_PW=NOTALLOWED allows the SMB server to use clear or encrypted passwords.
- _IOE_SMB_AUTH_SERVER sets the IP address of the Primary Domain Controller to authenticate SMB users.
- _IOE_SMB_BACKUP_AUTH_SERVER sets the IP address of the Backup Domain Controller (if it exists).

- _IOE_SMB_AUTH_SERVER_COMPUTER_NAME sets the computer name of the Primary Domain Controller.
- _IOE_SMB_BACKUP_AUTH_SERVER_COMPUTER_NAME sets the computer name of the Backup Domain Controller (if it exists).
- _IOE_SMB_AUTH_DOMAIN_NAME sets the domain name of the Windows domain.

Clear passwords

Using clear passwords is the easiest authentication method. Clear passwords allow the SMB user to use the z/OS password when connecting. When the user maps a drive letter to a shared directory, that user ID goes to the SMB server. You'll need to set _IOE_SMB_CLEAR_PW to REQUIRED. The PC password should match the z/OS password or the user will have to enter a **NET USE** command to supply the z/OS password. With the SMB server, you can use mixed case passwords for the z/OS password.

Encrypted passwords

Using encrypted passwords is a little trickier. The SMB user must specify the password in a RACF DCE segment associated with their RACF profile, which means configuring _IOE_SMB_CLEAR_PW to NOTALLOWED.

The downside to this method is that the z/OS password needs to be coordinated with DCE password, so you will have to set up encrypted passwords and create DCE segment for each user. To do so, follow these steps:

1. Activate class KEYSMSTR SETROPTS CLASSACT(KEYSMSTR).
2. Activate class DCEUUIDS SETROPTS CLASSACT(DCEUUIDS).
3. Define entry DCE.PASSWORD.KEY in class KEYSMSTR being sure to supply a 16-position KEYMASKED value.
4. Enter the command RDEFINE KEYSMSTR DCE.PASSWORD.KEY SSIGNON(KEYMASKED(nnnnnnnnnnnnnnnn)).
5. Define a RACF DCE segment for each z/OS user ID that requires the SMB encrypted password capability using the ALTUSER USER_ID DCE command.

To allow SMB users to connect to the SMB server after the encrypted passwords are enabled, each user must enter the following command from the shell:

```
$ smbpw smb _ login _ password
smb _ login _ password
```

Where *smb_login_password* is the SMB user's SMB password (specified twice).

How to use SMB from Windows

To connect to any share from Windows, the user can either map a network drive or enter the **NET USE** command.

To map the drive, from Windows Explorer select Tools >Map Network drive and enter:

```
\\system.domain.mycompany\share.
```

To use the **NET USE** command including map drive or wildcard, system name, share name, user (or windows default used), and password, enter:

```
net use * \\system.domain.
mycompany\share  /user:smbuser
password
```

After connecting to the share, the mapped drive is the local drive just like any other drive on the PC.

How to use SMB from Linux

To connect to any share from Linux, the user needs to create a mount directory and mount the share, as follows:

```
>mount -t smbfs //system.
domain.company.com/share /root/
mount _ dir -o username=smbuser,
password=password
```

After connecting to the share, the mount directory is local on the Linux system.

Try it for yourself

We hope this article inspires you to try SMB. We think your users will thank you. For complete information, check out *z/OS Distributed File Service SMB Administration*, SC24-5918. ■

Trust me, partner

Open up with Encryption Facility's new OpenPGP support

BY SAHEEM GRANADOS

With increasing legal requirements and regulations for privacy and rising financial costs associated with the loss or compromise of sensitive data, data encryption has become critical. Consider the following kinds of encryption for your enterprise:

- Database row/segment level encryption
- Bulk data encryption for archival purposes
- Data set or z/OS UNIX Systems Services file encryption for trusted business partner exchange.

Many z/OS solutions currently exist. For example, IBM Data Encryption for IMS and DB2 is designed to provide DB2 row level or IMS segment level encryption. IBM's TS1120 encrypting tape drive, using IBM's Encryption Key Manager, can provide bulk data encryption on tape with exceptional performance. (See "Data-security worries? Drive them away with tape encryption!" on page 32.) The optional feature of Encryption Facility for z/OS, Encryption Services, is designed to encrypt data sets that can then be exchanged with trusted business partners.

OpenPGP addresses all of the challenges facing trusted business partner exchange…

Unique challenges to encryption

When you consider trusted business partner data exchange, keep in mind the following unique challenges:

- Exposure of data while in transit
- Need for confidentiality
- Need for non-repudiation
- Need for cross-platform access.

Encryption Services addresses most of these challenges. It relies on the System z proprietary format for data encryption, which provides the following advantages:

- High performance cryptography and compression using specialized z/OS hardware
- Native data set support that allows you to maintain data set format and record length attributes.

The System z format is best suited for environments where z/OS is the only platform on which the data will be accessed. Entitled licenses to a decrypt-only client and a Java reference implementation allow you to quickly adopt the System z format within an enterprise that requires trusted relationships with other z/OS businesses.

Encryption Facility V1.2 provides a new component that supports the OpenPGP Internet draft standard as described in IEFT RFC2440. While the System z format is still supported in V1.2, OpenPGP addresses all of the challenges facing trusted business partner exchange; furthermore, it offers additional benefits, by making available many implementations on other platforms that support OpenPGP.

What is OpenPGP?

OpenPGP is a draft Internet standard defined in RFC2440. It defines a number of services that an OpenPGP-compatible system needs to provide such as:

- Confidentiality through encryption
- Authentication/non-repudiation through digital signatures
- Data integrity through modification detection codes
- Key exchange using OpenPGP certificates
- Optimized data packaging through compression.

With many of these encryption services, you have a choice of algorithms to use. For example, you can choose either the ZIP or ZLIB algorithm for data compression. For digital signatures, you can use either the Rivest-Shamir-Adelman (RSA)/ Secure Hash Algorithm -1 (SHA1) or Digital Signature Algorithm (DSA). For information, see the Web site **ietf.org/rfc/rfc2440.txt**.

More about OpenPGP and System z Encryption

System z format and OpenPGP format both require the same fundamental steps for performing encryption. The difference between System z and OpenPGP lies in the format of the encrypted data.

To generate a symmetric session key required to encrypt the data, consider the following:

- The System z format supports Advanced Encryption Standard (AES) with 128 bit keys and Triple Data Encryption Standard (TDES) keys.
- The OpenPGP format supports TDES and AES with 128, 192, and 256 bit keys, and Blowfish. On an IBM System z9, AES 128 and TDES can leverage specialized z/OS cryptographic hardware for acceleration, while, on older systems, only TDES can be used for acceleration. Blowfish does not benefit from hardware acceleration.

To generate a required encrypted session key, two options are available:

- *Password/passphrase based encryption (PBE)* is the process of taking a password or passphrase and using it as the basis for the generation of a symmetric key. For the OpenPGP support, this calculated symmetric key is then used to encrypt a random session key. The System z format uses this key as the session key. The symmetric encryption algorithms can be used to encrypt the session key.
- *Public key based encryption* uses a business partner's public key to encrypt a random session key for both formats. To allow multiple business partners to access the same data, both the System z and OpenPGP formats can use multiple public keys to encrypt the random session key. The output file is created by writing out the encrypted session key (multiple times if there are multiple recipients), followed by the encrypted data to be exchanged. The System z format supports RSA for public key based encryption. The OpenPGP format supports RSA and ElGamal algorithms for public key-based encryption.

Establishing trust

Establishing trust is paramount when you exchange data. You establish trust by exchanging public keys, or through PBE by exchanging secret key material in the form of a passphrase. Exchanging *secret* key material instead of *public* key material is a critical decision for any business that employs cryptography. Any time secret key material is exchanged, the odds increase that the key might be compromised, which in turn can circumvent the infrastructure you use to maintain trust.

Public/private key cryptography relies on the concept that a public and private key are mathematically linked to each other and to no other counterpart. Anyone can access the public key part, but the private key part is only accessible by its owner who needs to take special care to keep the private key private. Data that is encrypted using a public key can only be decrypted by the corresponding private key. Similarly, digital signatures are calculated using the private key and can only be verified by the corresponding public key. Given this and the fact that the private key should never be exposed, public key exchange is usually a more secure method for establishing a trust relationship.

For an introduction to the Encryption Facility product and the use of private and public keys, see "Now, where did I put my keys…? IBM Encryption Facility for z/OS" in *z/OS Hot Topics Newsletter Issue 14*, February 2006, GA22-7501-10.

Digital certificates

You exchange public key material through digital certificates that establish an entity's ownership of a public key. When you use digital certificate exchange to establish trust, consider the following questions:

- How do you verify that the certificate has not been manipulated?
- How do you verify that the certificate is really the certificate of the intended partner?

OpenPGP makes use of two different digital certificate types: X.509 and OpenPGP. Both rely on digital signatures to resolve these issues.

X.509 certificates rely on a certificate authority (CA) to digitally sign the certificate contents after the CA has verified the owner of the certificate. All parties in the trust relationship must agree on the CA and in turn establish a trusted relationship with the CA.

Because of financial costs, organizations often need to generate new public/private key material and corresponding certificates that they can use for signing other digital certificates. These internal certificates are signed by a third-party CA, and then established as an internal CA. The internal CA is then used through public key infrastructure (PKI) to sign digital certificates for exchange with business partners. This approach of multiple CAs organized into a hierarchy is often referred to as a chain of trust.

When the CA trust relationship is established, the digital signature the CA calculates for a partner's digital certificate allows other partners to verify that the certificate has not been tampered with and that it, indeed, is the certificate of the intended partner.

OpenPGP certificates are defined by RFC2440, and differences exist between OpenPGP certificates and X.509. The most significant difference between OpenPGP and X.509 certificate types are in the area of trust management. In an X.509 certificate, the CA or CA chain of trust is ultimately responsible for the authenticity of an X.509 digital certificate. OpenPGP certificates rely on a Web of Trust model, which is a decentralized approach to certificate authentication. This decentralized approach often requires additional processes and procedures to authenticate the ownership of an OpenPGP certificate.

The System z format support in Encryption Facility relies on X.509 certificates for the exchange of public key material. The OpenPGP support can use either X.509 certificates or OpenPGP certificates.

z/OS key management

Because public/private key cryptography is the most secure approach to establishing a trust relationship, key management is of great importance. For both formats, Encryption Facility V1.2 relies on the z/OS established key management strengths of RACF and ICSF. You can use RACF and ICSF to manage the repositories for X.509 certificates and the public/private key material. Through RACF, ICSF, and PKI services, you can import, generate, sign, and verify digital certificates and public/private key material to use with Encryption Facility. In addition to these z/OS services, Encryption Facility for OpenPGP provides commands to generate, import, and export as OpenPGP certificates any X.509 certificates whose key material resides in ICSF. Thus, it provides a bridge to existing z/OS key and digital certificate management processes.

A matter of trust

Encryption Services allows trusted business partner exchange through both the System z and OpenPGP formats. Although the formats are not interchangeable, each relies on the z/OS centralized key-management services as the foundation for the trust that is crucial to secure business processing. For more information on the Encryption Facility for z/OS product including Encryption Facility for OpenPGP, visit the following Web site: **ibm.com/servers/eserver/zseries/zos/bkserv/zswpdf/encryption_facility12.html.** ■

z-Doku

M A I N F R a m E

R			I					M
	F	E		N	A	m	R	
			a					
N		a		A	M	R		I
F		m	E	I		A		N
					N			
	M	R	F	a		E	N	
E					m			A

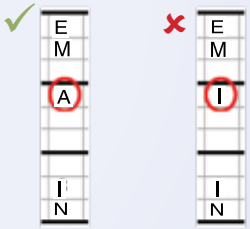
Solution on page 57

How to play z-Doku

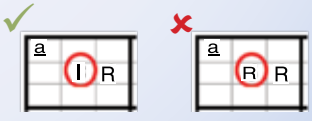
The object is to fill the other empty cells with the nine letters that spell out the word M A I N F R a m E (one unique letter only in each cell) according the following guidelines:

1. The letter can appear only once on each row:

2. The letter can appear only once on each column:



3. The letter can appear only once on each 3x3 region:



Data-security worries? Drive them away with tape encryption!

BY ERIKA DAWSON AND ARTHUR BARISKA

Are you losing sleep worrying that your confidential data could fall into the wrong hands? Do you wake from nightmares in which shadowy figures with nefarious intent carry off tapes of unsecured data? Do you lie awake wondering how to make your data more secure? If so, tape encryption could be your key to a good night's sleep.

How do you request encryption?

The IBM System Storage™ TS1120 Tape Drive (3592 Model E05) now supports outboard data encryption at the drive level using the 256-bit Advanced Encryption Standard (AES) algorithm. What does this mean for you? With host-based policy management driven through System Managed Storage (SMS), you can select which data to encrypt. You can specify a new recording technology format through data class that indicates whether or not your data is to be encrypted. Rest assured, regardless of where the drives are located and how they are managed, the same enablement solution works with or without system-managed tape. Figure 1 shows how to request encryption through data class by simply specifying EE2 (short for Enterprise Encrypted Format 2 or EEFMT2).

The first file sequence that the drive writes to a tape determines whether or not all data written to that tape will be encrypted. If the assigned data class indicates encryption, all data written to that tape will be encrypted. A mix of encrypted and non-encrypted data on the same tape cartridge is not supported.

How is data encrypted?

Now that you know how to request encryption, let's look at how the data is encrypted.

The tape drive encrypts data on each tape cartridge using a data key. All data on the same tape cartridge will then be encrypted using the same data key. Keys are managed by the IBM Encryption Key Manager (EKM) component for the Java platform. When the tape drive needs a data key (to encrypt or decrypt data), there is a communication flow between the drive and the EKM to acquire the necessary key.

This communication flow uses either in-band or out-of-band key management:

- Using in-band key management, a request to the EKM flows through ESCON® or FICON® channels to a new proxy interface in the Input/Output Supervisor (IOS) component of z/OS. From this proxy, the request flows over TCP/IP to the EKM.
- Using out-of-band key management, communication with the EKM is handled exclusively by the tape control unit. This style of key management is employed by System z hosts that support encryption, but do not support a proxy interface to the EKM.

For security, EKM encrypts (or wraps) the data key using one of the following keystores that z/OS supports:

- JCEKS
- JCERACFKS
- JCE4758KS (JCECCAKS)
- JCE4758RACFKS (JCECCARACFKS).

This encrypted data key is stored in several places on the tape cartridge in non-user areas of the tape in a structure called the Externally Encrypted Data Key (EEDK).

The JCERACFKS, JCE4758KS (JCECCAKS), and JCE4758RACFKS (JCECCARACFKS) keystores use additional software products such as the IBM Integrated Cryptographic Service Facility (ICSF) and the IBM Resource Access Control Facility (RACF) to manage the encryption keys. These products work in conjunction with the EKM. In addition, the JCE4758KS (JCECCAKS) and JCE4758RACFKS (JCECCARACFKS) keystores take advantage of the hardware cryptography provided by z/OS ICSF to improve the security characteristics of the data encryption key generated by the EKM.

How do you manage keys?

The data key is encrypted using the public key of a Rivest-Shamir-Adleman (RSA) public/private key pair. When you request that your data is to be encrypted, you can also specify up to two key labels. Think of a key label (also called an alias) as a user-friendly reference to a public/private key pair. The key labels can be specified through data class or through the DD statement (JCL, dynamic allocation, or TSO allocate). You can also set key label defaults at the EKM. These defaults will be used if you request encryption but do not specify key labels at the host. Think of the keystore as the home for your key labels and corresponding public/private keys.

```
Media Interchange
 Media Type  . . . . . _ _   (1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or blank)
 Recording Technology  . . . _ _ _  (18, 36, 128, 256, 384, E1, E2, EE2 or blank)
 Performance Scaling . . . . _    (Y, N or blank)
 Performance Segmentation  . _    (Y, N or blank)
```

Figure 1. Requesting encryption through data class

The ideal part of this solution is the ability to have centralized key management. Because the encryption key manager is supported across TCP/IP, there are many possible locations for your primary and secondary key manager. The key manager can reside on the same z/OS host that is writing and reading your data, on a different z/OS host, or even on a different platform. With this solution, z/OS can also be the centralized home of your key manager for both z/OS and Open Systems data.

How do you know it worked?

So you've used data class to request that your data be encrypted. How do you know that it worked, and your data was encrypted? The IEC205I message issued during tape data set close processing indicates whether the data was encrypted, and what key labels were used. Figure 2 shows a sample IEC205I message with the new encryption-related information.

```
IEC205I SYSUT2,ATNCMP1,STEP1,FILESEQ=1, COMPLETE VOLUME LIST,
 DSN=ATL.TESTJOB.EE2,VOLS=J11986,
 LISTED VOL(S) HAVE BEEN DATA ENCRYPTED,KL1CD:L,KL2CD:H,
 KL1=dfsmskeylabel1,KL2=dfsmskeylabel2,TOTALBLOCKS=10
```

Figure 2. Sample IEC205I message with encryption-related information

Looking at Figure 2, you might notice that there is additional information included that we have not discussed. For example, why do we allow two key labels to be specified? KL1CD and KL2CD refer to an encoding mechanism for the specified key label. The key labels can be stored using an encoding mechanism of "label", or they can be stored as a "hash" of the public key referenced by the key label.

Storing the key label as a hash value in the EEDK structure on the tape is useful for business partner data exchanges where you can use a different label to reference the same public/private key pair. One key label can be used for local (on-site) usage, and the other key label can be used for off-site or business-partner exchanges where you might only have your business partner's public key. To read the data, the data key only needs to be decrypted with one of the two private keys. This lets you write and read the tape data at your own location, and also enables your business partner to decrypt the data. Figure 3 shows how the key labels and their encoding mechanism can be specified through data class.

```
Encryption Management

            Key Label 1 . . .     (1 to 64 characters or blank)
            ________________________________________________

            Key Label 2 . . .
            ________________________________________________

    Encoding for Key Label 1  . . . . . _         (L, H or blank)
    Encoding for Key Label 2  . . . . . _         (L, H or blank)
```

Figure 3. Specifying key labels through data class

In addition to the IEC205I message that tells you your data is encrypted, with system-managed tape support, the recording format (EEFMT2) is also recorded in the Tape Configuration Database (TCDB). If you are using DFSMS removable media manager (DFSMSrmm) as your tape management system, RMM also tracks the recorded format (EEFMT2) in addition to the key labels.

During CLOSE processing, key labels are stored in an extended information segment (subtype 7) of the SMF 14/15 record.

What if there is a problem?

If all goes well, you established the appropriate policies at the host and were able to verify that your data was encrypted. But what if a problem occurs? Enhancements to the IOS000I message can help. Figure 4 shows the EKM reporting a failure to the host.

However, what if the problem points to a communication failure with the EKM? With in-band key management, you can use the MVS DISPLAY IOS,EKM command to determine which key managers were configured for use. Figure 5 shows sample display output.

You can also use the MVS DISPLAY IOS,EKM command with the VERIFY option to determine whether the IOS proxy can communicate with the EKM. Figure 6 shows sample output with the IOS proxy successfully communicating with the EKM.

```
IOS000I 0BD0,60,IOE,01,0E00,,**,JJC046,ATNCMP1

804C08C022402751 0001FF0000000000 0005EE3100000092
2004E82061BA2111
ENCRYPTION FAILURE
CU=00 DRIVE=000000 EKM=05EE31
```

Figure 4. Sample IOS000I message indicating encryption failure

```
DISPLAY IOS,EKM
IOS099I 11.19.25 EKM HOSTS
PRIMARY   HOSTNAME=192.0.2.0:3801
SECONDARY HOSTNAME=192.0.2.1:3801
```

Figure 5. Sample IOS,EKM display

```
DISPLAY IOS,EKM,VERIFY=ALL
IOS099I 18.52.53 EKM HOSTS
PRIMARY   HOSTNAME=192.0.2.0:3801
SECONDARY HOSTNAME=192.0.2.1:3801
MAX CONNECTIONS = 255, PERMANENT CONNECTIONS = 008
IOS631I PRIMARY ENCRYPTION KEY MANAGER WAS SUCCESSFULLY CONNECTED
IOS631I SECONDARY ENCRYPTION KEY MANAGER WAS SUCCESSFULLY CONNECTED
```

Figure 6. Sample IOS,EKM display with VERIFY

How do you put it all together?

With z/OS, the use of the in-band proxy in IOS allows key-management information to be exchanged over ESCON/FICON, instead of requiring the deployment of a secondary IP network. The reliability and physical security of the existing I/O attachments is one reason that you might choose to use the in-band key management path to the EKM. The primary key manager can reside on the system that is writing and reading to tape, and its secondary key manager can reside on another z/OS system. The primary and secondary key managers can also be established using Virtual IP Addressing (VIPA). That being said, you have many options for establishing where your primary and secondary key managers reside.

Figure 7 shows various possibilities.

Where else do you look?

Now that you are familiar with the tape encryption solution, you can read all about it:

- *z/OS DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592)*, SC26-7514
- *IBM System Storage TS1120 Tape Encryption Planning, Implementation, and Usage Guide*, SG24-7320
- *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418.

Also see enabling PTF UA90286 for z/OS V1R8. For z/OS V1R6 and V1R7, see enabling PTF UA90284 and UA90285. ■

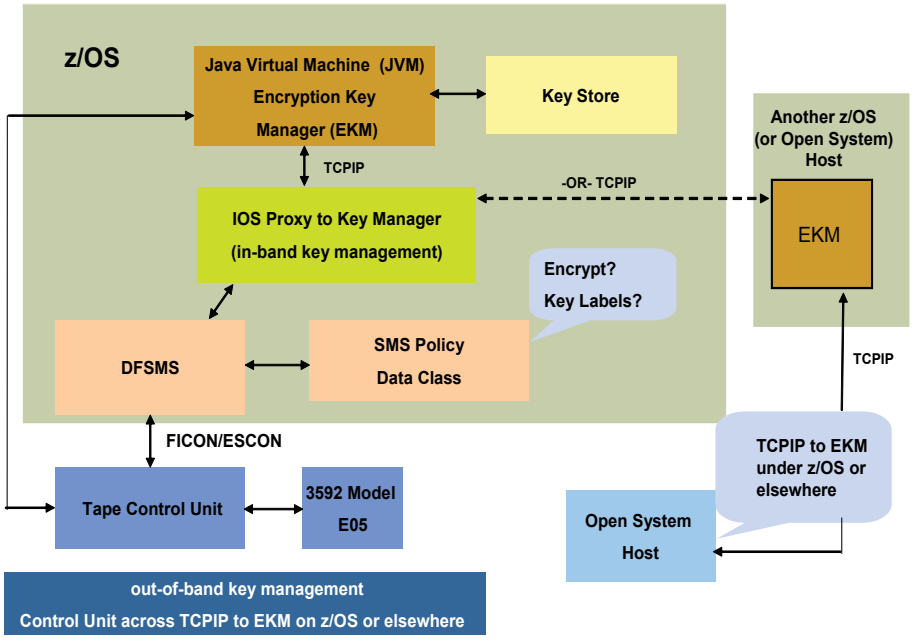


Figure 7. Putting all the pieces together

HOW TO use role-based security with z/OS

BY MARK NELSON, RANDY LOVE, AND ADRIAN LOBO

Among its many new features, DB2 Version 9.1 for z/OS introduces a new way to authorize users: the role. With role-based authorization, along with the new DB2 V9 trusted context, you can control access to DB2 objects based not only on the identity of the user or server making the request, but also on the characteristics of the connection from the DB2 client (on any platform) to DB2 on z/OS. The combination of roles and trusted contexts provides a flexible way to manage access control so that you can more narrowly control access. For example, you can associate a user with a role in a trust context so that the user can perform certain tasks. The user's authority to perform those tasks exists only when the task is performed under the auspices of the trusted context.

In this article, we describe these new constructs and how to use them to better define and manage your access controls.

The role

Let's start with the linchpin of the new authorization mechanism, the role. A role is a long DB2 identifier, which means that it is a mixed case value, with blanks allowed, with a length of 1 to 128 characters. Not everyone can create roles, of course. To create a role, you must have SYSADM or SYSCTRL authority. You create a role using a simple SQL statement:

```
CREATE ROLE TELLER;
```

Creating the role is only the first step. The next step is to define the rules that associate the role with requests or connections as they come to DB2. This association is made with the trusted context.

The trusted network context

Trusted context is a role that is associated with the connection. Within the trusted context definition, you define the selection criteria, which determines whether the connection is trusted and what role is associated with the connection. The criteria that you specify can include:

- Identity of user making the request
- Job name
- The IP address from which the request originates
- Domain name from which the request originates
- SERVAUTH name of the connection
- Level of encryption that is associated with the connection.

Creating a trusted context is just a little more involved than creating a role. Let's go through a few examples to show the flexibility of the trusted context. How would you create a trusted context named CTX_01 to assign the role 'MAINTENANCE' to a batch job named MAINT, which is running under user ID OPERMNT? Here's how:

```
CREATE TRUSTED CONTEXT CTX_
01
BASED UPON CONNECTION USING
SYSTEM AUTHID OPERMNT
ATTRIBUTES (JOBNAME 'MAINT')
DEFAULT ROLE MAINTENANCE;
```

A connection that comes from the same z/OS system as the one on which the DB2 subsystem is running is called a *local connection*. A connection that originates on a different platform or a different z/OS system is called a *remote connection*.

The attributes of the connection can be used in the selection criteria for the context. Let's say that you wanted to create context CTX_02. The context assigns the role TELLER to a connection when all of the following is true:

- The connection originates from a server connecting with the authorization ID SRVRID1
- The connection uses the domain name `www.serverdomainname.com`
- The connection uses SSL encryption.

The SQL statement to create this context looks like the following example:

```
CREATE TRUSTED CONTEXT CTX_
02
BASED UPON CONNECTION USING
SYSTEM AUTHID SRVRID1
ATTRIBUTES (ADDRESS 'www.
serverdomainname.com',

ENCRYPTION 'HIGH'
                          )
DEFAULT ROLE TELLER;
```

This is the type of trusted context that you'll see with middleware servers, such as IBM WebSphere. These servers connect using a user ID that is the identity of the server itself. If the server supports the propagation of the end-user identity, you can use the trusted context definition to assign different roles to the end users.

Let's update the preceding example to assign the default role of TELLER to all connections with the exception of those that come from the end users FRED and WILMA. These two user IDs will get the role 'TELLER SUPERVISOR'. The trusted context definition that accomplishes this is in the shaded box that follows.

```
CREATE TRUSTED CONTEXT CTX_02
BASED UPON CONNECTION USING SYSTEM AUTHID SRVRID2
ATTRIBUTES (ADDRESS 'www.serverdomainname.com',
                    ENCRYPTION 'HIGH')
           DEFAULT ROLE TELLER
           WITH USE  FOR PUBLIC,
           FRED  ROLE "TELLER SUPERVISOR",
           WILMA ROLE "TELLER SUPERVISOR";
```

Granting access using the role

Getting a role assigned to a request is the first step. The next step is to use that role to grant access to DB2 objects.

When using native DB2 access controls, you do this with the SQL GRANT statement. For example, to grant SELECT on the table ACCTMSTR.ACCOUNT_BALANCE to requests that are running with the role TELLER, use a GRANT statement like this:

```
GRANT SELECT ON TABLE
ACCTMSTR.ACCOUNT_BALANCE TO
ROLE TELLER;
```

If you are using the RACF Access Control Module (DSNXRXAC), the new RACF WHEN(CRITERIA(…)) is used to allow a role access to a DB2 object. To give the role TELLER SELECT authority on this table, use this RACF PERMIT command:

```
PERMIT DSND.ACCTMSTR.
ACCOUNT_BALANCE.SELECT
CLASS(MDSNTB)
 WHEN(CRITERIA(SQLROLE(TELLER)))
 ID(*)
```

To give the 'TELLER SUPERVISOR' role UPDATE access to this table, use this RACF PERMIT command:

```
PERMIT DSND.ACCTMSTR.ACCOUNT_
BALANCE.UPDATE CLASS(MDSNTB)
WHEN(CRITERIA(SQLROLE('TELLER
SUPERVISOR'))) ID(*)
```

The SQLROLE value is in quotation marks because 'TELLER SUPERVISOR' contains a blank. RACF does not fold the SQLROLE value to upper case automatically. Because the role name can contain lower case characters, thus making "TELLER", "Teller", and "teller" three different roles, be sure to enter the entire SQLROLE value in precisely the case that you want.

A word about ownership

DB2 grants certain privileges called the "implicit privileges of ownership" to the owner of an object. Prior to DB2 Version 9.1 for z/OS, owners of objects were always authorization IDs. Because authorization IDs represent an ndividual's authority, if the individual's function within the organization changed, the authorization ID would still retain all of its ownership privileges.

DB2 Version 9.1 for z/OS solves this dilemma by allowing roles to own objects. Assigning the role as owner to an object is done by adding the "ROLE AS OBJECT OWNER" clause to the CREATE CONTEXT statement as follows:

```
CREATE TRUSTED CONTEXT CTX_03
BASED UPON CONNECTION USING
SYSTEM AUTHID OPERMNT2
ATTRIBUTES (JOBNAME 'MAINT')
DEFAULT ROLE MAINTENANCE
WITH ROLE AS OBJECT OWNER;
```

The value of the role and the trusted context

In this article, we've only scratched the surface of roles and trusted contexts. As you can see, the combination of trusted contexts and roles can simplify your security architecture and administration. It allows for tighter access control by controlling access based not only on who is making the request but also on how the request is being made. And, you can assign ownership based on a functional basis instead of based on an identity.

For more information, about roles, trusted contexts, and the RACF support for DB2 Version 9, check out:

- *DB2 Version 9.1 for z/OS Administration Guide*, SC18-9840
- *DB2 Version 9.1 for z/OS Access Control Module Guide*, SC18-9852
- *Securing DB2 and Implementing MLS on z/OS*, SG24-6480
- *z/OS V1R8.0 Security Server RACF Security Administrator's Guide*, SA22-7683-10
- *z/OS V1R8.0 Security Server RACF Command Language Reference*, SA22-7687-10. ■

Pass it on without a trace

IBM Session Manager and RACF PassTickets

BY RICH GUSKI AND DALE TANNER

Wouldn't it be nice to enter your password once and have it automatically log on to the many applications running on the z/OS systems? Wouldn't it be even better if the RACF password wasn't exposed while traveling across the network? When an IBM Session Manager customer asked IBM if this was possible, the development team took the challenge and gave them a secure and practical solution.

What is IBM Session Manager for z/OS?

IBM Session Manager is a z/OS product for VTAM and TCP/IP that provides a secure and user-friendly single sign-on interface. After a user signs on using Session Manager, it provides a menu to continue signing on to various applications with the ability to move from one active session to another. You can write scripts that automatically perform sign-on, sign-off, and transactions. As part of the initial sign-on, IBM Session Manager captures the user ID and password and uses them in the scripts to automatically log on to other applications. Auto-logons save time because only one sign-on is needed. All subsequent logons for later application sessions are automatic when the user selects them from the menu.

The only caveat: using Session Manager meant that each user's RACF password had to be captured in the clear (not encrypted), saved for later use, and then passed around the network during subsequent sign-ons.

PassTicket support

IBM Session Manager addressed the customer's security issue by supporting RACF PassTickets. RACF PassTickets are short-lived, system-generated-and-verified password substitutes that replace RACF passwords. Because they last only minutes, and by default are used only once, they can be much more secure than using the RACF password within a logon script. For example, if someone were to intercept a PassTicket, it would very likely be expired and useless.

How Session Manager uses PassTickets

Session Manager uses an exit routine and scripts that specify a user's RACF user ID and an application name to log on to the target application. Session Manager calls RACF to generate a PassTicket. RACF generates the PassTicket based on the user ID, the application name, the current time (within one second), and a secret key given to RACF for the application by the system security administrator. Because the PassTicket generation and later evaluation processes use cryptography, the resulting PassTickets are highly random and unpredictable. Session Manager then passes that PassTicket in place of the user's RACF password during the application logon. The application that receives the logon passes the user ID and PassTicket to RACF on the system where it is running. On this system, RACF evaluates the PassTicket just as it would have validated the RACF password. If the PassTicket evaluation is successful, the user signs on just as if he or she were using the RACF password.

Make your boss happy

Running IBM Session Manager with auto-logon scripts that implement support for PassTickets can help you improve productivity at your installation. Won't your users be happy when you enable them to sign on once, without having to enter their user ID and password over and over again? In addition, you improve security by greatly reducing password exposure, so your boss will be happy too.

IBM Session Manager V1R3 or later includes a sample exit routine and several scripts that you can use to implement PassTicket sign-ons on an application-by-application basis. For more information, see **ibm.com/software/htp/cics/smanag/**. ■

Why did the security administrator cross the road?

Remote authorization and auditing, it's no joke!

BY MICHAEL ONGHENA, SCOTT WOOLLEY, AND ROSS COOPER

A chicken, an elephant, and a security administrator go into a bar. The bartender threatens to make the elephant and administrator leave because of their disorderly conduct the night before. The two plead to stay at their favorite hangout, and the bartender agrees to let them stay on probationary status. "Fine, if you promise to stay seated quietly at the far table you can remain. I'll trust your designated chicken to monitor your behavior and meter your alcohol consumption."

The bartender sends a pitcher of beer to the table for the chicken to pour into each friend's mug. Before each refill, the bird flaps over to the bartender, and asks for authorization to dispense the next round. Each time the elephant leers across the room or the administrator makes a rude comment, the chicken reports back to the bartender, who audits each infraction. And so it continues into the evening.

To get to the other side

Just as the bartender relegated dispensing and supervision duties to the chicken, security responsibilities are distributed across multiple platforms in computer enterprises. Customers increasingly look for ways to centralize security management where possible. A new service is available to leverage strengths of z/OS V1R8 security to handle authorization-checking and event-auditing functions across distributed systems.

The remote authorization service allows applications distributed across the enterprise to remotely query a z/OS system to check a user's authority to access a resource. The resource might be something the application accesses on behalf of a client, or it might be a surrogate for a resource defined to the local system. Access to the resource can be centrally managed by the z/OS security administrator.

The remote audit service allows distributed applications to write security-log entries to a z/OS server as conventional Systems Management Facility (SMF) records, creating a central enterprise-wide audit record repository. Those records can be reformatted by the Resource Access Control Facility (RACF) SMF Unload utility, and ultimately processed into security-audit reports.

To screw in a light bulb

So you don't have a chicken on hand? You will need to following products and their prerequisites to use the new services:

- z/OS V1R8 IBM Tivoli Directory Server (ITDS)
- z/OS V1R8 Enterprise Identity Mapping (EIM)
- z/OS V1R8 Security Server (RACF), or an equivalent security manager with RACROUTE and R_auditx support.

The new services are part of the ICTX component shipped with EIM. Although there is no need to use or configure the EIM application programming interfaces to access the services, you must configure the ICTX component to work with the z/OS ITDS implementation of the Lightweight Directory Access Protocol (LDAP). The new services take the form of LDAP extended operation requests between client applications and the ITDS. The ITDS does not need to be configured for TDBM or any directory (other than the ICTX extended operation component) in order to use the new services.

The remote services are documented in *Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference*, SA22-7875.

You take away his credit card

Applications that use the remote authorization service must perform a number of operations in order to execute a remote authorization query:

1. Create a DER-encoded input parameter list that contains the data for the authorization query (see the Abstract Syntax Notation One and Distinguished Encoding Rules sidebar on page 40).
2. Bind to IBM Tivoli Directory Server.
3. Send an extended operation request to ITDS, specifying the remote authorization object identifier (OID), and the DER-encoded parameter list.
4. Read the DER-encoded ITDS response to determine if the authorization was successful.

Input to the remote authorization is described using the ASN.1 notation shown in Figure 1:

```
RequestValue ::= SEQUENCE  {
  RequestVersion INTEGER,
  ItemList       SEQUENCE of
     Item           SEQUENCE {
        ItemVersion    INTEGER,
        ItemTag        INTEGER,
        User           IA5String,
        Resource       IA5String,
        Class          IA5String,
        Access         INTEGER,
        LogString      IA5String
     }
}
```

Figure 1. Remote authorization request syntax

A quick glance at the parameter list shows familiar elements of a System Authorization Facility (SAF) authorization check: `User`, `Resource`, `Class`, and requested `Access`. `User` is a user ID that is defined to RACF (or another security manager). `Resource` and `Class` describe a known SAF general-resource protection profile. The requested `Access` is an integer whose value represents one of the possible access levels: Read, Update, Control or Alter.

The `LogString` is a place where the caller can specify up to 200 bytes of additional information. If the application does a mapping from a native or a distributed user ID to a SAF user ID, this is a great place to store information about that original native ID to maintain an audit trail that goes all the way back to the originating user.

Footprints in the butter

As is the case with the remote authorization service, the remote audit service also requires a client-side application to generate a DER-encoded input parameter list and send it to ITDS.

The input parameters to the remote audit can be described using ASN.1 notation shown in Figure 2.

The fields in the parameter list correspond to parameters in the SAF R_auditx callable service function. This service creates a SMF Type 83 audit record. We see standard audit parameters: `Violation`, `Class`, and `Resource` here, along with audit information that is common to z/OS, `Event` and `Qualifier`. The `DataFieldList` contains a number of `DataField` structures that are simple `Type/Value` pairs, also known as SMF relocate values. Relocate values have specific meanings, such as the identity of the user accessing the `Resource`.

What do you get when you … ?

The response from ITDS is the same for both the remote authorization and audit services. A severe ITDS error results in a standard error code and message.

```
RequestValue ::= SEQUENCE  {
   RequestVersion     INTEGER,
   ItemList           SEQUENCE of
      Item               SEQUENCE {
         ItemVersion        INTEGER,
         ItemTag            INTEGER
         LinkValue          OCTET STRING SIZE(8),
         Violation          BOOLEAN,
         Event              INTEGER,
         Qualifier          INTEGER,
         Class              IA5String,
         Resource           IA5String,
         Logstring          IA5String,
         DataFieldList      SEQUENCE of
            DataField          SEQUENCE {
               Type               INTEGER,
               Value              IA5String
            }
      }
}
```

Figure 2. Remote audit request syntax

However, a standard response from the remote services is another DER-encoded block of data. This response data is described using ASN.1 notation.

The important value in the response data is the `ResponseCode`. The `ResponseCode` indicates what really happened. See Figure 3, Table 1, and Table 2.

```
ResponseValue ::= SEQUENCE {
   ResponseVersion   INTEGER,
   ResponseCode      INTEGER,
   ItemList          SEQUENCE of
      Item              SEQUENCE {
         ItemVersion       INTEGER,
         ItemTag           INTEGER,
         MajorCode         INTEGER,
         MinorCode1        INTEGER,
         MinorCode2        INTEGER,
         MinorCode3        INTEGER
      }
}
```

Figure 3. Remote authorization and audit response syntax

Response Code	Explanation
0	Authorized. The specified user has the requested access to the specified resource.
4	Unknown. SAF does not know if the user has access to the resource, possibly because the resource is not defined to SAF.
8	Unauthorized. The specified user does not have the requested access to the specified resource.

Table 1. Common remote authorization response codes

Code	Reason
0	Success
3	Logging not required for this record based on currently set audit controls

Table 2. Common remote audit response codes

Most other response codes reflect errors in the parameter list or serious environmental errors. The `MajorCode` and `MinorCode` values provide

additional diagnostic information. The list of `ResponseCode`, `MajorCode` and `MinorCode` values can be found in *Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference*, SA22-7875.

Orange you glad I didn't say banana?

If you look back at the ASN.1 representation of the authorization and audit parameter lists, you can see that the authorization and audit data are encapsulated in a `SEQUENCE` (an ASN.1 structured type) named `Item`. Furthermore, an `ItemList` encapsulates a `Sequence of Item` structures. This enables the caller to perform an operation on each `Item` in the `ItemList` using a single LDAP request. Because the same operation will be performed on each `Item` in the `ItemList`, all of the `Item` structures must be of the same type. In other words, audit and authorization `Item` structures cannot be mixed in the `ItemList`.

The `ResponseValue` from LDAP has the same nesting of `Item` structures within an `ItemList`. Each has its own set of `MajorCode` and `MinorCode` values in the response, while the `ResponseCode` reflects the highest `MajorCode` among all the `Item` structures in the `ItemList`.

The ability to send multiple request `Item` structures to ITDS at once can significantly improve the efficiency of applications that use the remote authorization and audit services. This capability fits especially well with the remote audit service because an application can queue up audit records asynchronously and send them in bulk.

That's not a lion, that's a giraffe

So now you've seen how distributed applications can use the remote authorization and audit services to leverage the strengths of z/OS V1R8 security. You've seen how a distributed application can use the remote authorization service to remotely query a z/OS system to determine if a user can access a particular resource. You've seen how distributed applications can use the remote audit service to write SMF records to a z/OS server. For each of these services, you've seen how applications create DER-encoded input parameter lists which they send to ITDS. You've also seen how, to improve efficiency, an application can send multiple requests to ITDS at once. Most importantly, however, you were introduced to an elephant and a security administrator who, under the watchful eye of their designated chicken, are on probation at their favorite bar...

As the night winds down and the bartender yells out "LAST CALL!", the chicken flaps back to the bartender to make his final report. "The elephant is behaving rudely again." Hearing this, the elephant jumps up, grabs the chicken, and starts crushing it in his giant elephant hands. The bartender pulls a shotgun out from behind the bar, and unloads both barrels into the elephant's backside.

The moral of the story? A bird in the hand is worth two in the tush. ■

A new service is available to leverage strengths of z/OS V1R8 security to handle authorization-checking and event-auditing functions across distributed systems.

Abstract Syntax Notation One and Distinguished Encoding Rules

Two friendly applications want to exchange messages. Before they can exchange information, they first need to agree on content, structure, and encoding. Abstract Syntax Notation One (ASN.1) provides a flexible, unambiguous, and standard language to describe the content and structure of data. The Distinguished Encoding Rules (DER) define how the data described by ASN.1 should be encoded. ASN.1 tells the receiving application what kind of data to expect, and DER tells it exactly how to read the data.

```
MyData::=
  Name      UTF8STRING,
  Age       INTEGER
```

DER represents the data as a triplet of type, length, and value. A simple example illustrates the DER encoding of an integer, followed by a string of UTF-8 characters.

The preceding ASN.1 encoding states that the receiver can expect a name (in UTF-8 format) followed by an age (in INTEGER format). To encode Name 'Ian' and Age '7', the resulting DER encoding is specified as follows.

UTF-8	Length	Value			INTEGER	Length	Value
		'I'	'n'	'a'			
0C	03	49	61	6E	02	01	07

Table 3. DER representation in bytes (all values in hexadecimal)

HOW TO set up TKE for disaster recovery

BY JESSICA P. BONNER AND MICHAEL J. JORDAN

Because more people are using Trusted Key Entry (TKE) to help secure master key administration and to fulfill industry and government compliance requirements, it's critical that you understand the setup of TKE and how to load master keys at the disaster recovery (DR) site. This article looks at ways to manage your master keys at your DR site and covers important topics like planning, preparation, failover, failover testing, and cleanup.

Planning

The following environmental and procedural considerations can impact the TKE administration for both the primary and DR sites:

- TKE version
- Cryptographic card environment
- Master key media
- Whether the DR site is in-house or outsourced
- Staffing

TKE versions and smart card support

Ideally, the TKE version at the DR site should be the same as the primary site. If your installation requires the use of smart cards for secure master-key storage, verify that the TKE versions in both the primary and DR site are at levels that support smart cards. Otherwise, you can store the master keys in binary files that are useable with any TKE version. Smart card support for TKE is available on TKE Versions 4.2 through 5.1.

Host crypto-card environment

Differences in the host crypto-card environments for the primary and DR sites require different TKE administration procedures. For example, if the primary site has servers without the Cryptographic Coprocessor Feature (CCF) (that is, as part of z890/z990 or z9 processors) and the DR site has CCF servers (on the z800/z900 processor), the TKE procedures for the DR site must include steps for administering both the PCI Cryptographic Coprocessor (PCICC) and the CCF. The CCF has only16 predefined authorities, no role-based access checking, and a different set of master key registers from the PCICC. These differences affect the administration procedures associated with the following tasks:

- Defining authorities and roles
- Granting access
- Loading master keys

Important tip: If you plan to have a CCF system and a z890/z990 or z9 processor use your cryptographic key data set (CKDS), ensure that the CKDS set is initialized on the CCF system.

Master key media

If you plan to use smart cards, ensure that the CA smart card and the complete set of TKE smart cards are available at the DR site either as original cards or backup copies. The TKE cards must contain all signature authorities and master key parts. If you plan to use binary files, the binary files also need to contain all signature authorities and master key parts and must be available at the DR site (either on a diskette or DVD).

DR site: In-house or outsourced?

Although differences exist between an in-house or outsourced DR site, a primary consideration for TKE is cleanup. If the DR site is outsourced, ensure that sensitive production data is removed from both the host server and the TKE workstation.

Staffing

Your administrative process to set up TKE and load your master key parts dictates your staffing needs at the DR site. Ensure that the resources are available, or make contingency plans to account for shortfalls.

Preparation

Before you are ready for a failover or failover testing, be sure to establish DR site processes and procedures to gather a minimum set of data from both the host server and the TKE workstation. The processes, procedures, and information must be available at your DR site.

If there are differences between your primary and DR site, update your DR site processes and procedures to accommodate these changes. For example, if the primary site has a non-CCF server and the DR site has a CCF server, create a mapping of primary site authorities to DR site authorities.

The host data that you need includes a copy of each of the following items:

- CKDS
- Private key data set (PKDS)
- TKDS (token data set, new with z/OS V1R9).

The TKE workstation data must include:

- Signature keys required to create authorities and roles
- Access control point settings
- Master key parts.

The mechanism for collecting this data depends on whether the data is stored in smart cards or binary files.

Failover and failover testing

When you complete your TKE DR plan, you can set up your DR site. If your DR site is dedicated, much of the setup can be done in advance. However, for this discussion, assume the setup is done when you arrive at your DR site for failover or failover testing.

Before you can load the master keys, ensure that the TKE workstation can communicate with the host crytographic cards:

1. Update TKE and TKE 3270 session connectivity to match the DR site IP addressing.
2. Start ICSF on target z/OS systems.
3. Define and activate the TKE host program on z/OS. You can use the TKE host program from the primary site as long as the port number has been updated for DR site setup and the TKECM data set name has been changed to a new name.

The CKDS and PKDS in the DR site are unusable until you enter the master keys.

When you establish communication between the TKE workstation and the host crytographic cards, you can start to set up the TKE workstation and load the master key. Depending on your environment, follow these steps:

1. Ensure that you have copied the files from the diskettes or DVDs at your primary site to the data directories on the TKE workstation. You can use the file management utility to copy files.
2. Ensure that you have initialized the crytographic adapter in the TKE workstation to load the default roles and profiles.
3. For smart card users, use the smart-card utility program (SCUP) with the backup CA smart card created at the primary site to enroll the TKE crytographic adapter in your zone. This step is required to load master key parts from primary site smart cards copies.
4. Use Cryptographic Node Management Utility to configure the cryptographic adapter in the workstation. For smart card users only, use the TKE crypto-adapter logon keys from the smart cards copied from your primary site to create your profiles.
5. In the TKE application create or change authorities. Whether you need to change or create authorities depends on the server type. If you are using a CCF server, the authorities are predefined; to load your authority keys, you must change these authorities. If you are using a non-CCF server, you must create your authorities to load the authority keys.
6. Set access control points (ACP) using the values recorded from the primary site. The CCF does not have any ACP, and the ACP for the PCICC is a subset of the ACP for the PCIXCC/CEX2C.
7. Load your master keys. From TKE, load all of the master key parts. With CCF, there are three master keys:
   - The new master key, which is equivalent to a symmetric master key
   - A signature master key
   - A key-management master key.

   The signature master key and the key-management master key should be loaded with the asymmetric master key.
8. Using the TSO panels, set the New Master Key/Symmetric Master Key to change the Crypto Module status to active.

Cleanup

Before you leave the DR site, perform any cleanup. After a failover or failover testing, the host crytographic cards for the DR site and TKE workstation contain sensitive production data.

Perform the following cleanup tasks for the host crytographic cards and host data:

1. Initialize all affected domains with the TKE zeroize function (not to be confused with HMC zeroize function) to delete the master keys, retained keys and to reset the ACP.
2. Delete authorities by first loading a default authority, and then mapping it to a role through the ACP access control issue and access control cosign. Finally, from the authorities page, delete each authority that you created as part of failover or failover testing.
3. From the roles page, delete each role that you created as part of failover or failover testing.
4. Delete any sensitive host data sets.

Clean up of the TKE workstation:

- Use the CCA CLU task to zeroize and "un-own" segment 2 and 3, to delete the zone enrollment for the crytographic adapter in the workstation.
- Delete any hard drive data copied from the primary site.

As a final step, when the primary site is back online, it is a good practice to change your master keys and smart card PINs.

And there you have it. Follow these steps when you create your disaster recovery plan, and you can significantly reduce the time and expense associated with recovery. ■

Staying in the race with STG Lab Services

IBM Systems and Technology Group (STG) Lab Services can help optimize your data center and provide system solutions. STG Lab Services has the vast knowledge and deep skill set to support you through the entire information technology race. We focus on the delivery of new technologies and niche offerings, while collaborating with IBM Global Services and IBM Business Partners to provide complementary services that help lead you through the turns and curves to keep your business running at top speed.

We apply the intellectual and technical capital of IBM development toward making sure IBM Systems products succeed and increase your satisfaction.

Our services for System z include:

- Java performance on System z
- High-availability assessments
- Crisis-readiness assessments
- Tape encryption and IBM Encryption Facility Services
- Getting started with service oriented architecture (SOA) on System z
- Data solutions
- Linux and z/VM
- Virtualization
- Data center services
- IT optimization.

If you don't see what you need, let us know. We thrive on delivering custom solutions and providing assistance with other environments!

Want to take us for a spin? If so, contact **stgls@us.ibm.com** or check us out at: **ibm.com/systems/services/labservices.** ■

zFavorites

zFavorites:
www.ibm.com/servers/eserver/zseries/zos/zfavorites/

It's the zFavorites for System z® credit card CD! You're gonna love this! It has all sorts of helpful Web links, like those for:

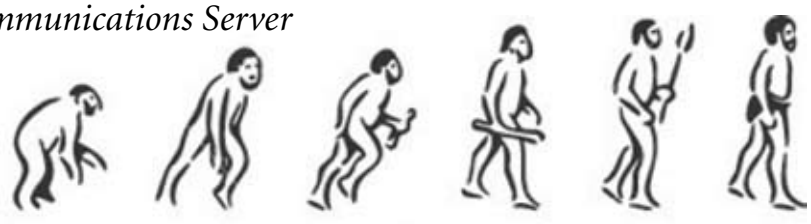
- Hardcopy
- Operating systems
- Software
- Language and tools
- ISV development and applications
- Product documentation
- Marketing information
- Education
- Support
- Links to FREE downloads
- IBM Redbooks sampler
- WebSphere Application Server
- XML

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

Additional copies of zFavorites CD (GK3T-4331-11) are separately orderable.

The evolution to easy

IBM Configuration Assistant for z/OS Communications Server



BY MARK T. WRIGHT AND JEFF CATES

The z/OS operating system has been around for a long time. Many of us z/OS programmers have been around for a long time, too. How long? Back in college, some of us used punched card machines to control those IBM mainframes. (Then one day we found TSO and never looked back.)

If you look around z/OS today, you might still find some things that harken back to the old days.

Like configuration, for instance. For many z/OS elements, doing configuration involves reading thick manuals and entering configuration statements and parameters into data sets.

z/OS Communications Server used to be one of those elements. However, with the arrival of IBM Configuration Assistant, z/OS Communications Server has changed with the times.

Lightweight, but no light weight

Configuration Assistant is a lightweight application that you can install and run on your workstation. And you'll want to, because this tool provides you with an intuitive graphical user interface (GUI) designed to make network configuration tasks easier and less error prone.

You can download Configuration Assistant from the z/OS Communications Server Web site: **ibm.com/software/network/commserver/zos/support/.**

The evolution begins

Configuration Assistant became available with z/OS V1R7. Initially, this tool helped configure IP security, filtering, and Application Transparent-Transport Layer Security (AT-TLS).

For z/OS V1R8, we expanded the tool's capabilities to help you configure Intrusion Detection Services (IDS) and Quality of Service (QoS).

Helping to drive the improvements, people who work with z/OS Communications Server suggested many excellent usability enhancements, as well.

With z/OS V1R9, we continue the evolution with new features, some of which incorporate suggestions we received from people like you.

Let's look at each of these enhancements.

Handy new functions

In z/OS V1R9, Configuration Assistant supports new functions called Policy Based Routing (PBR) and Network Security Services (NSS). PBR adds to the IP routing capabilities of z/OS Communications Server, allowing you to configure additional logic into the routing decisions that are beyond the scope of OMPROUTE, the routing daemon of z/OS.

For example, to separate bulk transfers from transactions, you might want to route all FTP traffic over a specific link. PBR provides this capability by letting you route IP traffic based on the originating application or the originating address. With Configuration Assistant, you have a user-friendly GUI to configure PBR through a series of mouse clicks.

NSS helps you to manage certificates and key rings for communications servers that use IPSec. Here, Configuration Assistant provides a GUI to help you configure NSS servers and clients from a central location. With Configuration Assistant, it is easier to see the relationships between clients and servers.

Saving your data

After you configure a technology, Configuration Assistant helps you install the configuration files to a z/OS policy agent. At this point, you should save your changes in a binary file before exiting. Because these binary files are central to your business, you might need to have this information saved directly on DASD, with automatic backups and other management processes. In z/OS V1R9, Configuration Assistant allows you to save your changes to the z/OS file system.

Further, whenever you start Configuration Assistant, it retains your preferences and downloads your data from this z/OS location during initialization.

In addition, a simple locking mechanism protects this file, preventing multiple users from configuring the same data at the same time (a possibility now that the configurations can be saved at a central location). Even if you save your data on common LAN DASD, the Configuration Assistant locking mechanism protects your data.

This locking function is also incorporated in earlier versions of the tool for z/OS V1R7 and z/OS V1R8. Download the latest level of the tool to ensure that you have this function.

Tracking history

Many z/OS installations have well-established change management procedures that are used for documentation and audits. Because you requested it, Configuration Assistant now provides history tracking — multiple levels of history tracking, in fact. You can enter comments when saving changes or you can emulate current conventions by providing history comments in your configuration file prologs.

Central configuration

In z/OS V1R8, Configuration Assistant helped you configure AT-TLS, IPSec, IDS, and QoS in separate binary files. While this is useful, we learned that you often prefer to configure multiple technologies together, and have health-checking function applied to the configuration data across those technologies.

In z/OS V1R9, Configuration Assistant displays your available technologies and helps you to see which of them can be configured together. See Figure 1.

Configuration Assistant also includes a health check function that scans within a particular technology that you have configured, or across all configured technologies, for errors and oversights. It will also warn you about things that just don't look right.

Given that your configuration can now be consolidated, Configuration Assistant provides an import function to help you migrate an existing z/OS V1R7 or z/OS V1R8 configuration into a single collection for z/OS V1R9. You might, for example, have separate configurations for IPSec and IDS. In V1R9, you can use the import function of Configuration Assistant to combine that data in one place.

FTP enhancements

After updating your configurations with Configuration Assistant, you can install your changes using a built-in FTP client. For this step in the process, some of you indicated that you would like more help with debugging FTP problems.

In V1R9, Configuration Assistant expands its FTP diagnostics to add more information about FTP failures. This includes a complete log of the FTP flows and what specifically caused the failure.

In addition, before z/OS V1R9, the built-in FTP client used only passive-mode data connections. In z/OS V1R9, Configuration Assistant is enhanced to allow you to set FTP connections to use only active mode, to use only passive mode, or to try both.

With the arrival of IBM Configuration Assistant, z/OS Communications Server has changed with the times.

Sorting objects

Configuration Assistant provides a set of reusable objects for each networking technology you might need to configure. One such object is called a traffic descriptor, which is a named configuration object that describes the data traffic for an application. For example, to define TN3270 server traffic, you have a traffic descriptor named TN3270_Server and you attach details to it that describe the data traffic for a Telnet 3270 server. Details might include the protocol of TCP, the local port of 23, or the job name of the TN3270 server.

In z/OS V1R8, Configuration Assistant always showed the list of reusable objects in the order in which they were created. In z/OS V1R9, however, you can alter the sorting order by clicking on the column header.

By default, Configuration Assistant sorts reusable objects alphabetically.

The evolution continues...

Configuration Assistant continues to grow, supporting new technologies and application functions. The goal is to make life easier for you, the person who configures z/OS Communications Server.

By now, you've probably gotten the idea that we actively seek your input as you use Configuration Assistant. Quite true — your comments can help us continue the evolution toward easier network configuration. ■

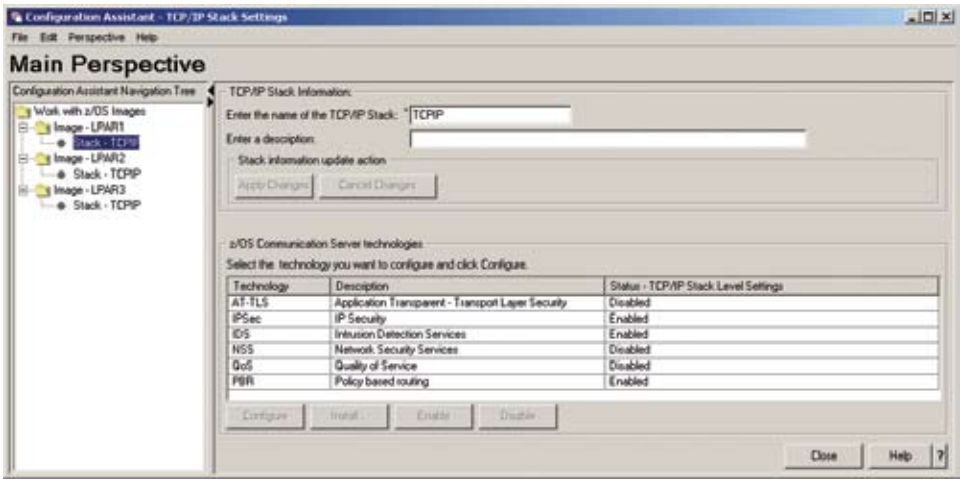


Figure 1. Displaying network technologies in Configuration Assistant

You look mahvellous!

System Automation for z/OS and the Tivoli Enterprise Portal

BY JÜRGEN HOLTZ

For years, the NetView® Command Facility (NCCF) has been the user interface of choice for operators to interact with IBM Tivoli System Automation for z/OS (SA z/OS). The Status Display Facility, with its charming 3270 style, has been used to monitor the state of automation, and NCCF panels have provided various resource views and facilities to make life for operators easier.

While these old, traditional user interfaces are very fast and productive for experienced users, their appeal for the next generation of z/OS system programmers is low. Less-experienced users find them only somewhat intuitive, and the dialog capabilities of these 3270 screens are limited compared to graphical user interfaces in general.

But the biggest issue with such user interfaces is their unique nature — each product has its own console with its own look and feel. You can imagine what a hard time operators have coping with the surveillance of multiple products, to say nothing of the fact that it is impossible to look at more than one product's data at the same time.

Now the first step has been taken to position SA z/OS in an integrated, end-to-end monitoring infrastructure. SA z/OS Version 3.1 is now available in the Tivoli Enterprise™ Portal (TEP). Using this graphical console, you can access a wealth of monitoring data from the Tivoli OMEGAMON® XE products on z/OS. An IBM Tivoli monitoring agent provided by SA z/OS gathers data about the status of automation on z/OS. This data is in turn presented in SA z/OS-specific TEP workspaces in tables or bar charts, side by side with performance data from other IBM Tivoli monitoring products that are installed in the same environment.

Introducing the Tivoli Enterprise Portal

The Tivoli Enterprise Portal, or TEP, is a Java-based graphical user interface that runs either as an application on the desktop or as an applet inside a browser. The TEP presents data in workspaces, each consisting of one or more views. Figure 1 is an example of a typical workspace.

The navigator view in the upper left shows the physical hierarchy of the managed IT environment. The hierarchy spans the whole enterprise, including distributed systems such as Windows and Linux, plus z/OS. The z/OS systems subtree represents z/OS monitors, such as OMEGAMON XE for z/OS and SA z/OS, listed by sysplex name and system name. The remaining space is reserved for product-specific views that usually show related data in a single window to facilitate performance analysis and problem determination.

Common monitoring architecture

A nice characteristic of all these workspaces is that the look and feel is the same for any product that is integrated into the portal. For example, if you are already familiar with using the OMEGAMON XE for z/OS product, you will see no difference when using SA z/OS.

This is possible because all products that are integrated into the TEP are architected using the common IBM Tivoli Monitoring Services infrastructure. The TEP connects to the Tivoli Enterprise Portal Server, which controls data retrieval from the monitoring agents. The portal server in turn is connected to the Tivoli Enterprise Monitoring Server hub. The monitoring server retrieves alerts and collects performance and availability data from the monitoring agents, and passes this information to the TEP for it to display. Finally, the monitoring agents collect data about managed systems and deliver it to the monitoring server to which they are connected.

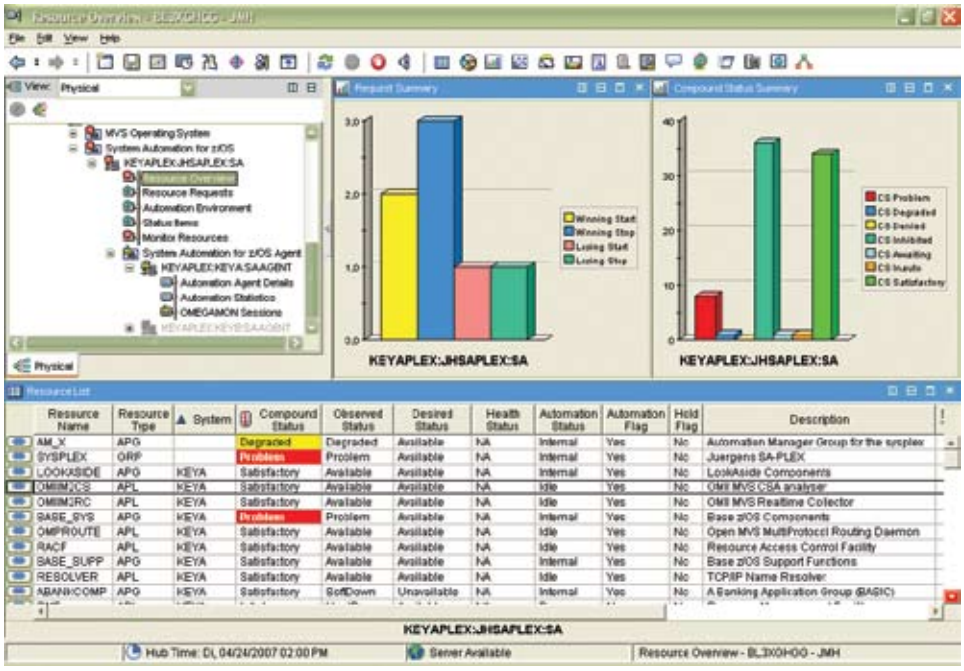


Figure 1. Resource Overview workspace

How does SA z/OS enrich the TEP?

SA z/OS provides a number of predefined workspaces. And, like any other workspace in the TEP, SA z/OS workspaces can be modified and tailored to fit your specific needs. So, you don't like the filters applied to the default views that are delivered with the product? Simply edit the workspace properties and specify your own filters. The capabilities provided by the TEP to customize the workspaces far exceed those of the traditional 3270 world!

Automated resources

The Resource Overview workspace (see also Figure 1) provides a tabular view of all resources in an SA z/OS subplex, including their observed, desired, and compound states. In addition to the table, two bar charts tell you immediately if any resource is in an unsatisfactory status and whether operators have overruled the goals defined in the automation policy. Using the links provided, you can drill down to the Resource Details workspace, where you can find additional information for a given resource, such as forward and backward dependencies, votes, or status messages.

System health

The Monitor Resources workspace shows you at a glance whether your IT environment is doing well. You can use monitor resources, for example, to collect exception data from Tivoli OMEGAMON II® for MVS and map this to a health status. You can then trigger automation if a certain health status is reached to prevent further damage and possibly avoid an outage.

Installation-defined status items

Status items, presented in yet another workspace, are new SA z/OS-controlled variables that allow you to set and maintain the status of any object that you want to monitor in your environment. One customer, for example, uses status items to monitor JES3 SPOOL utilization or to list specific JES3 queues — information that is repeatedly collected through console commands.

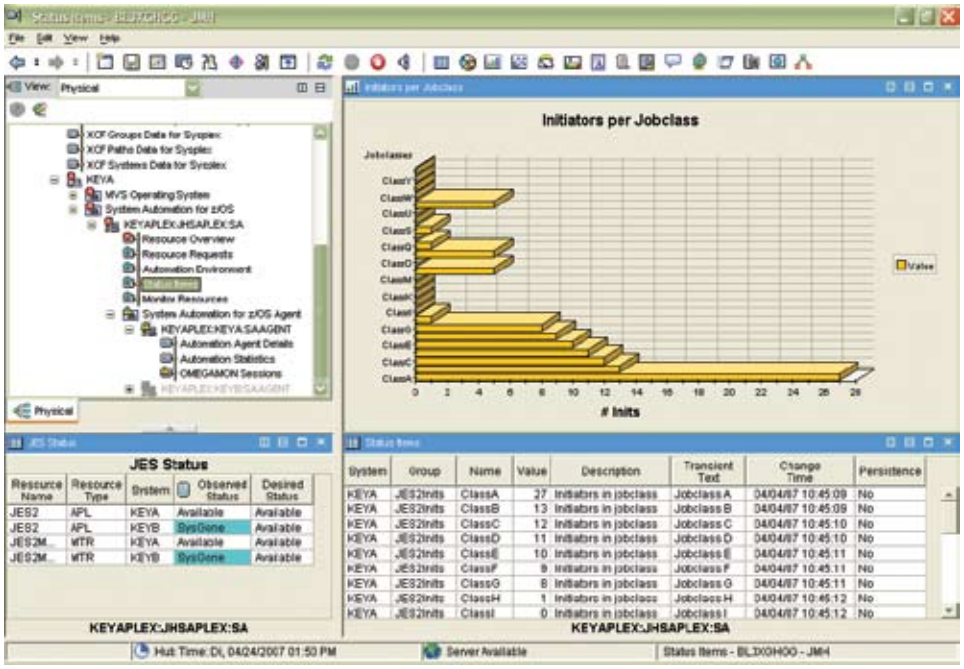


Figure 2. Example of an installation-defined Status Item workspace

Status items are created by the NetView-based automation agent using the new INGSTX command. For example, the following command assigns a value of 10 to a status item called JES3.SPOOL:

```
INGSTX SET JES3.SPOOL
STIVALUE=10 STIDESCR="SPOOL
UTILIZATION" STITRTXT="OK"
```

A status item repository in the SA z/OS automation manager maintains the status items during the automation agent session or, if you want, until the automation manager is cold-started again. Figure 2 is an example of a customized Status Item workspace showing initiator distribution over defined job classes.

More workspaces

Other workspaces provided by SA z/OS show you the status of the automation agents and the automation managers in the local automation environment, as well as detailed request information, such as start and stop requests and the origin of such requests. Additionally, detailed automation statistics are shown for each automation agent, indicating how many messages are being automated or how many commands have been issued in the report interval. If you use OMEGAMON sessions to retrieve health-monitoring data from any of the Classic OMEGAMON monitors in Tivoli OMEGAMON II for CICS, DB2, IMS, and MVS, the OMEGAMON Session workspace reports the status of such sessions and how effective they have been.

Situations

SA z/OS also provides predefined situations that allow you to monitor SA z/OS itself. These generate alerts with different severities if, for example, resources have an unsatisfactory compound or health status, or an automation agent or automation manager is not ready. When a situation occurs, the icon in the navigator view of the related workspace changes color based on the severity of the situation. The different alert colors help operators to quickly locate any areas in the enterprise that need special attention.

One great-looking interface

With the TEP you can monitor the performance and availability of your operating systems, subsystems, and applications, as reported by monitors such as Tivoli OMEGAMON XE for z/OS. With SA z/OS integrated into the TEP you can extend existing views with the operational and automation views of all these resources — all within a single console.

So, are you interested in bringing SA z/OS into the TEP? Then go to **ibm.com/servers/eserver/zseries/software/sa/techresources/tep.html** and read the description of PTF UA33038, where you'll find information about installation, configuration, and usage as well as where to obtain the SA z/OS workspace-enablement CD. ■

HOW TO simplify your WLM administration

BY STEFAN WIRAG AND MARIANNA FRANK

Wouldn't it be nice to edit, analyze, and printout z/OS Workload Manager (WLM) service definitions on your workstation? To make that possible, we developed the *WLM Service Definition Editor (WSE)*, a workstation-based tool that offers features to simplify the complex work of WLM policy administrators.

Getting started

Download and install the WSE installation program from the WLM homepage **ibm.com/servers/eserver/zseries/zos/wlm/**. After installation, configure the built-in FTP-based transfer mechanism to download your service definition from the host. This transfer mechanism enables you to download or upload service definitions that are stored in ISPF tables on the host.

Downloading service definitions

To download service definitions, WSE submits a batch job to the Job Entry Subsystem (JES) that converts the ISPF service definition to XML. The generated XML service definition is then sent to the workstation and displayed in the editor. The upload of service definitions edited with the tool works in a similar way. First, the XML service definition is transferred to the host. Next, WSE submits a batch job to JES that converts the XML service definition to ISPF tables. You can then install the uploaded service definition and activate as usual using the WLM Administrative Application or batch install. WSE can handle service definitions of all z/OS releases.

The transfer mechanism is configured through FTP profiles and FTP connections in WSE.

- In an *FTP profile*, you specify the parameters that are required to log on to the FTP server on the host and to submit a batch job to JES for conversion of ISPF service definitions. Those parameters are your logon userid, the JES job *CLASS* and *MSGCLASS* for the conversion batch job, the host code page, and the high-level qualifier of the ISPF libraries in your installation.
- In an *FTP connection*, you specify a relationship between the ISPF service definition data set on the host and the XML service definition file on the workstation. A download triggered for an FTP connection converts the specified ISPF data set and stores it in the specified XML file. An upload stores the specified XML file in the specified ISPF data set.

FTP profiles and FTP connections are separated so that you have to define required FTP/JES settings only once and can reuse them for the download or upload of different service definitions.

After you have downloaded your service definition, you can explore the following features and functions.

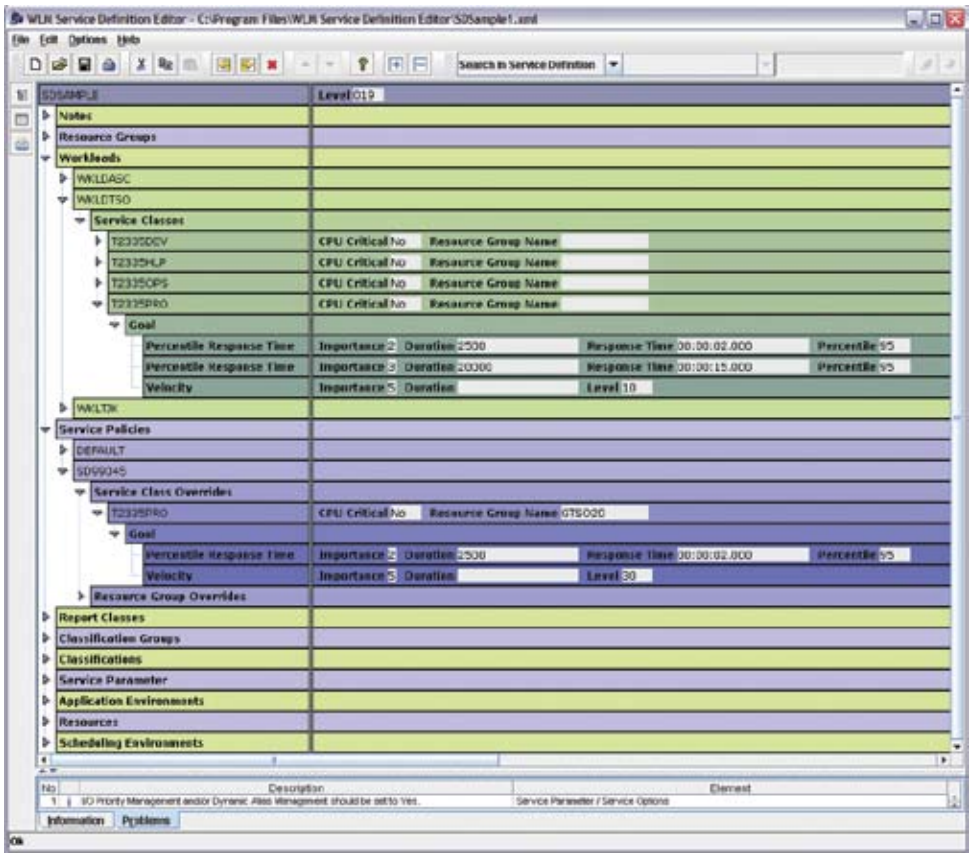


Figure 1. Tree View of a service definition

WLM Service Definition Editor (WSE), a workstation-based tool that offers features to simplify the complex work of WLM policy administrators.

Viewing service definitions

Do you sometimes have to become familiar with a service definition that was created or updated by someone else? WSE offers different views and features to analyze and edit service definitions:

Tree View

The *Tree View* displays the service definition as a tree that is nested according to the structure of the service definition elements. Each node or row in the tree represents one service definition element. The Tree View not only allows you to analyze a service definition; but it also allows you to edit or create a service definition by inserting or deleting elements. When you search the Tree View for arbitrary text, all the nodes that contain the term expand automatically. Furthermore, the Tree View provides direct navigation from the element definition to all the places where this element is referenced. For example, you can navigate from a service class node to the subsystem nodes or service policy nodes where the service class is referenced. Figure 1 shows the Tree View of a service definition with expanded service class nodes.

Table View

The *Table View* displays the service definition as a set of tables similar to a worksheet. Each service definition component (for example, the set of all report classes, service classes, classification rules, and so forth) is displayed in a separate table. The Table View allows you to edit or create service definitions. A nice feature is that although tables are rearranged or sorted you still can edit element properties. Hence, you can move relevant elements and element properties into focus by rearranging a table and then altering element properties. The Table View provides similar direct navigation as the Tree View.

Figure 2 shows the combined workload and service classes table of a service definition. Each row in a table represents one service definition element. To support the investigation of a service definition, you can rearrange tables by moving columns to other positions or by sorting table columns. This allows you to compare certain elements or properties. You can also sort multiple table columns at the same time. For example, you can sort all service class periods according to their goal type, and then you can sub-sort the periods according to the period number in order to verify the velocity level of your velocity periods. The Table View provides a search function for arbitrary text within tables. You can search different locations. You can search through all tables, one table, or only one table column. Matches with the search text are displayed in bold letters. In Figure 2, we searched for the word "TSO" in the entire table.

Print View

The *Print View* displays the service definition as an HTML document. Each service definition component is displayed in a separate section. The Print View does not provide editing or search functions, but displays a service definition in a compact layout. The editor provides the printout of service definitions in the Print View layout. For the printing operation, the editor launches the configured Web browser of the operating system.

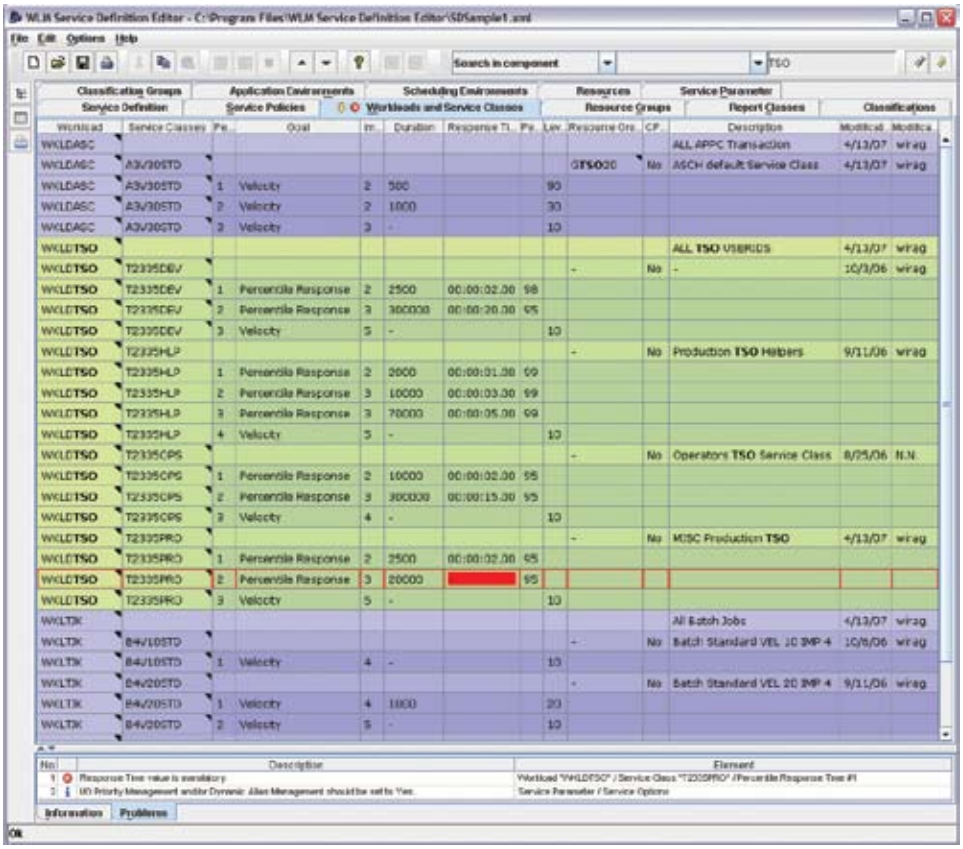


Figure 2. Table View of workloads and service classes with error indication

Developing service definitions

WSE works like an Integrated Development Environment (IDE) for a programming language. This means, you can start with the specification of a new element such as a service class; then you can interrupt the specification to look something up by switching to another view or rearranging and sorting tables, and then return and continue the specification of the service class. In order to enable this kind of interaction, WSE displays all problems and indications for missing elements or properties within the service definition in a problem list. (See Figure 2). You can navigate to the element related to the problem by double-clicking on the problem list entry. To get additional information on the problem just move the mouse cursor on the problem list entry and an explanation as well as a how-to-fix-it hint will appear immediately.

In the Table View and Tree View, the editing of elements works in a similar way. Each row or node represents a service definition element that can be directly manipulated. In order to update the property of an existing element just double-click on the table cell or tree element and you can alter the property. You can choose possible values for properties with a fixed amount of values from a pop-up menu. Use the toolbar or the context menu of a table or tree to insert or delete elements.

The editor supports cut, copy, and paste of service definition elements. Hence, you can extend a service definition quickly by duplicating existing elements. During such operations, the editor takes care of required name uniqueness. That is, it automatically appends suffixes to names so that they become unique.

The editing functions are context sensitive; at each instant, your choices are only those functions that make sense and do not violate the correctness of the service definition. If desired you can operate WSE through short-keys.

Healthy service definitions

Because of many dependencies that have to be considered during the specification of classification rules and the linking of those rules with service and report classes, the specification of an optimal service definition is sometimes not easy. Therefore, WSE has built-in health checks and hints. While you are editing a service definition, WSE checks in the background the specification and displays warnings or informational messages as appropriate in the problem list. For example, you are notified that velocity goals are possibly too high, of heterogeneous report classes, missing default service or report classes, missing classification rules for system-related address spaces, and so forth. The health checks and hints are based on recommendations from IBM and other independent consultants.

Working with multiple service definitions

WSE allows opening multiple service definitions simultaneously. Each service definition is displayed in a separate editor window, which means you can compare service definitions online. Furthermore, you can copy elements of one service definition and paste them into another service definition. For example, you can copy a whole subsystem specification with all classification rules or a workload with all associated service classes with one copy and paste operation. Using this feature makes it possible to compose a new service definition by reusing definitions from existing service definitions with a few mouse-clicks or keystrokes

Simplify

There you have it, a convenient way to simplify tasks for z/OS WLM service definitions on the workstation. WSE helps make administering WLM easy. ■

Long ARM of the law

New CICS and IMS functionality for EWLM on z/OS

BY SUE SHUMWAY AND HIREN SHAH

If you use IBM Enterprise Workload Manager (EWLM) to manage workloads across your enterprise, wait until you see what's rolling into town. Peeling around the corner with lights and sirens blaring, new levels of CICS and IMS that add all-new Application Response Measurement (ARM) 4.0 capability are sure to get your scanners buzzing.

Application Response Measurement (ARM)

ARM 4.0 is the force behind the ability of EWLM to monitor transaction flows among ARM-instrumented applications across your entire EWLM domain. This allows you to extend your enterprise management tools to applications and create a comprehensive end-to-end management solution.

EWLM extends capabilities of z/OS Workload Management (WLM) services to all members of the IBM eServer family and enables you to monitor workloads across heterogeneous environments that contain multiple, interacting servers.

To accomplish this, EWLM influences WLM goal-oriented management decisions by mapping EWLM policies to WLM policies and processing the ARM correlators that partner up with transactions as they are processed around the domain.

CICS Transaction Server for z/OS (CICS)

With previous levels of CICS, if you wanted to manage your CICS transactions on z/OS, you had only the WLM proprietary instrumentation interfaces at your disposal. CICS TS V3.2, however, strengthens that method by adding ARM correlators and enhanced semantics for the proprietary interfaces to its holster.

CICS TS V3.2 now boasts support for internal management of ARM correlators that buddy up with transactions, giving those ARM correlators a warrant to enter and search CICS and report back to EWLM. CICS requests the correlators from EWLM, accepts them from the various transports, and forwards them on. CICS then uses an enhanced version of the existing WLM instrumentation that enables WLM to construct data collection in an ARM-suitable format for EWLM consumption.

Figure 1 illustrates this mission, showing how ARM correlators can now pass between CICS, WLM and EWLM and provide end-to-end transaction flow and identification data in the EWLM Control Center.

For more information about CICS TS V3.2, see **ibm.com/software/htp/cics/tserver/v32/**.

Information Management System (IMS)

As with CICS TS V3.2, the new IMS V10 now sports ARM enablement as well. This new EWLM support allows you to include IMS when you manage workloads across environments and displays end-to-end IMS transaction flow data in the EWLM Control Center.

For more information about IMS V10, see **ibm.com/software/data/ims/v10/**.

To protect and serve

There's no way to escape the long ARM of the law, and no reason you'd want to! These new levels of CICS and IMS help protect and serve with the new ARM capabilities they provide, so check them out and start building your own force! ■

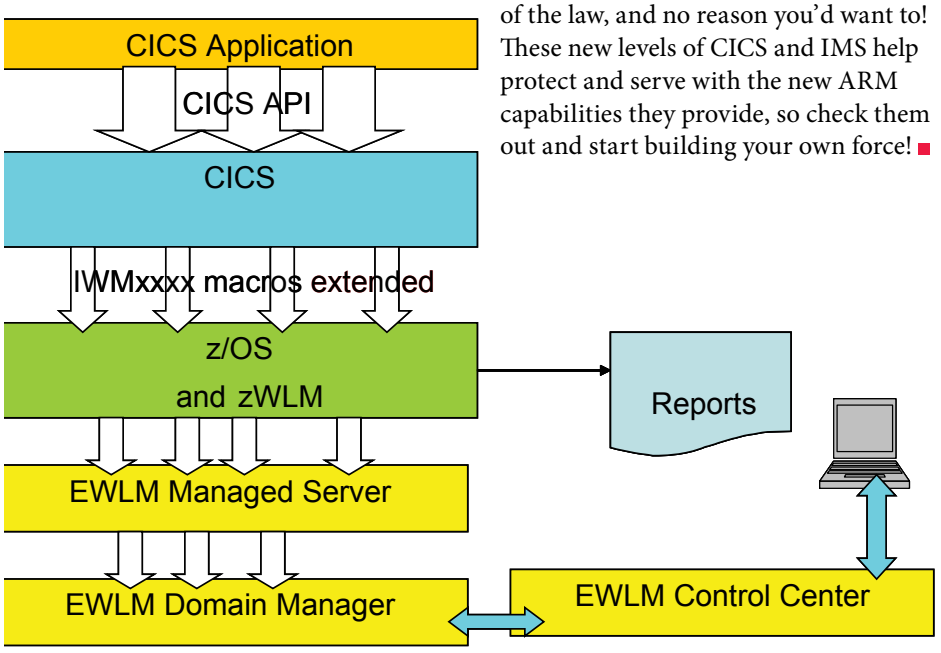


Figure 1. CICS, WLM, and EWLM end-to-end transaction flow in the EWLM Control Center

Go for the gold!

Mining the XCF data mountain

BY HARALD BENDER

Prodigious mountain peaks and treacherous descents are fine for the great outdoors, but there is nothing very picturesque about towering stacks of performance data reports looming over your desk. Yet, when it comes to things like XCF, having extensive data can be a good thing — as long as you can easily convert that "mountain" into a unified picture of your sysplex. Such an aerial view would be useful for unraveling mysteries related to XCF performance at your installation.

This massive metaphor leads us, quite naturally, to RMF Spreadsheet Reporter and its new XCF Trend Report macro. With this new function, you can:

- Load and combine XCF data from multiple systems
- Generate graphical views for your XCF sysplex performance.

SMF is where we begin

Before you can analyze your data, you need to have the data available on your workstation in a spreadsheet format (uh-oh).

No worries, we'll get there! Usually, your starting point is the mountain of raw XCF data, collected on your systems and stored to SMF Type 74 Subtype 2 records.

Tackling the mountain

Now it's time to turn that data mountain into a molehill by creating readable XCF reports from your SMF data.

In RMF Spreadsheet Reporter, do the following:

1. On the Resource notebook page, specify as input the SMF dump data sets from all of the systems in your sysplex.
2. Select the XCF Activity option on the Reports selection dialog and use the remote job execution engine function to run the RMF Postprocessor from your workstation. The Postprocessor creates a listing data set, which contains XCF reports — usually lots of them — all with a single-system scope.
3. Download the Postprocessor listing data set to your workstation.
4. Extract the reports from the listing.
5. Convert the reports to spreadsheet format.

Here's the best part: You can combine these steps into a single task through the Create Working Set function of RMF Spreadsheet Reporter.

Sifting for data gold

At the bottom of the available spreadsheets list, find the new XCF Trend Report macro. Open this macro and load your XCF data created in the previous step, as follows:

1. On the Main notebook page, choose Select Report Working Set and process data…
2. From the list of available working sets, choose the desired set and select OK.
3. Use the Select Systems dialog to combine the XCF data from multiple systems.
4. Use the Select Interval dialog to include or exclude data from individual intervals.

Forest, trees, or both

The RMF Postprocessor XCF Activity Report consists of three sections, as follows.

XCF Usage By System

This section shows XCF buffer statistics on transport class level from the scope of the individual system images.

XCF Usage By Member

This section displays request statistics for XCF groups and their members. In most cases, a component joins one member per system to an XCF group.

XCF Path Statistics

This section consists of activity and delay statistics on path granularity. Either a coupling facility structure or a channel-to-channel (CTC) device can represent an XCF path.

The XCF Trend macro processes data from these sections. It consists of various spreadsheets with colored tabs:

- Select the yellow tabs for data about XCF usage by system.
- Select the green tabs for XCF path statistics.
- Select the blue tabs for actions, such as loading data, or online help.

From the group of yellow tabbed spreadsheets, for example, let us look at the TCSystems spreadsheet (Figure 1).

Here, the metric selection drop-down box allows you to select between all basic metrics contained in the XCF Usage By System section. Based on the request counts, you can also choose all kinds of rates for display.

In this example, the spreadsheet makes the main facts obvious:

- Most of the XCF signals are assigned to the transport class DEFAULT.
- Signal activity reaches its peak at 1:00 pm.

Perhaps, you're not as keen on information about the Transport Class level as you are to know which systems are the high XCF signaling contributors. If

Figure 1. TCSystems spreadsheet with outbound request rates.

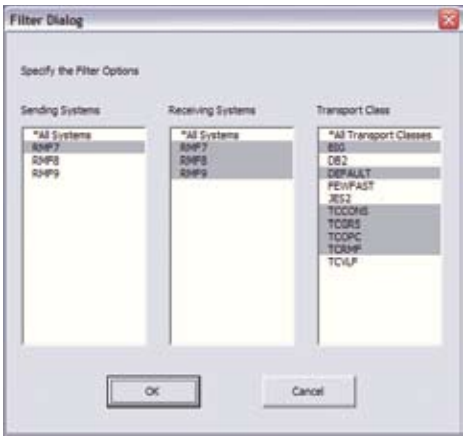


Figure 2. Filter dialog for the TCSystems spreadsheet.

so, you can, from the Reporting Category drop-drown box, change the scope from Transport Classes to Sending Systems or Receiving Systems.

When you finish your adjustments, the chart might still look overloaded because of the high number of transport classes and systems. That is why we added the filter dialog. Use it to hide the irrelevant data and focus on those transport classes and systems that are important for your personal analysis as Figure 2 shows.

Going green (tabs)

In a large installation, having fifty or more XCF paths is not unusual. As a result, the "data mountain" for XCF data on path granularity can grow larger than you would care to climb.

Don't worry; simply use the spreadsheets with the green tabs. The PathSystemsOut spreadsheet shows the metrics contained in the XCF Path Statistics section of the Postprocessor report. Once again, the RMF Spreadsheet Reporter filtering capabilities allow you to focus on the essentials. See Figure 3.

Instead of presenting lots of single-system reports that you would need to analyze individually, the PathSystemsOut spreadsheet in Figure 3 makes the important thing clear immediately: Most of the XCF signaling traffic in this example uses one coupling facility structure, IXCPLEX_PATH4.

At this point, you might use the PathActivity spreadsheet to determine whether this structure is causing a bottleneck. If so, you could take some actions to achieve better balancing.

Climb aboard!

The new XCF Trend Report macro is included with RMF Spreadsheet Reporter Version 5.2.4, which is part of the RMF product. You can get the latest version without charge at the RMF Web site: **ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/rmftools.html**. ■

Figure 3. PathSystemsOut spreadsheet (filtered)

HOW TO navigate zFS

BY ANN TOTTEN AND CHARLES HAIGHT

As you have probably heard by now, IBM has stated that, beginning with z/OS V1R7, zSeries File System (zFS) is the strategic z/OS UNIX physical file system (PFS). This means that, although there is no current plan to discontinue support of hierarchical file system (HFS), all further physical file system enhancements will pertain only to zFS. From a user's point of view, the physical file system type (HFS or zFS) is irrelevant. They both look and feel the same. The zFS file system is a new frontier with some subtle differences and a whole new command interface to assist you. We'd like to tell you about some of these new commands that we think will help you in your effort to tame this frontier.

Anatomy of zFS

Before we jump right in to this new land of zFS administrative commands, let's talk about the anatomy of zFS. zFS is actually two parts. The first part is the zFS aggregate, which can be thought of as a container for the file system; it is a VSAM linear data set. The second part is the actual zFS file system and it is contained within the aggregate or VSAM linear data sets (LDS).

There are two types of zFS aggregates. The first type is the HFS compatibility-mode aggregate, in which there is one aggregate and one file system that share the same name. This type is much like an HFS file system where the HFS maintains a one to one correlation with a PDSE. The mount for the HFS compatibility-mode zFS is issued the same way that a mount for an HFS is issued. That is, there is no explicit attach for the VSAM LDS before issuing the mount; the zFS code does the attach-on-mount and the detach-on-unmount. The second type of aggregate is the multi-file system aggregate where one zFS aggregate plays host to several zFS file system.

Now here is the curve ball — forget all about the multi-file system aggregate. As of z/OS V1R8, you are no longer able to mount file systems contained in a multi-file system aggregate in a shared file system environment. IBM will discontinue support for multi-file system aggregates in a future release. That being said, our discussion of zFS administration commands will focus on compatibility mode aggregates and file systems.

HFS and zFS mount commands

Let's fast-forward to when you finish your HFS to zFS migration. Here are some of the basic similarities and differences that you will encounter.

The mount commands are the same for HFS and zFS. The only exception is that the file system type (FILESYSTYPE) is now zFS instead of HFS. z/OS has added migration aids over the last several releases and one of those enhancements sniffs around for a data set name that matches the file system name specified on the mount command. If z/OS UNIX finds a VSAM LDS with a matching name, it directs the mount command to the zFS PFS. If a PDSE is found, the mount command is directed to HFS. So for now, you don't need to change the TYPE parameter on BPXPRM, automount policies, /etc/rc, or any other place. Eventually, you'll want to change the value to avoid the extra processing.

zfsadm commands are:

aggrinfo	obtain information on an attached aggregate
apropos	search by help text
attach	attach an aggregate
clone	clone a filesystem
clonesys	clone multiple filesystems
config	dynamically reconfigure the zFS kernel
configquery	query configuration settings for the zFS kernel
create	create a filesystem
define	create a VSAM linear dataset with SHAREOPTIONS(3,3)
delete	delete a filesystem
detach	detach an aggregate
format	format an aggregate
grow	grow an aggregate
HELP	get help on commands
level	display service level
lsaggr	list aggregates
lsfs	list filesystem information
lsquota	list filesystem and aggregate space usage
lssys	list systems
query	query information for the zFS kernel
quiesce	quiesce an aggregate
rename	rename a filesystem
setquota	set filesystem quota
unquiesce	unquiesce an aggregate

Figure 1. zfsadm help information

The zFS file system is a new frontier with some subtle differences and a whole new command interface to assist you.

zfsadm commands

Now that we have the file system mounted, it's time to consider the new zfsadm commands. This article discusses only a few, however for details on all the zFS commands, see *Distributed File Service zSeries File System Administration*, SC24-5989 in the z/OS Library at **ibm.com/zseries/zos/bkserv/r8pdf/dfs.html** .

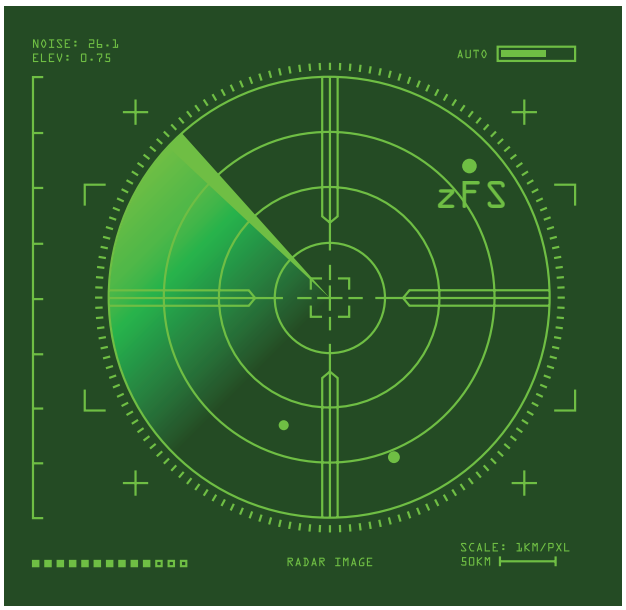
If you happen to find yourself in front of your keyboard in the shell, you can simply type `zfsadm help` to see a list of zFS commands along with a short explanation of each one. Append `–help` to a command to find out the proper arguments and usage for it.

For example, one of the shell commands that system programmers and administrators often use to display the free space on a mounted file system is the df command. By default, the df command displays the space in 512 byte units.

Figure 1 shows you the information displayed when you type `zfsadm help`. On this list you see the zFS counterpart to the df command is `zfsadm lsquota`, which shows the file system and aggregates size, the quota in 1KB units. The quota of a zFS file system is the logical size of the file system. The `zfsadm lsquota` command also shows the size of the aggregate and usage in 1KB units

To see how to enter this command against our subject file system, enter `zfsadm lsquota –help` to see the usage statement as shown in Figure 2.

Now, just plug in the appropriate arguments to obtain the space information.




```
IOEZ00243I Usage:
zfsadm lsquota  {-filesystem <name> | -mfilesystem <mount name>}
[-aggregate <aggregate name>] [-level]  [-help]
```

Figure 2. Usage statement for -help

Other zfsadm command options

Here are some helpful zfsadm command options:

zfsadm lsaggr

The `zfsadm lsaggr` command is very useful if you want a list of all aggregates that a specific image in your sysplex owns.

If you type `zfsadm lsaggr –system SYSX` where SYSX is your LPAR, you can view a list of all of aggregates owned by that LPAR along with the mode (R/W, R/O, and so on). The `df –v` command can provide similar information, but with a lot of additional information (and the information it returns doesn't discriminate by LPAR).

zfsadm aggrinfo

The `zfsadm aggrinfo` command is similar to the `df –v` command, but has a couple special uses. You can enter the `zfsadm aggrinfo` command for a specific aggregate or for all the attached aggregates. One thing that `zfsadm aggrinfo` shows that `df –v` doesn't is whether an aggregate is quiesced. This is useful information to know because sometimes end users don't understand why the file system is unavailable. Because the D OMVS,F command doesn't show that a zFS file system is quiesced, it is yet another reason why you might want to use `aggrinfo`. This command also shows space available based on the different sizes that zFS stores them in. This is useful to see if a specific size has been depleted thus making it appear to the user that there is space available even though there is no more space for the file they are attempting to save.

zfsadm lsfs

The `zfsadm lsfs` command does not really match any existing command, but displays a couple of interesting pieces of information. Like the other commands mentioned, this one can be entered against a specific aggregate or all of them. Using this command, you can see the creation and update date for that aggregate. The date is sometimes helpful to know when searching for a specific file system or when trying to see whether a file system predates some specific event. Secondly, the command shows whether an aggregate has been cloned. *Cloning* is taking a point-in-time snapshot of the metadata in the file system. If this snapshot exists, it resides in the aggregate along with the compatibility-mode file system, and has the same name but with a .bak extension.

zfsadm grow

The `zfsadm grow` command increases a full zFS aggregate by the size specified. This, of course, is dependent on whether there is more space available for it to grow (and you defined the aggregate with a secondary allocation).

Exploring gems in a new frontier

This article provides a brief tour of the new zFS world and its commands. As you can see, there are many zfsadm commands designed to help you with the administration of your file systems. If you take the time to explore the new zFS frontier, we're sure you'll enjoy the scenery. ■

Supersize your memory and it just might increase your performance

BY ED MEKEEL AND MARK WISNIEWSKI

We've all seen the ads from a certain fast-food chain, allowing you to increase the value of your meal purchase by selecting the "supersize" option. Actually, it happens in lots of retail markets. It's an appeal to consumers who want to get the most from the dollars they spend.

Well, in a similar vein, configuring your processors with additional memory might help you get the most out of your hardware, by taking a bite out of performance issues and increasing application throughput.

This article describes at a high level some of the possible performance benefits of configuring and using more memory, such as savings in elapsed time and response time.

However, as with some fast-food restaurant patrons, your computing environment might not be hungry enough to consume all that extra memory, and you might be satisfied with current or "regular" memory configurations. To know for sure, you would need to conduct a detailed analysis of your environment to identify the performance advantages you might see from configuring more memory. We hope to talk more about this in a future article in *z/OS Hot Topics Newsletter*.

Value meal

Reducing DASD I/O and keeping the data in memory, close to the actual processing, has long been proven to have measurable performance value. It just makes sense to avoid extraneous CP and elapsed time to reference a buffer pool, requiring a DASD I/O to retrieve the data, and interrupting the current work being executed. By configuring more memory and by referencing processor memory, you might be able to eliminate read hits against the cache for DASD control units. The more unnecessary I/Os you can avoid, the greater the potential for performance savings.

Extensive menu

Currently, IBM System z hardware supports a variety of memory sizes:

- IBM eServer z990 server supports up to 256 GB of memory.
- IBM System z9 server supports up to 512 GB of memory.

z/OS V1R8 and later releases support the maximum amount of processor memory (up to 4 terabytes).

By supporting larger amounts of memory on the larger systems, we support the growth of your environment and applications. We continue to stress-test our offerings to find and fix constraints.

z nuggets

Some of our studies of configuring and using larger amounts of memory to exercise a range of workloads and products are demonstrating good performance benefits, such as:

- Reduced response times for DB2 transactions
- Reduced system CPU time
- Less database I/O
- Reduced completion times for batch jobs.

Performance results might look something like those for the sample environment shown in Figure 1. In this database-workload study, we progressively increased memory to monitor the effects on system performance over three separate measurements, and then projected what results might look like with more or less memory. There were some nice double-digit performance improvements in some of the metrics as memory just about doubled from the first to the second measurement point.

As you can see, there is a point of diminishing returns — that first chicken nugget tastes a lot better than the twentieth. This finding gets back to that predictive methodology we mentioned earlier to assess the performance advantages you might see from more memory. However, for certain environments, there can be real advantages to getting past the knee in the curve.

Combo meals

Studies show that keeping your hardware and software current can produce substantial performance advantages. For example, by moving to z9 servers with enhanced I/O functions including fibre-channel connection (FICON) 4 and Modified Indirect Address Words (MIDAWs), along with configuring more memory, more work can be done for the same million service units, and the same work can be done much faster.

We'll keep adding to the menu

We continue to study how increasing the amount of memory can affect typical workloads. Our intent is to provide you with as many options as possible to ensure that you achieve the performance and throughput levels your business demands. As the studies yield consistent results, we'll make sure to update you with the latest findings. ■

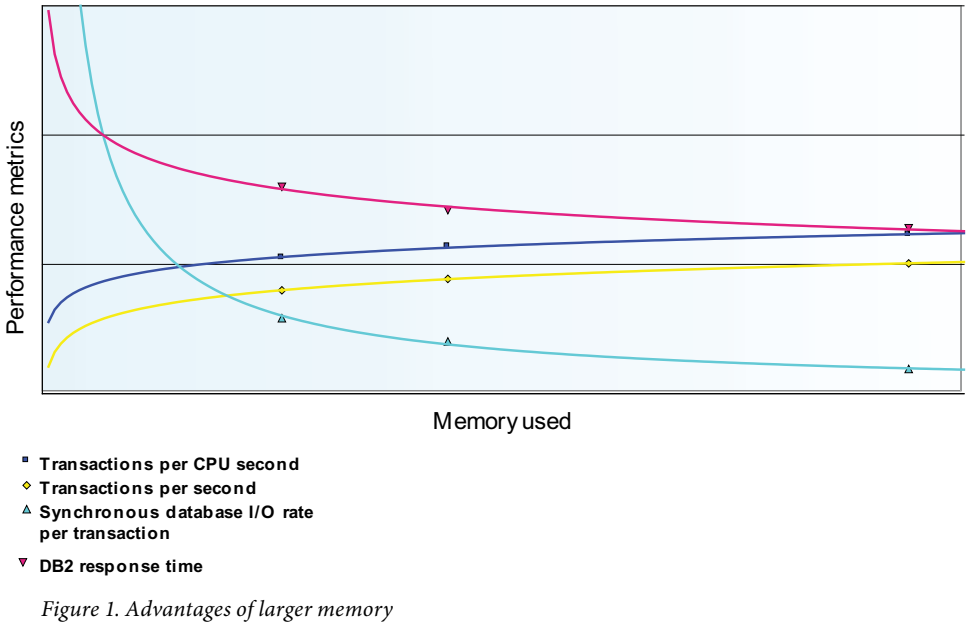


Figure 1. Advantages of larger memory

z-Doku ANSWERS

FROM PAGE 31

M A I N F R a m E

R	a	A	I	m	F	N	E	M
I	F	E	M	N	A	m	R	a
m	N	M	a	R	E	I	A	F
N	E	a	m	A	M	R	F	I
M	A	I	N	F	R	a	m	E
F	R	m	E	I	a	A	M	N
a	m	F	A	E	N	M	I	R
A	M	R	F	a	I	E	N	m
E	I	N	R	M	m	F	a	A

Write your own check!

z/OS Health Checker

BY KRISTINE LOGAN

Hopefully, you're already using the IBM Health Checker for z/OS, part of z/OS since V1R7, to keep an eye on the health of your system. The IBM Health Checker for z/OS framework runs checks on your live system, dynamically alerting you to potential problems, places where your installation is not in sync with the best practices and recommendations that can help your system look and feel its best. Checks won't change your system, but they will help you head off problems and boost availability by issuing messages to report the results of the check. When a check finds a potential error, the message will be a write-to-operator (WTO) message that includes details about the problem and how to resolve it. In the message buffer, you'll also find the message details.

Manage your checking

You can manage checks using SDSF, IBM OMEGAMON for z/OS Management Console, or commands. And you can customize IBM Health Checker for z/OS for your particular installation. We have lots of great checks available, and the number is growing. They often roll out between z/OS releases, so for up-to-the-minute information on what checks are available now, at **ibm.com/servers/eserver/zseries/zos/hchecker/check_table.html**.

Now create your own checks

No one knows your system and your applications better than you do, so we think it's time you took the plunge and wrote your own checks, for your installation and for your customers. A quick look at your outage history, your support calls, or your product documentation will bring up any number of ideas, we're sure. Create your own personal trainer, hall monitor, or enforcer for your installation!

Choose your check type

There are a couple of types of checks you can write:

- Assembler checks – Everyone loves to write in assembler, right? Okay, some of you love it for its geeky and powerful assembler services. If you want to use assembler services in your check, write either a local or remote assembler check. The basic difference between local and remote checks is that local checks run in the IBM Health Checker for z/OS address space, while remote checks run in the caller's address space. This means that:
  - Local checks are simpler to write than remote checks, but the data you can access from the IBM Health Checker for z/OS address space is a little more limited than that you might be able to access from a remote check running in the caller's address space. In addition, your check must avoid any potentially disruptive actions like I/O intensive operations, serialization, or waits. Local checks must be APF-authorized.
  - Remote checks have more freedom. This can make it easier to access data and read data sets. Write a remote check when your check cannot access the data it needs from the IBM Health Checker for z/OS address space or needs to perform potentially disruptive actions. Remote checks do not have to be APF-authorized. For a remote check, you must provide synchronization and communication, including, for example, a pause element token. But we have great examples to show you how to do this at the end of this article.
- REXX exec checks – Finally, starting with z/OS V1R9 (and getting rolled back to V1R8), you can use a hip language to write your checks! Take advantage of REXX, which makes it easy to read and write data, as well as to parse data and perofrm I/O. REXX exec checks run under System REXX. IBM provides three System REXX interfaces: HZSLSTART, HZSLFMSG, and HZSLSTOP.

IBM Health Checker for z/OS provides lots of stuff to make check writing easier

When you write your check, take advantage of the perks provided to make it easier, including:

- A two KB check work-area provided for check data for both assembler and REXX exec checks.
- Return and reason codes that are set for your check when you issue a message or stop the check. For assembler checks, you'll issue messages and stop checks using HZSFMSG REQUEST=CHECKMSG and HZSFMSG REQUEST=STOP interfaces. For REXX exec checks, use HZSLFMSG and HZSLSTOP System REXX interfaces to issue messages and stop checks.
- Assembler macros and System REXX interfaces for use in checks.

A checklist for writing an effective check

A few simple points can help make your check really effective:

- Confine a check to checking one setting. That will make checks easier for users to resolve and manage, and easier for you to support if something goes wrong.
- Make checks tunable, so that users can customize them for their systems. For example, make the specific values your check looks at into parameters with default values that users can customize. Figure 1 shows how a check can use the value it is checking as a parameter.

```
/*********************************************************************/
/*                                                                   */
/* Put code here to process the parameters in HZS_PQE_ParmArea.      */
/* Quotes that were specified in the PARMS value are not included    */
/* in HZS_PQE_PARMAREA or its length.                                */
/*                                                                   */
/* For our example,                                                  */
/* - assume that the required parameter string is LIMIT(nnn) where   */
/*                                                                   */
/*  nnn is a 3-digit decimal number.                                 */
/* - If the syntax is not correct, call HZSLFMSG with a STOP         */
/*   request for a reason of BADPARM                                 */
/*   o HZS1001E is issued for the error.                             */
/*   o The check is disabled with a status of BADPARM                */
/*   o The check is not called again until the PARMS are updated     */
/*      with a new value.                                            */
/* - Parse message HZS1209E, HZS1213E, HZS1214E, and HZS1215E are    */
/*   issue to the console and placed in the message buffer.          */
/* - The modify command is used to UPDATE check PARMS                */
/*                                                                   */
/*     F hzsproc,UPDATE,CHECK=(chkname,chkowner),PARMS=('value')     */
/*                                                                   */
/*********************************************************************/
If HZS_PQE_LOOKATPARMS = 1 THEN
  DO
    Parse UPPER Var HZS_PQE_PARMAREA,
                    "LIMIT("Limit_Value")"EndLimitVar+1
    Select
/*********************************************************************/
/*  Go on to process the parameter                                   */
/*********************************************************************/
```

Figure 1. Using a value as a parameter for a check

```
/*********************************************************************/
/* Process parameters for HZS_SAMPLE_REXXTSO_CHECK                   */
/*********************************************************************/
/*                                                                   */
/* For our example,                                                  */
/* - assume that the required parameter string is DSN(value) where   */
/*  value is the name of a sequential data set that contains data    */
/*   to be processed by this check. We use TSO services to do the    */
/*   validation.                                                    */
/*                                                                   */
/*********************************************************************/
 ADDRESS TSO "Alloc "HZS_PQE_PARMAREA" SEQ OLD"
 IF RC ^= 0 THEN
    DO
      HZSLFMSG_REQUEST = "STOP"
      HZSLFMSG_REASON = "BADPARM"
      HZSLFMSG_RC = HZSLFMSG()
      IF HZS_PQE_DEBUG = 1 THEN
       DO                  /* Report debug detail in REXXOUT      */
         SAY "PARMS: ||"HZS_PQE_PARMAREA"||"
         SAY "HZSLFMSG RC"  HZSLFMSG_RC
         SAY "HZSLFMSG RSN" HZSLFMSG_RSN
         SAY "SYSTEMDIAG"   HZSLFMSG_SYSTEMDIAG
       END
       EXIT                /* The check is not performed          */
    END
```

Figure 2. Using a TSO service in a REXX exec

Figure 2 shows how you can use TSO services in a REXX exec to evaluate the parameters in a data set.

- Make sure that your check doesn't raise unnecessary exceptions. After a user has seen and resolved a check exception, they shouldn't have to see it again unless something changes. Make sure your check issues exception messages that give all the information needed to resolve the exception. And if the check is tunable, the users can customize check values, so that they don't keep getting exceptions when a check they've resolved already runs.
- Design your check to stop itself when the environment is inappropriate for the check or if the parameters are not valid. For example, if your check will only provide useful results if it runs in a sysplex environment, have it stop itself rather than run in a non-sysplex environment. If your check detects a parameter that is not valid, have it stop itself using use HZSFMSG to communicate the check's results, as shown in the following figure.

```
If Limit_Value = "" THEN
           DO
             HZSLFMSG_REQUEST = "STOP"
             HZSLFMSG_REASON = "BADPARM"
             HZSLFMSG_RC = HZSLFMSG()
```

- Concentrate on check messages. Your check messages are arguably the most important part of your check. They tell the user what the problem is, what the impact might be, and how to resolve the potential problem.
- When you select a default severity for your check (high, medium, or low), keep in mind that users will automate check messages, so that the message might result in automated action, based on severity.
- Wherever possible, have the system run your check, not only at intervals, but when applicable values change. For example, a check that looks at a resource name list (RNL) should run at intervals, as well as whenever the system detects a change in the RNL. You can use the HZSCHECK REQUEST=RUN interface.

The IBM Health Checker for z/OS framework runs checks on your live system, dynamically alerting you to places where your installation is not in sync with the best practices and makes recommendations to help your system look and feel its best.

Use the examples!

We have what you're really looking for – examples. There are examples at the following Web site: **ibm.com/servers/eserver/zseries/zos/integtst/samplel.html#health**

Local check examples in SYS1.SAMPLIB:

- HZSSADCK - Sample HZSADDCHECK exit routine
- HZSSCHKR - Sample local check routine
- HZSSMSGT - Sample message input.

Remote check examples in SYS1.SAMPLIB:

- HZSSRCHK - Sample remote check routine.
- HZSSMSGT - Sample message input
- HZSSCHKC – sample of using the HZSCPARS parameter parsing service.

REXX-exec check examples in SYS1.SAMPLIB:

- HZSSXCHK – Sample REXX exec check
- HZSSXCHK - Sample PARMLIB statements for adding a REXX check
- HZSSMSGT – Sample message input.

And for even more information...

Use the following resources to find all the information that you need for IBM Health Checker for z/OS:

- For complete descriptions of all the checks, as well as how to manage them, see the reference section in the latest copy of the *IBM Health Checker for z/OS User's Guide*, SA22-7994 at **ibm.com/servers/eserver/zseries/zos/hchecker/**.
- To see the latest z/OS checks available and how to obtain them, see **ibm.com/servers/eserver/zseries/zos/hchecker/check_table.html**
- For further information see "An apple a day… keeps the PMRs away!" in *z/OS Hot Topics Newsletter Issue 13*, August 2005, GA22-7501-09.
- Look for the following IBM Health Checker for z/OS SHARE presentations at the SHARE Web site (**www.share.org/**):

Share, Tampa, 2007

2826 - Writing A Remote Health Check in z/OS V1R8

2818 - What's New in Health Checker for z/OS V1R8

Share Baltimore 2006

2827 - Writing Your Own Health Checks for z/OS V1R7

2818 - What's New in Health Checker for z/OS V1R8? ■

Level 2 with me

BY EVAN HARUTA AND KATHY PFEIFFER

In our latest Level 2 with me, we talk about recognizing potential problems before they can become real ones. Let's look at what some hypothetical users say about installation and migration, and how it can affect your JES2 and Program Management Binder environments.

Never underestimate the power of preparation

If you plan to install or migrate JES2 (z/OS V1R7 or later), or plan to invoke the Program Management Binder on z/OS V1R8, read the following advice for tips on how to install and migrate these components successfully to prevent undesirable results.

Thanks to Kris Puzza and Ritu Bhargava from the Level 2 test team, and Marna Walle from System Build and Installation for their help with this version of "Level 2 with me."

JES2 checkpoint data set errors

Dear Level 2,

During z/OS V1R7 disaster-recovery testing, I received the following JES2 error messages. Message $HASP276 indicates that JES2 initialization processing is unable to access the displayed checkpoint data set. I haven't changed the definition of my checkpoint data set. Did something change in JES2 z/OS V1R7 that would cause the checkpoint data set to be unrecognized?

```
$HASP060 ISGENQ FAILURE - RC=00000010 RSN=00001006
   RNAME=SYSSP3SYS1.HASPCKP3
ISG363I SYNCHRES RESERVE FLUSH INCOMPLETE FOR
   DEVICE: 0213
   IOSVDSTF RC: 8

   SYNCHRES REASON: API SPECIFIED.
   RESOURCE: SYSZJES2, SYSSP3SYS1.HASPCKP3
   JOBNAME: JES2    ASID: 0018 TCBADDR: 00AFF530
$HASP284 JES2 INITIALIZATION CHECKPOINT DIALOG STARTED
$HASP276 JES2 CAN NOT USE THE REQUESTED CHECKPOINT
   DATA SET(S) DURING A COLD START BECAUSE OF I/O
   ERROR RESERVING CKPT1
```

Signed: *Unresolved checkpoint*

Dear Unresolved checkpoint,

In z/OS V1R7, JES2 replaced the RESERVE macro with ?ISGENQ SYNCHRES=YES. If the JES2 checkpoint data set resides on a DASD volume that was defined as SHAREDUP, the ?ISGENQ call will not be able to obtain the immediate reserve and will lead to the above errors.

Advice and recommendations:

Before installing or migrating to z/OS V1R7, determine if the SHAREDUP device attribute is being used for the JES2 checkpoint. Select Device Parameter --> Feature Definition from the Device --> OS Configuration Definition panel. If the DASD volume containing the JES checkpoint data set is defined using the SHAREDUP device attribute, you should change the definition. SHAREDUP is an old attribute that applied to physically partitionable systems and VM environments, and avoided the physical reserve. Change the definition from SHAREDUP to SHARED.

References:

z/OS V1R7.0 Hardware Configuration Definition User's Guide, SC33-7988

Invoking the Program Management Binder on z/OS V1R8 and later

Dear Level 2,

I am using the Program Management Binder COMPAT option of COMPAT=CURRENT to specify that my output be defined at the current level of the binder. When I attempt to run it, I receive the following error message. Should I have changed the option to use the default setting of COMPAT=MIN?

```
IEW2509S 3602 MODULE CDAEQED IDENTIFIED BY DDNAME INPUT IS AN
UNSUPPORTED VERSION AND CANNOT BE PROCESSED.
```

ISPF browse fails with the following message if I attempt to browse the PDS/E from a system earlier than z/OS V1R8:

```
IEC036I 002-C8,mod, jjj,sss,ddname[-#], [dev,ser,dsname( member)]
```

Signed: InCOMPATible

Dear InCOMPATible,

If you invoke the Program Management Binder on any release of z/OS and override the Binder default setting of COMPAT=MIN, you produce program objects at the current level of the binder. On z/OS V1R8, this produces a PM5 (new) format module that cannot be run on systems earlier than z/OS V1R8. The program objects can be run only from z/OS releases that understand that format level. This should be OK for programs that are not intended to run on versions earlier than z/OS V1R8.

Advice and recommendations:

If you intend for the programs to be able to run at releases earlier than the one on which they are being built, COMPAT=MIN ensures that program objects are built at the lowest possible program management level, given the program features in use. Thus, they can be run from many more environments. Common places where the COMPAT statement can be specified are in the binder JCL or on the SMP/E GLOBAL zone UTILITY entry for the Program Management Binder.

If the fixes for APARs OA13294 and OA13525 are applied, program objects can be viewed and copied on earlier z/OS releases. Nonetheless, they still cannot be run on releases of z/OS that are earlier than z/OS V1R8.

References:

z/OS V1R8.0 MVS Program Management: User's Guide and Reference, SA22-7643

z/OS V1R8.0 MVS Program Management: Advanced Facilities, SA22-7644 ■

To reorg, or not to reorg?

Knowing when to reorganize a VSAM key sequenced data set

BY STEPHEN BRANCH

When should you reorganize a VSAM key sequenced data set (KSDS)? This often-asked question plagues many VSAM users.

If you think of control interval (CI) splits and control area (CA) splits as bad things, you are probably tempted to reorganize whenever your VSAM KSDS reaches a certain threshold number of CI or CA splits. Is this, however, a valid reason for reorganizing a VSAM? Moreover, does reorganization necessarily improve performance?

This article identifies reasons for reorganizing a VSAM KSDS. We look at some alternative criteria that might help us answer the big question: To reorg, or not to reorg? (Apologies to the Bard.)

Ghostly ISAM rattles its chains

Perhaps memories of the now defunct Indexed Sequential Access Method (ISAM) still influence your thinking that reorganization is needed. ISAM was notorious for having a sequential overflow area, which chained added record entries. As records were added and they went into the overflow area, the chain would grow. As the chain grew, searches took longer and longer. At some point, you needed to reorganize the ISAM data set, to incorporate the records in the overflow area into the prime ISAM area.

Or, you might be thinking of a partitioned data set (PDS), which does not reuse space as members are replaced. When a PDS fills up, you have to compress it, essentially reorganizing it.

Neither of these conditions applies to a VSAM KSDS, which does not use overflow areas and does reuse empty space when possible. VSAM uses CI splits to add records when a CI becomes full and CA splits when the CA is filled. Neither contributes to performance problems.

In short, there is never a valid reason to reorganize a VSAM KSDS based on CI splits.

CA splits are another story. When a VSAM KSDS CA fills up and a new record is to be added, the CA splits; that is, half of the CA records are copied to another empty CA. This leaves two half empty CAs, the one that split and the one that was copied into. The new record can now be added.

To thine own space be true

You can avoid CI and CA splits by providing free space when adding records. Use the FREESPACE parameter, which allows a percentage of free space to be left in each CI or CA when the data set is initially loaded. This percentage is the minimum free area provided in the CI or CA. Free space ensures that when a record is added randomly, there is sufficient space available for the insertion of the new record.

Adding free space to a KSDS can reduce CI and CA splits. It can also cause wasted space. Unless you know that the records will be uniformly added to the VSAM KSDS, you will have some empty space that is never used. Thus, if new records are to be inserted around a certain key range, it is best not to specify FREESPACE. Instead, let the CI and CA splits occur where they are most needed.

Key plot points

A VSAM KSDS is designed around the premise that the range of key values will be uniformly distributed. As one record with a key from a range is deleted, the space can be used for the addition of a new record with a key having a value in that range. Names, phone numbers, and identification numbering schemes are good examples of keys that work well.

Unfortunately, some keys are not uniformly distributed and give the impression that VSAM is wasting space. Most notable of these types of keys are those involving a time stamp or date. Time stamps that do not repeat, like the system clock and dates, by nature increase in value.

An application that adds keys will typically add to the end of the KSDS. Its deletes tend to occur on earlier records causing records to be removed from the beginning of the KSDS. The space available at the beginning of the data set is waiting for records with keys that will never be generated.

There is never a valid reason to reorganize a VSAM KSDS based on CI splits.

Further, if the record added does not have the highest key in the KSDS, and you have not specified the sequential insert strategy (SIS), the CA split to add the key would leave the prior CA half-empty. There is no way to reclaim empty space at the beginning of the data set or the half-empty space in each CA, short of performing reorganization. Using SIS prevents the half-empty space.

Yet needful too

Otherwise, CA splits can be a good thing. They should occur where they are needed, and not always be viewed as a reason to reorganize.

Deleting a large number of records, which causes empty CAs, can also affect the performance of sequential searches. VSAM must read the last CI in the empty CA to discover it has no records. If a number of empty CAs exist, the reads can cause performance problems when issuing sequential or direct key greater-than-or-equal requests. A reorganization is in order to address this situation.

Before z/OS V1R7, VSAM had a limit of 255 extents. As the KSDS approached this limit, we recommended that the data set be reorganized, and the primary space allocation be increased to account for added records. We have seen cases in which reorganizations caused more extents because the primary space allocation was not increased.

With z/OS V1R7 and later, the 255-extent limit is removed for SMS-managed VSAM data sets if Extent Constraint Removal is specified in the data class. The theoretical limit when Extent Constraint Removal is specified is the maximum number of volumes (59), times 123 extents per volume, or 7257 extents. Even so, you should consider reorganizing and increasing the primary space allocation long before nearing the 7257 extent limit. The number of candidate volumes available to the KSDS might be the more important limiting factor of the KSDS rather than the number of extents.

Catalogs are special VSAM KSDSs that do not extend to new volumes; therefore, their extent limit is 123 extents. Catalog Management produces a warning message when the number of extents for a catalog reaches 80% of this limit. The threshold percentage can be changed from the 80% default value using MODIFY CATALOG NOTIFYEXTENT. We recommend not changing the default and reorganizing the catalog as soon as you can.

A more considered time

So when should you reorganize a VSAM KSDS? Here are some times when it's a good idea:

- If you have a key that is increasing, like one that uses a time stamp, and you are deleting records with lower key values, reorganize the KSDS to reclaim unused space.
- If your data set is approaching an extent limit, 123 for catalogs or 255 for a VSAM KSDS, reorganize the KSDS and increase the primary space allocation. You might want to increase the secondary allocation, too.
- If you have specified Extent Constraint Removal, you might want to determine a limit for the number of extents allowed for your installation, or the maximum number of candidate volumes for the KSDS. If you approach either of those limits, reorganize the KSDS and consider increasing the primary and secondary allocation.
- If you have a catalog for which the 80% warning message is issued. Again, consider increasing the primary and secondary space allocations for the catalog at this time.

Brevity is the soul of splits

For CA splits, consider reorganizing the VSAM KSDS only if you have a non-random record key.

For CI splits, you can skip the reorganization altogether. ■

HOW TO invoke SDSF function with REXX

BY GREG THOMPSON

As you know, a good way to find out about the jobs running in your system is to extract job-related data from the System Display and Search Facility (SDSF). You begin this process, ironically enough, by submitting a batch job — on SDSF.

When you retrieve job data through batch SDSF, you see that the data is formatted to resemble the SDSF end user screen. Any further work you might want to do with the data, such as creating a meaningful usage report for your boss, entails parsing the output in some way, perhaps manually.

In z/OS V1R9, you can access SDSF data directly through a REXX program (an exec). To help you do so, SDSF has added new commands and support for REXX variables.

New SDSF functions

In z/OS V1R9, SDSF adds these new commands and variables:

- ISFCALLS for setting up the SDSF host command environment
- ISFEXEC for running SDSF commands, such as those that access SDSF panels
- ISFACT for entering action characters and overtyping columns
- Special REXX variables that provide functions equivalent to other SDSF commands and help you to access messages and table data.

Getting started

To use the new REXX support, you must start the SDSF host command environment in REXX. Enter the new ISFCALLS command as follows:

```
rc=isfcalls('ON')
```

To remove the host command environment, enter the same command, but with 'OFF' instead of 'ON.'

Accessing SDSF panels

To access an SDSF panel from your program, use the ISFEXEC command. When you do so, SDSF creates REXX stem variables for each row and column on that panel, which your program can query. The column names are not the titles, but rather, are the names that you would use to create a field list in ISFPARMS. (If you are new to SDSF, ISFPARMS is an installation-editable module in SDSF that defines global options and panel formats for SDSF, including the order and titles of the columns.)

For example, the ISFEXEC command allows your program to access the SDSF status display as follows:

```
Address SDSF "ISFEXEC ST"
```

Now suppose that you want your program to query the status of a particular job. If you know the job name and job ID, you can code a loop in your program to determine the job status:

```
Do x = 1 to JNAME.0
/* Loop all rows returned */
 If JNAME.x = "MYJOB01" &
JOBID.x = "J0000123" then
```

With the job status obtained, your program can then interrogate more SDSF fields. For example, to determine which queue the job is in, your program could set a variable to store the queue name, as follows:

```
JOBQUEUE = QUEUE.X
```

Limiting the result size

The new SDSF commands produce data in the form of stem variables. To reduce the amount of storage used by REXX for these operations, your program can use filters to access only the data you need.

In the previous example, you can filter the results by job name. To do so, include the ISFPREFIX filter, as follows:

```
ISFPREFIX = "MYJOB01"
```

Other filters you might find useful are:

- ISFDEST for filtering by up to four job destinations
- ISFOWNER for filtering by job owner
- ISFSYSNAME for filtering by system name
- ISFINPUT for specifying whether SYSIN data sets should be included in the results.

You can also limit the size of the results by using the ISFCOLS variable, which specifies which columns to return. (More on this idea shortly.)

To control the results in other ways, try issuing the commands with these optional parameters:

- ALTERNATE for requesting the alternate field list
- DELAYED for including the delayed-access columns
- NOMODIFY for specifying that row tokens used in modifying rows are not returned
- PREFIX for specifying a prefix for column names and TOKEN variables that are created
- VERBOSE for adding diagnostic messages to the ISFMSG2 stem variable. The messages describe each row variable that SDSF creates.

For example, to access the delayed columns in the alternate field list and use a unique prefix for the results, your program could issue the following commands:

```
ISFPREFIX = "MYJOB01"
/* Return only this job name */
ISFCOLS = "JOBID QUEUE"
/* Return only JNAME, JOBID, &
QUEUE */
Address SDSF "ISFEXEC ST"
PREFIX('PREF1_') (ALTERNATE
DELAYED)

Do x = 1 to PREF1_JNAME.0
/* If more than 1 job, check job
nbr */
  If PREF1_JOBID.x = "J0000123"
  then
```

The fixed field (the first column on the panel when it is used interactively) is always included in ISFCOLS output. Therefore, in this example, JNAME is always included.

By the way, avoid using variable names that begin with ISF. SDSF reserves that prefix for names of special REXX variables.

Using action and modify commands

The ISFACT command is required when using action characters or modifying SDSF data. On the ISFACT command, include the same panel command that you used on the ISFEXEC command, followed by a token that identifies the row, followed by parameters that describe the column and action.

You can access a job's output by using this command and specifying the 'SA' parameter in the NP action column. The 'SA' parameter dynamically allocates all data sets for a job. Use the SDSF stem variable ISFDDNAME to access the data.

Another stem variable, ISFDSNAME, contains the data set name on a one-to-one basis with ISFDDNAME. Then you can read the data with EXECIO to your own REXX stem variables, as follows:

```
Address SDSF "ISFACT ST
TOKEN('"token.x"') PARM(NP SA)"
Do jx=1 to isfddname.0
   If isfddname.jx = "SYSUT2"
   then
   /* read sysut2 */
     "EXECIO * DISKR"
     isfddname.jx "(STEM line.
     FINIS)"
End
```

Note that the token definition in the ISFACT command includes single quotation marks around it. The quotation marks are required.

The PARM field can contain a list. For example, if you want to alter the SYSOUT class to 'A' and the forms of a job to '1234', you can specify in the PARM list each column of the output command, followed by its new value:

```
Address SDSF "ISFACT O
TOKEN('"token.3"') PARM(OCLASS A
FORMS 1234)"
```

More useful functions

You might also find these special variables for panels and panel commands useful:

- ISFACTIONS, which specifies whether the action characters for the current panel should be returned in the ISFRESP stem variable.
- ISFCOLS, which specifies the set of columns that SDSF is to create as variables.
- ISFDCOLS, which specifies the set of delayed-access columns for which SDSF is to create variables.
- ISFROWS, which contains the number of rows created for a tabular panel. (This value is also found in the zero stem of the column variables; for example, JNAME.0.)
- ISFSORT, which specifies the sort criteria.
- ISFUCOLS, which contains the list of modifiable columns for the panel.

Using the slash command

The slash ('/') command used with ISFEXEC allows you to enter system commands.

You can use these options with the slash command:

- INTERNAL, which specifies that console ID zero (0) should be used.
- WAIT, which specifies that SDSF should wait the full delay interval before retrieving the response.

For example:

```
Address SDSF "ISFEXEC
'/$DA'(WAIT"
```

Use the following REXX variables with the slash command:

- ISFCONS, which specifies the console name for the user session log
- ISFDELAY, which specifies the response delay limit for system commands
- ISFULOG, which logs the MVS system commands used and any responses generated during the session.

Handling problems

Each ISFEXEC and ISFACT command sets a return code in REXX variable RC. It is a good practice to have your program check the return code from each command.

In addition, your program can check for any messages that might be issued in response to a command or variable in the SDSF message variables ISFMSG and ISFMSG2.

Lastly, SDSF provides two variables for SDSF trace. They are:

- ISFTRACE, which specifies that a trace option should be used when enabling SDSF trace
- ISFTRMASK, which specifies to use a trace mask when enabling SDSF trace.

Be sure to include an ISFTRACE DD statement when using these trace functions.

Information at the ready

For more information about using REXX with SDSF, see the SDSF online help. Just enter REXXHELP from any command line in SDSF to access it.

The online help has links to commands, action characters, and columns that you can type over. Similarly, you can enter the command COLSHELP for a description of the column names you need for REXX.

Information about using REXX with SDSF is also provided in the publication, *SDSF Operation and Customization*, SA22-7670. ■

Don't go mental, go metal!

Using the METAL option in the XL C compiler

BY CHWAN-HANG LEE, KENDRICK WONG, AND RAYMOND MAK

Assembly language is powerful. It's close to the underlying hardware architecture, it allows you to access low-level system features that interact closely with the operating system, and it has no runtime dependency. In short, it allows you to do almost anything you want.

Nevertheless, assembly language forces you to spell out all the minor details in the code — you need to focus on the leaves and trees, not the forest and the landscape. You spend time on the low-level details and not the high-level logic — hand holding the underlying machine every step of the way. It's tedious. It takes time to write and debug, and the resulting code can be difficult to maintain.

Wouldn't it be nice if you could use a high-level language to do low-level programming? The design for the METAL option of the z/OS V1R9 XL C compiler kept this in mind. The concept of metal is that the code generated can interoperate with code inside of the operating system allowing it to be close to the hardware — the metal. The METAL option gives you the ability to use C language features to express low-level programming logic, for example, writing user exits. You are free from tasks such as managing the registers and developing the correct instruction sequences. Let's look at what the METAL option can do.

Close to the metal

The XL C compiler generated code requires the Language Environment® to establish an overall execution context. The C library functions also require the Language Environment in order to provide their services, most noticeably the functions to manage the heap storage and dynamic storage area, to do file input-output and to handle exception conditions. If your C program needs to get close to the metal, it needs the means to acquire these services directly from the operating system. Because you can only acquire these using assembler macros, the compiler needs to incorporate your assembler statements in the C program. The compiler also needs to make the C program follow conventions expected in the operating system environment.

As a first step, the METAL option generates code in assembly source program format. You can then feed the assembly source through the High Level Assembler, like any other assembly programs. The resulting code is independent of the Language Environment. You can write the following statement, for example, if (x + y * z > 0) x = -x, and the XL C compiler can turn it into assembly instructions. You work on the high-level logic, and the compiler generates the assembly code. The code is close to the metal.

Keep things local

One reason why a normal C program requires support from the Language Environment is the requirement of a stack, on which local variables reside. Normal C code relies on the runtime to manage this stack. When doing low-level programming, this could be a roadblock because you might have your own storage management scheme.

METAL C gives you control of stack allocation by letting you customize the function prolog and epilog code. This is code generated by the compiler at the beginning and end of a function to acquire sufficient storage for local variables. METAL C provides a default code sequence; but you can override it by providing your own. You can allocate the stack according to your own scheme.

Protect your investment

What happens to the existing library of high-level assembly code? These are often the result of years of cumulative work, encapsulating the knowledge and experience of many skilled programmers. You don't have to throw them away. METAL C allows you to tap into these treasures.

Function calls and argument passing in METAL C follow the MVS linkage convention. For example, when you call a function from C, the compiler sets up the arguments so that register 1 points to the parameter list, as you would when calling an assembly routine. When coding the C call statement, you just treat the assembly routine like any other C functions — the C programs and your assembly programs are interoperable. You are now poised to enhance your collection of assembly programs with new additions written in C.

Find the pearls

Of course, it isn't just the assembly routines that we need to worry about. The assembler macros contain the pearls of low-level services. There's a large body of system services that are available only as assembler macros — you might even have a few of your own. This is the good stuff! But you can only get to it through assembly code, so we need a way to open the clamshell and get the pearl.

No problem! METAL C has an opener! You can embed a short sequence of assembly instructions inside a C function, in between other C statements. You can also reference the values in C variables from these assembly instructions. For example, within a C function, you can:

- Set up the parameter values for the STORAGE macro
- Call the STORAGE macro
- Return the obtained storage address in a C variable.

You can do all this within a C function! You can even develop a C library to get at the system services, and design a high-level application interface tailoring for your purpose.

The key is in the register

Sometimes an application written in assembly code might put aside a certain register for a special purpose throughout the application (possibly to hold the address of a key data block for efficiency). Of course, you can use embedded assembly instructions to access the data block. However, METAL C provides an even easier way! You can tie a register to a pointer variable. The address value in the C pointer is the address stored in the register. You can declare a C structure representing the layout of the data block, and then access the data through a pointer variable in C expressions with no assembly code.

Reach far

Speaking of pointers and data blocks, what if you need to manipulate a very large table of data and you are not ready for AMODE 64? An application written in assembly code can use data spaces to do it. METAL C allows you to access data spaces through far pointers. These are C pointer variables but wider — the lower half is an address within the data space; the upper half is the access list entry token (ALET). You can use a far pointer like a normal pointer; you can reference it, increment/decrement it, compare it with other far pointers to the same data space, and more. You can write a sort routine in C, for example, to sort the data, and then call this C routine from existing assembly code. In short, you can now use a high-level language to exploit the advantage of data spaces.

What about printf()?

Just like other C library functions, *printf()* needs the Language Environment. Because we want the code to be independent of the Language Environment, we don't have the full C library. Nevertheless, not all is lost; we still provide a useful subset of these functions in the METAL C library. Be sure to see the companion article for more details ("Practice alchemy with z/OS Metal C" on page 69). We cannot provide normal input-output functionality, but we can still provide functions like *sprintf()*. Not *printf()*, but rather *sprintf()*, which is equally useful. You can still enjoy the convenience of %- substitution in the format string; and you can place the resulting string in a buffer, and then print this buffer using, for example, a WTO message.

Putting it all together

All this information represents some significant additions to the XL C compiler, which are available in the z/OS V1R9. You can use these features in a number of ways. As indicated, METAL C does not require you to rewrite your existing assembly code. METAL C code is meant to interoperate with existing assembly routines and work without the Language Environment. You can use METAL C when adding new features to your existing application written in assembly language. This protects your investment in existing code, and allows you to leverage it using a programming tool that can better express the high-level logic. If you like, you can rewrite selected routines in C, for example, code that requires frequent maintenance. In addition, you can write user exits in C. You have a power tool to work close to the iron, and to top it all off, you can enjoy the capabilities of the XL C compiler to optimize your programs tailored to the latest hardware you have.

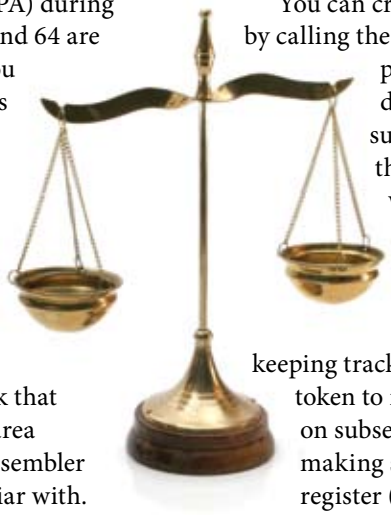
For more information on the METAL compiler option and related features, see *z/OS V1R9.0 Metal C Programming Guide and Reference,* SA23-2225 and *z/OS XL C/C++ User's Guide,* SC09-4767. ■

Practice alchemy with z/OS Metal C

BY BARBARA NEUMANN AND TOM PETROLINO

Practice some alchemy in your programs! Can you turn metal into gold?

Although the intent of the Metal C support is to eliminate the need for a runtime library and environment, we recognize that C programmers have become accustomed to calling basic functions that could be useful even in a system environment. We created the Metal C Runtime Library so that Metal C users would not have to re-invent these functions. This runtime library is a new base element of z/OS, and is completely independent of Language Environment. The library modules are made available in the link pack area (LPA) during IPL. Both AMODE 31 and 64 are supported, as long as you are calling the functions in Primary address space control (ASC) mode. The library functions make use of the default linkage provided by the compiler's METAL option, which requires a small contiguous stack that uses the standard save area convention that z/OS assembler programmers are familiar with.



We have included a starter set of functions that would come in handy in a system environment. This includes functions to:

- Test and manipulate memory and strings like memcpy() and strcmp()
- Classify characters such as isalpha() and islowerj()
- Process text strings using the sprintf() and sscanf() family of functions.

You can even generate a random number or two, using the rand() and rand_r() functions. And if you are used to having heap management functions like malloc() and free() available, the Metal C Runtime Library supplies these as well. There is even a __malloc31() function to obtain below-the-bar storage while running AMODE 64.

However, unlike many of the other functions already mentioned, the heap management functions require the runtime to remember information about the heaps. To accomplish this, the Metal C Runtime Library introduces the concept of a Metal C environment. This environment is used to anchor heap storage and to keep track of other persistent data, such as the random number generator seed.

You can create a Metal C environment by calling the __cinit() function, and passing it information that describes the environment, such as the subpool and size of the heap segments. __cinit() will set up the environment and return to you a token that represents the environment. Your program is responsible for keeping track of this token. You use the token to reference the environment on subsequent library calls by making sure that general-purpose register (GPR) 12 contains the token when the calls are made. When the environment is no longer needed, you can call the __cterm() function to perform cleanup, freeing all resources that had been obtained when using the token. For more information to try out your new metal-working skills, see *z/OS Metal C Programming Guide and Reference*, SA23-2225, and **ibm.com/systems/z/zos/metalc/.** ■

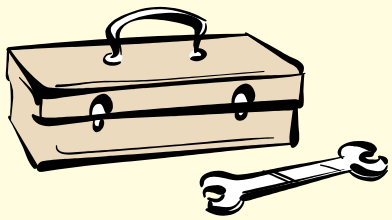
Some tools and toys for z/OS UNIX System Services

BY ANTHONY GIORGIO

Do you know about the z/OS Tools and Toys for UNIX System Services Web site? It's the home of a variety of software for z/OS UNIX. You can find a wide assortment of tools and utilities, some written by IBM developers and some ported from other platforms. A number of popular open-source utilities are available for download. Look for tools like:

- GNU bash - the bourne again shell, which is compatible with /bin/sh and offers many extensions found in csh and ksh.
- GNU grep - a tool to help you search files for patterns. It includes extensions to search data sets.
- gzip - a program that compresses and decompresses files.
- gmake - a tool that controls the generation of executables from a program's source files.
- bzip2 - a data compression utility that can often compress files to be half the size of the files the compress utility produces.
- zip and unzip - a set of programs that work with compressed zip files.
- less - a pager similar to the more command.

There is no official IBM support available for any of this software, but it can still make a useful addition to your z/OS UNIX portfolio. Find the Tools and toys site here: **ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html**



Wake up your system! New enhancements and products for Java on z/OS

BY NORMAN AARONSON

While it's true that the products in the Java software developer kit (SDK) for z/OS are based on the same technology and contain the same function as the other IBM SDKs, they also contain unique value for z/OS. Delivered without charge, these products have passed the Java compatible test suites and are available in both SMP/E and non-SMP/E formats.

The IBM JZOS Batch Toolkit for z/OS SDK is a set of tools that addresses many of the functional and environmental shortcomings in current Java batch capabilities on z/OS.

This article briefly describes the Java SDK products for z/OS and exciting recent enhancements for batch processing and offers some tips for application migration.

JZOS! Recently integrated into the z/OS Java products!

z/OS Java continues to be a mainstay of WebSphere Application Server for z/OS V6.0 and V6.1. But, we've also recently included in the product customer-driven improvements for applications that do not run under WebSphere.

The IBM JZOS Batch Toolkit for z/OS SDK is a set of tools that addresses many of the functional and environmental shortcomings in current Java batch capabilities on z/OS. It includes a native launcher for running Java applications directly as batch jobs or started tasks, and a set of Java methods that make access to traditional z/OS data and key system services directly available from Java applications.

The combination of the launcher, data access, added system services, and environmental enhancements makes running Java on z/OS as batch jobs easier, particularly for traditional z/OS programmers. The net result of these enhancements is that the look and feel of running Java applications is much closer to other z/OS batch jobs, and the way batch jobs for Java can be managed is now like other z/OS batch applications written in COBOL, PL/I, or other compiled languages.

For an overview, see **ibm.com/servers/eserver/zseries/software/java/jzos/overview.html**.

The products! A summary

Here's a brief summary of the IBM SDKs for z/OS.

IBM SDK for z/OS, Java 2 Technology Edition, Version 1.4 (5655-I56)

This 31-bit product is at the SDK1.4.2 level and is intended to be especially useful for Java applications that run on the System z Application Assist Processor (zAAP). Enhanced now with JZOS batch technology and application programming interfaces, this product will be in service through September 2009. WebSphere Application Server V5.1 and V6.0 use SDK1.4.2.

IBM 64-Bit SDK for z/OS, Java 2 Technology Edition, Version 1.4 (5655-M30)

Applications previously constrained by 31-bit addressing limitations can break that barrier with this 64-bit SDK. However, we strongly recommend that any customer considering 64-bit z/OS Java SDKs use the later, more-functional 64-bit SDK5 product. The older 64-bit SDK1.4.2 product is planned to be withdrawn from marketing in September 2007 and from service in September 2008.

IBM 31-bit SDK for z/OS, Java 2 Technology Edition, Version 5 (5655-N98)

This SDK5-level product contains the latest JZOS batch technology and application programming interfaces, is particularly useful on zAAP processors, and is imbedded in WebSphere Application Server for z/OS V6.1.

IBM 64-Bit SDK for z/OS, Java 2 Technology Edition, Version 5 (5655-N99)

Applications previously constrained by 31-bit addressing limitations can break that barrier with this 64-bit SDK, plus take advantage of zAAP processing and the JZOS batch enhancements and APIs. If you want a 64-bit z/OS SDK,

this is the one to use. See the following Web site for when to use it: **ibm.com/servers/eserver/zseries/software/java/j5pcont64.html#j5position**.

Migrating from one z/OS Java SDK product to another

z/OS Java SDKs are totally independent of each other. You can order and concurrently run one, two, or all in your installation, and each can be serviced separately. There are, however, some considerations for migrating your Java applications, as described in the sections that follow.

Compatibility considerations

Most SDK1.3 Java applications can run on SDK1.4 SDKs and most SDK1.4 applications can run on SDK5. Some applications, however, cannot. See the following Web sites for compatibility considerations:

- **http://java.sun.com/j2se/1.4.2/compatibility.html**
- **http://java.sun.com/j2se/1.5.0/compatibility.html**

Deprecated application programming interfaces

A deprecated application programming interface (API) might still be supported, but it is no longer recommended and might soon become obsolete. Deprecated application programming interfaces are usually replaced with improved interfaces — use those instead.

Should you care? Yes. There is no guarantee that a future version of Java will contain a deprecated API. In fact, deprecated application programming interfaces might not work the same way as in previous releases.

Java applications using deprecated APIs will receive a warning. Don't ignore it! Determine whether to change your application now or when the API disappears in a future release.

You can find deprecated application programming interfaces listed on the following Web sites for Java:

- **http://java.sun.com/j2se/1.4.2/docs/api/**
- **http://java.sun.com/j2se/1.5.0/docs/api/deprecated-list.html**

z/OS considerations

For considerations unique to z/OS Java, see the product content pages on the z/OS Java Web site at **ibm.com/servers/eserver/zseries/software/java/**.

Want to get a head start on Java 6?

To prepare for Java 6, review the Java 5 compatibility guide at: **http://java.sun.com/javase/6/webnotes/compatibility.html#incompatibilities** and review the deprecated application programming interfaces at: **http://java.sun.com/javase/6/docs/api/deprecated-list.html** ■

Options, options, options!

XML parsing options for z/OS

BY BILL CAREY

As the need to process XML data increases, you should understand the options available for handling this data format. On z/OS, a number of parsing choices are readily available. For Java the primary choice is the XML4J parser, which is available in the Java SDK, along with the corresponding XSLT processor, XSLT4J.

In the non-Java programming environments, there are a number of alternatives, notably:

- Native built-in programming-language support exists in the Enterprise COBOL and PL/I products
- The XML Toolkit provides an Apache Xerces-based parser called XML4C and a corresponding XSLT Processor called XSLT4C
- XML System Services (z/OS XML) provides a new base system component (for z/OS 1.7 onward) for XML parsing.

These alternatives have various characteristics in terms of functionality, standards support, and performance that should be considered when you are choosing a parser to be used for a specific situation. Good sources of information about these options are available at the following Web sites:

- XML Toolkit for z/OS: **ibm.com/servers/eserver/zseries/software/xml/**
- z/OS XML System Services: **ibm.com/servers/eserver/zseries/zos/xml/**
- Enterprise COBOL for z/OS: **ibm.com/software/awdtools/cobol/zos/**
- Enterprise PL/I for z/OS: **ibm.com/software/awdtools/pli/plizos/about/**

One point that you should be aware of is the ability to use the XML Toolkit within CICS. With the advent of CICS TS V3.1 and its enhanced Open Transaction Environment support, the XML Toolkit is now supported in CICS for OPENAPI programs. In addition, you should be aware of the ability to utilize the XML Toolkit from COBOL and PL/I programs when the built-in XML parsing support does not meet your needs. For example, if you need to perform schema validation of the XML data or want to utilize the XSLT4C stylesheet processor to perform a transformation, you might want to consider this option. You validate the data by writing some intermediary C++ code that transitions between the COBOL or PL/I application and the XML Toolkit interfaces. See the *XML Toolkit for z/OS V1.9 User's Guide*, SA22-7932, for an example illustrating this.

Also, keep in mind that often there is higher-level processing available that removes the burden of you needing to parse XML data yourself. For example, you can use XML handling that is provided by products such as WebSphere Application Server, WebSphere Message Broker, and WebSphere Transformation Extender, to name a few. Powerful XML processing capabilities are also provided using PureXML in DB2 Version 9.1 for z/OS. ■

Demystifying COMPAT option sub-levels

BY RITU BHARGAVA AND BARRY LICHTENSTEIN

Using the COMPAT option, you can specify the compatibility level that the binder uses when binding (link-editing) programs. The main purpose is to tell the binder to generate a program object (PO) at a particular program-management (PM) level. The COMPAT option is analogous to the XLC C/C++ TARGET option, which provides compatibility with earlier versions of Language Environment C/C++ programs. Like the C/C++ TARGET option, you can use the binder COMPAT option to ensure the program you are link-editing can run on a specific release of z/OS.

You can specify the COMPAT level in terms of the z/OS system release, for example, COMPAT=ZOSV1R9, rather than COMPAT=PM5. The program being link-edited can load and run on the specified z/OS system release and any higher release by making sure that it does not exploit any features or functions that would require a newer release of the loader. If you are creating a program that needs to run (target) on a system that's not at the latest level from the system you are building the program, use the binder COMPAT option during the link-edit. Therefore, if you are building a C/C++ program, you can use the C/C++ TARGET option during compile specifying the same target level.

When you specify a COMPAT level in terms of the z/OS system release, it corresponds to a specific program-management level and program management sub-level (such as PM4SUB3). For best results, specify the z/OS system. It's also the way that you can request a specific program management sub-level. In other words, while you can use COMPAT=PM4 instead of COMPAT=ZOSV1R3, using COMPAT=PM4SUB3 results in an error; you must use COMPAT=ZOSV1R7 instead. Table 1 shows the corresponding z/OS system releases and program management levels.

z/OS system release	Program management level + sub-level	Summary of features added
z/OS V1R9 z/OS V1R8	PM5	• Cross-segment references in relative immediate instructions in program objects
z/OS V1R7	PM4SUB3	• Relative/immediate instructions across compile units or compression of non-program data
z/OS V1R6 z/OS V1R5	PM4SUB2	• RMODE 64 has been specified by a compiler for deferred load data segments (such as XL C/C++ compiled with LP64,RENT)
z/OS V1R4 z/OS V1R3	PM4	• Input modules contain 8-byte adcons • Any ESD record is AMODE 64 • Input contains symbol names longer than 1024, unless EDIT=NO • A value of 64 is specified on the AMODE option or control statement
z/OS V1R1 z/OS V1R2	PM3	• Binding modules compiled using the XPLINK attribute • DYNAM=DLL • XOBJ format input to the binder without going through the Language Environment prelinker, or rebinding modules containing input from such sources • Hidden aliases (from ALIASES control statement) • Support for deferred classes or initialized text in merge classes in GOFF format input modules or data buffers passed via the binder API.
OS/390® V2R10	PM2	• User-defined classes passed in GOFF format input as well as certain other information supported only in GOFF format • Names (from input modules or created by control statements which cause renaming) that are longer than 8 bytes • Use of RMODE=SPLIT • Device-independent record format • Text length greater than 16 megabytes • More than 32,767 external names • OVLY is supported and forces PM1 to be used.

Table 1. z/OS releases with associated program management levels

Clarifying sub-levels

In a case where the program management levels are identical and only the program management sub-levels differ, this indicates that there is new binder functionality; however, none of the program object information that the loader uses is incompatible (only information that the binder uses for rebinding, diagnostics, and so forth is different). Therefore, when considering a COMPAT level, you can use just the program management level without regard for the sub-level to determine the earliest release on which to use it.

For example, if you target z/OS V1R7 with COMPAT=ZOSV1R7, you create a PM4SUB3 format program object. Then, in addition to z/OS V1R7, that program object can be run on releases all the way down to z/OS V1R3 because that was the first release to introduce program management level PM4 format (effectively, this is level 4, sub-level 1, or COMPAT level PM4SUB1 format — but the binder does not use the SUB1 designation).

You still need to be aware of the consequences of the program management sub-level when it indicates that there is new binder function because this means that you cannot do anything besides run the program.

For example, a format program object that is at COMPAT level PM4SUB3, contains information that only a binder at z/OS V1R7 and above can understand. While the program can run based solely on the program management level, without regard for the sub-level, it cannot be rebound with a binder (the IEWL program, the TSO LINK command, or the UNIX ld utility) at a lower release program management level and sub-level. No other programs running on a lower release z/OS system than that program can use the Program Management Binder services against it. The services include:

- IEBCOPY
- SMP/E
- SPZAP
- Any programs that use the binder APIs (such as the AMBLIST, IEBCOPY, and UNIX utilities cp, mv, c89, and nm)
- The fast data APIs
- The binder C/C++ API interfaces introduced in z/OS V1R9.

Any attempt to do so by a properly coded program fails with the following message:

```
IEW2509S 3602 MODULE *NULL*
IDENTIFIED BY DDNAME /0000001
IS AN UNSUPPORTED VERSION AND
CANNOT BE PROCESSED.
```

Explaining COMPAT options

If you are building your application to run on only the z/OS release on which you are building it, the best choice is the binder default, COMPAT=MIN. This builds your program object so that it can run on the widest range of z/OS releases. If you intend to run on a lower release z/OS system, then use COMPAT to specify that lower release. In fact, you might end up building a higher-level program object than you would have built with COMPAT=MIN. The advantage: the binder verifies that what it produces can run on that lower z/OS release.

If you introduced code that exploits new program object features, COMPAT=MIN forces you to a higher program-management level than you might have expected. For example, this situation happens if you bound in a C/C++ part that was not built with the appropriate TARGET option (a functional option or the ARCHITECTURE option that it implies), and then used some new capability. The options LP64, RENT, are

```
PROCESSING OPTIONS:
  . . .
  COMPAT                PM4SUB3
. . .
SAVE OPERATION SUMMARY:
  . . .
  PROGRAM TYPE          PROGRAM OBJECT(FORMAT 4 OS COMPAT LEVEL z/OS V1R7)


*****  M O D U L E   S U M M A R Y  *****
. . .
. . .
-----------------------------------------------------

                     PO FORMAT:PM4
                     OS COMPAT LEVEL:SUB3

. . .
          *****  PROGRAM OBJECT PROCESSED BY BINDER
```

❑Program Management Level

❑Sub-level

Figure 1. Binder processing options and save operation summary and the AMBLIST module summary

available with z/OS V1R5 and later. So when you specify COMPAT=MIN with code compiled using the LP64, RENT options, it minimally produces a program object with FORMAT 4 OS COMPAT-LEVEL z/OS V1R5 (PM4SUB2). If you attempted to use a lower level program object like COMPAT=ZOSV1R4, the binder fails with a message like the following:

```
IEW2606S 5393 MODULE
INCORPORATES Z/OS V1R5 PROGRAM
OBJECT FEATURES AND CANNOT BE
SAVED IN A Z/OS V1R3 COMPATIBLE
PROGRAM OBJECT FORMAT.
```

Both the program management level and sub-level are in the binder (IEWL, c89) listing and the AMBLIST utility LISTLOAD output.

The Binder processing options and save operation summary are parts of the operation summary information that writes to SYSPRINT when using the LIST option. The AMBLIST module summary writes to SYSPRINT for all control statements except with LISTOBJ, LISTLPA, or MODLIB with LISTIDR. Figure 1 shows an example of both of these summaries.

Understanding PM levels

Whenever IBM produces a new program-management level, we must also include support for previous supported releases. This is because some programs (such as ISPF and IEBCOPY) are sensitive to the information in the directory, which might change between program object formats. For example, with the introduction of COMPAT=ZOSV1R5 (PM5), came PTFs for APARs OA13294 and OA13525. These PTFs do not allow COMPAT=ZOSV1R5 program objects to run on a system before z/OS V1R8. Instead, they provide the ability to copy, list, and browse the program objects on those earlier releases.

We hope this article helps demystify the binder. To find out more, see *z/OS V1R9 Migration*, GA22-7499-11 and *z/OS MVS Program Management: User's Guide and Reference*, SA22-7643. ■

Show your school spirit and promote Large Systems Thinking

BY KATHY PFEIFFER AND DON RESNIK

The IBM Academic Initiative System z team has developed a robust program to build mainframe skills for the 21st century. IBM provides colleges and universities worldwide with educational resources at no cost (course materials, textbooks, professor education, remote system access to mainframes) to build skills. Today, over 300 schools are participating in the program; that's up from 24 schools in 2003.

However, preparing a new generation of professors and students educated in Large Systems Thinking requires assistance and involvement from the entire mainframe community. IBM welcomes clients, Business Partners, independent software vendors (ISVs), and everyone in the mainframe community to help spread the message — Teach Large Systems Thinking! Our team is often asked, "Can I get involved in this skills-building initiative?" Our response is always a resounding, "Yes!" The more people who participate in this initiative to build mainframe skills, the more influence we'll have with educators and the better for the mainframe community.

As a member of the mainframe community, you play an important role in helping to influence educators and building future skills. Here are some ways you can get involved.

1. Select a school: Identify a school to approach about teaching mainframe technologies. This can be a school that is located in your geographic area, one you are a graduate of, or one you recruit from. See the **ibm.com/university/systemz** for a list of schools that already participate in the program.

2. Contact the IBM Academic Initiative team: Send an e-mail to: **zskills@us.ibm.com**. We'll work with you to discuss different alternatives for moving ahead.

Reach out

The Academic Initiative works with the mainframe community to support individuals and groups who want to reach out to a school. In some cases, we provide you the materials and you contact the educator. In other cases, our team can join you on a call with an educator. Another alternative is a roundtable on campus. Roundtable events bring faculty, IBM, and the System z community together, face-to-face, to discuss the concern about skills that are needed in the industry. At roundtable events, we do the following activities:

- Introduce Large Systems Thinking. (Why teach mainframe technologies?)
- Validate the demand for students educated in Large Systems Thinking.
- Show faculty how to leverage the IBM Academic Initiative program and its resources.
- Provide educators with industry input for Large Systems education programs.

Roundtable events provide an opportunity for you to openly discuss the demand for mainframe skills in your industry. We've held many of these events, and they've been very successful. This demand for students educated on the mainframe is what faculty wants to hear. Roundtable discussions validate and help educators understand skills requirements and influence schools to define and offer courses and programs that are responsive to industry needs.

Influencing educators is important; however, they are only a part of the equation. Influencing students is important too. Here are some ways you can help influence students who are considering studying enterprise computing.

Offer internships or co-op opportunities

These opportunities provide an incentive for students to take the courses and provide future employers with an early look at potential new large systems employees.

Be a guest lecturer or adjunct professor

Visit the campus and meet with students to build interest in pursuing a career in enterprise technologies.

Host a Student Day

Invite students to visit your company. Have them meet with key technical people to talk about opportunities within your company. Provide a tour of your location.

As you can see, there are many ways you can contribute to this initiative. Together, we can make a difference. Now, run to your closet, show your school spirit, and pull out that college jersey! ■

IBM Destination z

In June, IBM announced the formation of "IBM Destination z", a new community of IBM Business Partners, companies, and educational institutions. IBM Destination z promotes working together with these communities to deliver strong business value to new or current System z customers, such as yourselves. Check out the new portal Web site at: **ibm.com/systems/destinationz**. It offers a focal point to-provide quick and efficient access to IBM and Business Partner System z resources, solutions, and support that will help create synergies and maximize member effectiveness.

Got Comments?

Let us know what you think about z/OS Hot Topics. Send your comments to the following address:

newsletr@us.ibm.com

Our contributors

Norm Aaronson, an IBM Senior Engineer, is the Release and Project Manager for Java on zSeries. His 33 years with IBM includes projects in VTAM, Cornell Supercomputing, VM, RS/6000®, and IEEE Standards.

E. Ozan (Oz) Baran has been with IBM since January 2004. He is a software engineer in z/OS Integration Test.

Arthur Bariska is an ASTQB Certified Test Professional at IBM Tucson. For more than half of his 32 years with IBM, he has worked in System Verification Test for DFSMS.

Harald Bender has worked with z/OS and earlier operating systems for more than 15 years. He leads the architecture and ongoing development of Resource Measurement Facility (RMF).

Gita Berg is the z/OS Marketing Manager. Her ten years of mainframe experience span sales and sales support, product launches, partner enablement, and product marketing.

Ritu Bhargava works in z/OS Level 2 defect support. She specializes in z/OS Language Environment, the C run-time, and Program Management Binder.

R. David Boenig is the service team lead for DFSORT Development. Previously, he was a Level 2 representative for DFSMS products, specializing in DFSMSrmm and DFSORT.

Jessica P. Bonner, a Staff Software Engineer at IBM Poughkeepsie, has worked on ICSF and TKE for the z/OS Cryptography Team. Currently, she leads the TKE Function Test Team.

Stephen Branch is a Senior Software Engineer in DFSMSdfp™ Development. He has been with IBM for 24 years.

Bill Carey is a Senior Software Engineer in System z Software Development. His current focus is on XML technology strategy and design. Bill has been with IBM for 34 years.

Jeff Cates is a Senior Software Engineer in z/OS Communications Server Development at IBM Raleigh. In his 26 years with IBM, he has helped to develop a variety of software products.

Ross Cooper, a Software Engineer, has been with IBM Poughkeepsie for eight years. His projects have included RACF, EIM, and PKI.

Erika Dawson is an IBM Senior Technical Staff Member (STSM) in DFSMS Development at IBM Tucson. Most of her 20-year career has involved software support for IBM tape products.

Noshir Dhondy, an IBM Senior Engineer, works in System z Hardware Design and GDPS® Technical Marketing Support. He holds a B. Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay, and an MS in E.E. from the University of Pittsburgh.

Jodi Everdon is a Staff Software Engineer at IBM Poughkeepsie. She is a managing editor for *z/OS Hot Topics*, the Information Planner for z/OS Distributed File Service zFS and SMB, and the author of *z/OS Problem Management*.

Marianna Frank is a Software Engineer at the IBM Lab in Boeblingen. Since joining IBM in 2006, she has worked on IBM Dynamic Infrastructure and WLM.

Anthony Giorgio is a Staff Software Engineer at IBM Poughkeepsie and a member of the On-Demand Infrastructure Team. Previously, he worked on dbx Debugger and ported tools. Anthony joined IBM in 2001 with a BS and MS in Computer Science from Polytechnic University.

Saheem Granados, CISSP, is an Advisory Software Engineer. In his nine years with IBM, he has worked on security software, including Security Server LDAP Server and Trusted Key Entry (TKE). Currently, he designs Encryption Facility V1R2 OpenPGP support.

Susan Greenlee is the project manager for the System z Linux and Virtualization Development Team. Her 17 years with IBM includes more than ten years of experience in z/OS development.

Rich Guski is a Senior Technical Staff Member at IBM Poughkeepsie. His current focus is on security design and architecture for System z.

Charles Haight is an Advisory Software Engineer in z/OS System Verification Test at IBM Poughkeepsie. His primary focus is on zFS, with previous work in z/OS UNIX and z/OS Language Environment.

Evan Haruta is an Advisory Software Engineer in the Systems and Technology Group at IBM Poughkeepsie. He works in z/OS Level 2 for DIV, auxiliary and real storage management.

Jürgen Holtz works in Design and Development for Tivoli System Automation for z/OS at IBM Boeblingen. Since joining IBM in 1990, he has worked on RMF and WLM.

Steven B. Jones has been with IBM for 20 years, working in a variety of BCP components, including global resource serialization (GRS) and XCF. He now leads the development teams for SMF, Allocation, and Scheduler.

Michael J. Jordan is an IBM Senior Technical Staff Member (STSM), and the strategist for z/OS Cryptographic Services. His previous projects include Parallel Sysplex and RACF.

Frederick Lates, an Advisory Software Engineer, works in z/OS Platform Evaluation Test (zPET). Since joining IBM in 1978, his assignments have included systems support for IBM Global Services.

Alfred Lease Jr., an Advisory Software Engineer, works in z/OS Platform Evaluation Test (zPET) and z/OS Integration Test. His more than 30 years at IBM includes experience in Operations, Development, Service, Test, and Customer Support.

Chwan-Hang Lee, a software developer at IBM Toronto, has worked on z/OS C/C++ and PL/I compilers for more than ten years. Currently, he leads the development of optimization and code generation for z/OS.

Barry Lichtenstein is a Senior Software Engineer with z/OS Program Management Binder Development. His 25 years with IBM includes work in z/OS Language Environment and z/OS UNIX. Barry holds an MS in Comp. Sci. from Rensselaer Polytechnic Institute.

Adrian Lobo, a Software Engineer, has spent his entire second life in DB2 Development, working on authorization.

Kristine Logan has worked on MVS and z/OS documentation for 20 fun-filled years. Working on IBM Health Checker for z/OS has been one of the most interesting projects of her career, with its emphasis on customer feedback and teamwork.

Randy Love, an Advisory Software Engineer, has been a RACF function tester since 1992 and a Test team leader since 1995. He has been involved with RACF support of DB2 since its inception (DB2 V5).

Raymond Mak is a software developer in C/C++ Compiler Development for z/OS and workstation platforms at IBM Toronto. Most recently, he worked on the METAL option. Raymond joined IBM in 1994.

Renata Rand McFadden is a Senior Software Engineer and team leader for DFS/SMB Development and L3. Her 11 years with IBM include work in z/OS UNIX, FCT, and SMB. Renata holds an MBA and MS degrees in Computer Science and Psychology.

Jim McGurl has been with IBM for 25 years and a member of System Test for over 20 years. Currently, he works on the zTestware Team, focusing on z/OS testing, tool creation, and TPNS workload development.

Ed Mekeel works in System z Software Performance at IBM Poughkeepsie. His 37 years with IBM includes five years in Operations and seven years in Automated Operations Software Development.

Laurel Morschhauser, an IBM Senior Software Engineer, is currently leading the latest z/OS ESP. Her 27 years of experience includes assignments in z/OS Development, Function Test, System Test, and Release Management.

Colin Moschen has been with IBM for 25 years and a member of System Test for ten years.

Wolfgang Mueller is the IRMM lead architect at the Storage Software Development Lab in Mainz, Germany. He has worked on the architecture and design of IRMM since 2001.

Mark Nelson, CISSP, is a Senior Software Engineer. He has spent the last 20 years working in RACF Design and Development, focusing on DB2, reporting, and auditing.

Barbara Neumann, an Advisory Software Engineer, is the team leader for z/OS Language Environment information.

Jun Ogata has been a member of the RACF Function Test Team for more than 15 years. He has also tested other components, most recently XML System Services for the last two releases.

Michael Onghena, an Advisory Software Engineer, is a RACF designer and developer at IBM Poughkeepsie. He has been with IBM for 15 years.

Tom Petrolino is an Advisory Software Engineer at IBM Poughkeepsie. He is a frequent speaker at SHARE for z/OS Language Environment.

Kathy Pfeiffer is an Advisory Software Engineer at IBM Poughkeepsie. She is the curriculum manager for the IBM Academic Initiative System z9 Program.

Rich Prewitt is a Senior Technical Staff Member (STSM) in z/OS Test Design and Architecture and leader of the z/OS SVT team. Rich has published a number of articles on testing and is a co-author of *Software Testing Techniques: Finding the Defects that Matter*, published in 2004.

Don Resnik is the program manager for the IBM Academic Initiative for zSeries program.

Bob Rogers is a Senior Technical Staff Member (STSM) in z/OS Design and a member of the zSeries Software Design Council. He is a frequent speaker at SHARE and holds a number of software patents, including one for the storage management algorithm of Hiperbatch™.

Hiren Shah, a Senior Software Engineer, works on EWLM. He has design and development experience in performance management software and was involved with the development of IRD on z/Series. Hiren holds an MS in Computer Science from New Jersey Institute of Technology.

Sue Shumway, a Staff Software Engineer, is an information developer for EWLM. Previously, she worked on WebSphere Application Server for z/OS and *z/OS Hot Topics Newsletter*. Sue holds a BS in engineering and an MS in Technical Communication from RPI.

Dale Tanner graduated from Wichita State University in 1982 and has been an MVS system programmer ever since. He joined IBM in 1999 and currently works in Integrated Technology Delivery, Server Systems Operations.

Greg Thompson works in IBM System Test where he tests new functions for JES and SDSF. Before joining IBM, Greg was a systems programmer at Qwest Communications International, Inc. for 30 years.

Ann Totten is a Senior Software Engineer in z/OS System Verification Test at IBM Poughkeepsie. Her previous assignments include z/OS UNIX and Open Group conformance testing in Functional Verification Test.

Robert Vaupel has worked at IBM Boeblingen for 20 years, after obtaining an MS degree in Computer Science from the University in Karlsruhe, Germany. His career at IBM has focused on performance and WLM.

Guenter Vater has worked at IBM Boeblingen for ten years. He holds a degree in Computer Science from the college in Esslingen, Germany. During his career, Guenter has worked for various projects within system automation and WLM.

Romney White joined IBM in 1997 after a long career as a customer, entrepreneur, and ISV. Currently a Senior Technical Staff Member in z/VM Strategy and Development, Romney is a frequent speaker at SHARE and other user group meetings and conferences.

Stefan Wirag is an Advisory Software Engineer at IBM Boeblingen. He works in WLM Development, with a focus on the integration of WLM and EWLM.

Mark Wisniewski has worked in System Performance for 25 years. He is responsible for assessing current and future systems growth constraints and improving performance across System z.

Kendrick Wong has been with the z/OS C/C++ compiler team for ten years. Currently, he is the lead developer for Common Debug Architecture, introducing the DWARF debug format for the z/OS C/C++ compilers.

Scott Woolley has spent much of his IBM career in RACF Development focusing on encryption, auditing, and identity mapping. Scott strives to find the humor in Security.

Mark T. Wright is a Senior Software Engineer in z/OS Communications Server Development at IBM Raleigh, where he is the lead developer for the initiative, Make It Easy. Mark has been with IBM for 25 years.

International Business Machines Corporation
Department 55JA, Mail Station P181
2455 South Road Poughkeepsie, NY 12601-5400
United States of America

Produced in the United States of America
08-07

The z/OS Hot Topics Newsletter is published twice yearly as a service to customers, providing articles of interest to the z/OS community.

Corrections and clarifications

Last issue had its share of errors and omissions, for which we editors duly hang our heads and groan:

- In the article about DB2 and DFSORT ("The latest thrilling celebrity hook-up"), we erroneously referred to "z/OS Version 8" when we meant DB2 Version 8. Apologies to the author and to our readers for the typo, which we corrected in a subsequent printing of the Newsletter.
- In the Contributors section, we inadvertently omitted the entry for author, Laurel Morschhauser, who co-wrote (with Kathy Kulchock) the issue's piece on z/OS early support programs, "Is there ESP in your future?" We apologize to Laurel for the oversight. We have included her bio in this issue's Contributors section (page 78).

Do you see z in your future? Take the System Programmer Mastery Test

BY R. DAVID BOENIG

In 2002, IBM conducted a survey of more than 850 mainframe system programmers. In response to the question, "How many years of experience do you have working with OS390 or z/OS?" more than half the respondents said they had over 20 years of experience on the mainframe. Only about 10% of the respondents had ten years or less of experience.

So what does this mean to you and me?

OPPORTUNITY!

The survey suggests that a large number of mainframe experts are nearing retirement. As this group of programmers and administrators begin to do so, they will leave many very large shoes to fill. These shoes will translate into opportunities for advancement in the mainframe-computing environment.

If you think you have a future working with z/OS and you would like to stand out from the crowd, read on.

Test your mettle for Big Iron

IBM recently introduced the System z Entry Level for z/OS System Programmer Mastery Test. This exam is designed to help a person determine whether he or she has mastered the basic skills necessary to perform the tasks of an entry-level system programmer for System z.

The exam covers topics such as:

- Your knowledge of hardware design and configuration
- Your ability to use ISPF and TSO/E.

The exam tests you on some core concepts that include the use of data sets, JCL, and how to view job output in SDSF.

You must also show that you have a basic understanding of application programming on z/OS.

In addition, you must correctly answer questions about concepts of online processing in System z, including the use of the following products:

- CICS
- WebSphere Application Server
- DB2.

For more information about the concepts covered in the exam, visit the following Web site:
ibm.com/certify/mastery_tests/objz01_mt.shtml

Start with the textbook

The System Programmer Mastery Test is based on *Introduction to the New Mainframe: z/OS Basics*, SG24-6366, an IBM Redbooks publication. This book is available as a PDF download (or in hardcopy for a fee) at the IBM Redbooks Web site: **www.redbooks.ibm.com/abstracts/sg246366.html**.

IBM recommends that you review the textbook before taking the exam. The same textbook is used in the ten-week course offered through Marist College of Poughkeepsie, New York, which I personally attended before taking the exam. By the end of the course, I had read the entire textbook and was well prepared for the exam.

Register for the test

To take the exam, you can register online at one of the following locations:

- Thomson Prometric: **http://securereg3.prometric.com/**
- Pearson VUE: **http://www.vue.com/**

The exam contains 64 questions in multiple-choice format, and lasts 90 minutes. A passing score is 66-75%, depending on which version you take.

The cost for taking the actual exam is $95, but you can try a practice version (which I recommend doing) for only $10. The practice exam is available online at: **http://ibmtesting.prime.prometric.com/**.

The future awaits

The z/OS System Programmer Mastery Test can help you to ensure that you have a basic knowledge and understanding of the System z environment. In passing this exam, you demonstrate to yourself and management, coworkers, and prospective employers, that you have achieved this mastery over the entry-level system programmer functions for z/OS.

Further, you will truly have an understanding of the System z environment that you can take with you into your future job opportunities.

So, what are you waiting for? Test your mettle for Big Iron!

GA22-7501-13