IBM

z/OS Version 1 Release 10 Implementation

- EAV volumes, 64-bit common, WLM, JES3, JES2
- CIM, XML, RRS, SMP/E, LE, z/OS UNIX
- HiperDispatch, IOS, ISPF, XCF/XES

Paul Rogers
Patrick Bruinsma
Paul-Robert Hering
George Kozakos
Lutz Kuehner
Jean-Louis Lafitte
Gil Peleg

Redbooks

ibm.com/redbooks

International Technical Support Organization

z/OS Version 1 Release 10 Implementation

April 2009

SG24-7605-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

First Edition (April 2009)

This edition applies to Version 1 Release 10 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AD/Cycle®
AIX®
C/370™
CICS®
DataPower®
DB2®
DRDA®
DS6000™
DS8000®
ECKD™
Enterprise Storage Server®
eServer™
FICON®
FlashCopy®
GDPS®
Geographically Dispersed Parallel Sysplex™
HiperSockets™
Hiperspace™
HyperSwap™
IBM®
iSeries®
Language Environment®
MQSeries®
OS/390®
Parallel Sysplex®
POWER5™
PR/SM™
pureXML®
RACF®
Redbooks®
Redbooks (logo) ®
S/390®
SecureWay™
System Storage™
System z10™
System z9®
System z®
System/390®
SystemPac®
Tivoli®
TotalStorage®
VM/ESA®
VTAM®
WebSphere®
z/Architecture®
z/OS®
z/VM®
z9®
zSeries®

The following terms are trademarks of other companies:

InfiniBand Trade Association, InfiniBand, and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Java, JNI, RSM, Solaris, Sun, ZFS, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication positions the new z/OS® Version 1 Release 10 for migration by discussing many of the new functions that are available. The goal for the z/OS platform is to eliminate, automate, and simplify tasks without sacrificing z/OS strengths, and to deliver a z/OS management facility that is easy to learn and use.

This release of the operating system supports new capabilities that are designed to provide the following enhanced and new functions:

- An architectural limit of hundreds of terabytes (TBs) for DASD volumes, up from the current limit of approximately 54 GB per volume. Known as Extended Address Volume (EAV), this function is planned to initially support 223 GB per volume on z/OS V1R10 and IBM System Storage™ DS8000®, when available.
- Up to 64 engines, up to 1.5 TB per server with up to 1.0 TB of real memory per LPAR, and support for large (1 MB) pages on the z10 EC, providing performance for critical workloads.
- HiperDispatch can help provide increased scalability and performance of higher n-way z10 EC systems by improving the way workload is dispatched within the server.
- A new Basic HyperSwap™ capability (to be enabled by TotalStorage® Productivity Center for Replication Basic Edition for System z®).
- Improved management for processor capacity via a new Capacity Provisioning Manager.
- Improved productivity with simplifying diagnosis and problem determination.
- Expanded Health Check services.
- Automatic dump and re-IPL capability.
- zIIP-assisted z/OS Global Mirror (XRC) and additional z/OS XML System Services exploitation of zIIP and zAAP help make these workloads more attractive on System z.
- A new SMP/E version helps simplify the task of verifying required software fixes identified in Preventive Service Planning (PSP) buckets. PSP buckets identify required software fixes for new hardware devices, toleration and coexistence of new software releases, and for enabling new functions.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, z/OS UNIX®, JES3, and Infoprint Server. Before joining the ITSO 20 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England for eight years providing OS/390® and JES support for IBM EMEA and also in the Washington Systems Center for three years. He has worked for IBM for 41 years.

Patrick Bruinsma is an Accredited Senior IT Specialist with 10 years of advanced experience on z/OS, OS/390, DB2®, MQSeries®, WebSphere® MQ Workflow, Blaze Advisor, CICS®, Workload Manager, Java™, WebSphere Application Server for z/OS, z/OS UNIX (UNIX System Services), and SAP® on z/OS, as well as general installation and

implementation expertise on nearly all z/OS software. He is proficient in teaching technical education and has coauthored several Redbooks. In January 2007, Patrick was appointed System z Software IT Architect. For the last four years he has been focusing on the design and integration of large-scale commercial computing environments, mainly in the mainframe computing arena. He supports the System z sales team and provides expertise on platform positioning and solution design.

Paul-Robert Hering is an IT Specialist at the ITS Technical Support Center, Mainz, Germany. He advises customers on z/OS and UNIX System Services-related questions and problems. He has participated in several ITSO residencies since 1988, writing about UNIX-related topics. Before providing support on OS/390 and z/OS, Paul-Robert worked with VM and all its different flavors (VM/370, VM/HPO, VM/XA, and VM/ESA®) for many years.

George Kozakos is a Senior IT Specialist with IBM Australia. He has more than 20 years of experience in MVS system programming. George's areas of expertise include Server Time Protocol and GDPS®. He holds degrees in Computing Science and Pure Mathematics.

Lutz Kuehner is a Senior IT Specialist at IBM. He currently works as a System z IT Architect in Germany, providing support to the System Technologie Group sales organization. He has 22 years of experience in the mainframe field. His area of expertise include z/OS and z/OS UNIX, CICS and high availability topics in general. Lutz has written extensively on z/OS and z/OS UNIX.

Jean-Louis Lafitte is Senior IT Architect at GATE Informatic SA, an IBM Premier Business Partner in Switzerland. He has 37 years of experience on IBM Large Enterprise Systems, has worked on different parallel machines (IBM RP3, CM2, KSR1, IBM SP1, SP2 and BG/L, P), and has been associated with Parallel Sysplex® since 1984. Jean-Louis holds a Ph.D. degree in Theoretical Computer Science and several patents in System/390® architecture. He is a member of ACM and IEEE.

Gil Peleg is a Systems Programmer working for VeNotion Technologies Ltd. in Israel. He has 10 years of experience in mainframe systems programming and in teaching mainframe-related courses. He holds a degree in Computer Science. Gil also coauthored the z/OS V1R8 and z/OS V1R9 Implementation Redbooks.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

z/OS Version 1 Release 10 overview

This chapter describes the functional enhancements to z/OS Version 1 Release 10 (z/OS V1R10).

In this release of the z/OS operating system, IBM builds on its leadership capabilities by not only enhancing time-tested technologies, but leveraging deep synergies with the new IBM System z10™ Enterprise Class (z10 EC) server and System Storage family of products.

z/OS V1R10 and the new z10 server together deliver tremendous economies of scale and adaptability of resources. With support for up to 642 engines in a single logical partition (also available on z/OS V1R9) and up to 1.0 terabyte (TB) of real memory per LPAR (also available on z/OS V1R8 and V19), an application and data serving requirements may not have to be partitioned on unnatural boundaries.

In z/OS V1R10 (and available with z/OS V1R9), the memory architecture is extended to allow large (1 MB) pages to be used in addition to the existing 4 KB page size. This is expected to reduce memory management overhead for exploiting applications.

Just as important as system scalability, is how it performs with that scalability. z/OS V1R10 and with the z10 server is HiperDispatch (also available on z/OS V1R7, and later), a capability that can provide intelligent dispatching of z/OS workloads, to help improve the performance for higher n-way systems. A new Capacity Provisioning Manager planned for z/OS V1R10, and available on z/OS V1R9 with APAR OA20824, can monitor z/OS systems on z10 servers. Activation and deactivation of temporary capacity can be suggested or performed automatically based on user-defined schedules and workload criteria1. RMF or equivalent function will be required to use the Capacity Provisioning Manager.

1.1 z/OS V1R10 enhancements

z/OS V1R0 is the next step in the evolution of the System z mainframe. It redefines scalability, performance, availability, and economics for the platform. It provides control over system resources, flexibility and autonomics for unforeseen demands, world-class security and availability, and deep synergies within the platform. z/OS provides a solid foundation for extending existing applications, adopting new technologies, and above all leveraging an IT investment.

The goal for z/OS platform simplification is to eliminate, automate, and simplify tasks without sacrificing z/OS strengths, and to deliver a z/OS management facility that is easy to learn and use with integrated task guidance, and to embrace IBM's converged systems management strategy.

With every release, z/OS continues to refine its error checking, fault tolerance, isolation, error recovery, and diagnostic capabilities. z/OS V1R10 availability enhancements include designs for improved console processing, reduced need for JES2 restarts with JES2 Dynamic exit capability, support for automatic standalone dump and auto-IPL, and new health check services and checks.

z/OS V1R10 runs on the following IBM System z servers:

z10 EC, z9® BC, z9 EC, z900, z990, z800, and z890

1.2 z/OS V1R10 support for z10 EC

z/OS and its subsystems provide for scalability not only based on chip speeds, but on a single image, clustering, storage and data handling basis as well. This holistic and balanced approach to scalability means your System z environment is capable of handling the growth of your user base, applications, business processes, and data processing needs. Scalability improvements planned for z/OS V1R10 include:

- Up to 64 processors per logical partition, and up to 60 LPARs per server with System z10. With up to 64 processors per logical partition and as many as 32 z/OS logical partitions able to be configured in a Parallel Sysplex cluster, up to 2,048 engines' worth of processing capacity are available to application workloads.

  Note: The total number of processors defined in a z/OS logical partition is the sum of general-purpose processors (CPs), System z Application Assist Processors (zAAPs), and System z9® Integrated Information Processors (zIIPs).

- z/OS V1R8, z/OS V1R9, and z/OS V1R10 are designed to support 4 TB of real storage on a single z/OS image. The maximum amount of storage supported by these operating systems on System z processors is shown in Table 1-1.

Table 1-1 Storage supported on System z processors

Processor	Maximum amount of real storage supported
System z10	1 TB
System z9	512 GB
z990	256 GB

- Memory architecture is extended to support large (1 MB) pages. When large pages are used in addition to the existing 4 KB page size, they are expected to reduce memory management overhead for exploiting applications.

z10 EC server microcode and HSA

System z servers are designed to reduce planned and unplanned outages through the use of self-healing capabilities, redundant componentry, dynamic sparing, and the ability for concurrent upgrades and microcode changes. The z10 EC server provides additional microcode driver enhancements, and dynamic segment sparing for memory, as well as a fixed Hardware System Area (HSA).

HiperDispatch

A new function with z/OS V1R10 is HiperDispatch, which can help provide increased scalability and performance of higher n-way System z10 servers by improving the way workloads are dispatched within the server. HiperDispatch is designed to accomplish this by recognizing the physical processor topology, tracking where units of work have run, and attempting to redispatch them as close to the same physical processors as possible. This intelligent dispatching can help reduce the effects of memory latency to improve performance and reduce CPU time. HiperDispatch is available with the following environments:

- The z10 EC
- z/OS (z/OS V1R7 with the IBM zIIP support for z/OS
- z/OS.e V1R6/R7 Web deliverable and PTFs
- z/OS V1R8 or V1R9 with PTFs
- z/OS V1R10

Note: For more information, see the Preventive Service Planning bucket (PSP) for z10 EC, 2097DEVICE.

1.3 IBM System Storage DS8000

Exploiting the capabilities of a new 3390 device model on IBM System Storage DS8000 storage subsystems, the support is provided in the following areas:

- The architectural limit of hundreds of TB for DASD volumes is up from the current limit of approximately 54 GB per volume. This new support is called extended address volume (EAV). This function initially supports 223 GB per volume on z/OS V1R10 and IBM System Storage DS8000.

  Note: This new function is expected to provide immediate substantial constraint relief for installations with a large number of large VSAM data sets. This is also expected to help improve storage management administration over time, as a relatively small number of large volumes are thought to be simpler to manage than a larger number of smaller ones. IBM recommends the IBM HyperPAV licensed function on the IBM System Storage DS8000 series be leveraged to help manage the number of paths to devices defined as EAV.

- This support provides compatible access to data residing on cylinders below 65,520. Also, the existing 3390 device geometry (the track length and number of tracks per cylinder) is maintained on EAV.

- With z/OS V1R10, support is planned for SMS and non-SMS managed VSAM data sets (ESDS, KSDS, RRDS, and LDS) at any location on an extended address volume.

  Non-VSAM data sets, catalogs, page data sets, and VSAM data sets with the KEYRANGE or IMBED attribute are restricted to the first 65,520 cylinders. With this initial support, space after the first 65,520 cylinders is intended to provide constraint relief for applications using large VSAM data sets, such as those used by DB2, CICS, zFS file systems, SMP/E CSI data sets, and NFS mounted data sets.
- A new dynamic volume expansion function is designed to eliminate the need to copy volumes to increase their size.

In the future, IBM intends to expand support for EAV with larger volume sizes and support for additional data set types and access methods. For more information, refer to the Statement of Direction section.

1.4 zIIP and zAAP support

z/OS V1R10 has new supports to enable additional XML processing to be made eligible for the zIIP and zAAP specialty processors. IBM middleware (such as DB2 V9) and other products can benefit from this new functionality in addition to taking advantage of the z/OS XML System Services capabilities available today. These enhancements are expected to help improve the price performance of XML processing on z/OS and ultimately may help facilitate the decision to develop more XML-based applications on z/OS.

Enhancements in z/OS XML System Services and the XML Toolkit for z/OS (5655-J51) increase the amount of XML workload eligible for the zAAP and zIIP specialty engines.

z/OS XML System Services

z/OS XML System Services now includes additional zIIP exploitation, specifically enabling all z/OS XML parsing in enclave SRB mode to be eligible for a zIIP. For example, with respect to DB2, z/OS XML processing may be partially directed to zIIPs when utilized as part of a distributed request (like DB2 DRDA® today). This enhancement can help further benefit DB2 pure XML workloads by optionally directing all z/OS XML System Services parsing that is executed in enclave SRBs to the zIIP. This function is to be available on z/OS V1R8 and z/OS V1R9 with the PTF for APAR OA23828.

z/OS XML System Services validating parser

Validation support is designed to allow a program to determine whether an XML document meets the requirements expressed in an XML Schema definition (XSD). z/OS XML System Services adds validating parsing for both zIIP and zAAP processing.

XML Toolkit for z/OS (5655-J51)

The XML Toolkit for z/OS (5655-J51) is enhanced so that eligible workloads can use z/OS XML System Services. This allows eligible XML Toolkit processing for non-validating parse requests to exploit the zAAP. This function is planned to be available on the XML Toolkit for z/OS V1R9 with an SPE. XML toolkit support for processing validating parse requests using z/OS XML System Services, with the appropriate off-load of eligible work to zAAP, is planned at a future date.

Note: The delivery of these functions satisfies the statements of direction in Hardware Announcement 107-190, dated April 18, 2007, and Software Announcement 207-175, dated August 7, 2007.

z/OS Global Mirror

With z/OS V1R10, z/OS Global Mirror (formerly known as Extended Remote Copy, XRC) is announced and is enabled to exploit the zIIP specialty engine. The zIIP-assisted z/OS Global Mirror function is designed to make most of the z/OS DFSMS System Data Mover (SDM) processing eligible to exploit the zIIP specialty engine. This capability is available with any disk storage subsystem that supports z/OS Global Mirror and will be available via PTF for APAR OA23174 for z/OS V1R8 and V1R9.

With zIIP-assisted z/OS Global Mirror, the zIIP essentially becomes a z/OS data mirroring engine that can provide better price performance and improved utilization of resources at the recovery site. Most DFSMS system data mover (SDM) processing is eligible to be redirected to a zIIP processor, which can help lower server utilization at the recovery site, or create server “white space” to be used for other projects.

z/OS Global Mirror provides an asynchronous, multi-site remote mirror solution across intercontinental distances for z/OS, zVM (with some configuration limitations), and Linux® on System z data, and is one of the technologies that GDPS is based on. z/OS Global Mirror protects data consistency across all volumes that have been defined for mirroring and the volumes can reside on several different storage units. z/OS Global Mirror is also flexible, accommodating variable data volumes and network bandwidths, thus minimizing the possibility of data desynchronization. Customers currently using z/OS Global Mirror or some other long-distance remote copy solution should consider zIIP-assisted z/OS Global Mirror.

1.5 Miscellaneous base control program enhancements

The following enhancements have been implemented in z/OS V1 R10 for the base control program (BCP) components.

z/OS V1R10 console support

The first stage of a comprehensive overhaul for system message processing was made available as a feature for z/OS V1R4, and integrated in z/OS V1R5 and later releases. The overall objective of the console enhancements is to improve system availability by enhancing the capacity and reliability of message delivery. To accomplish this, major changes to the message production and consumption flow help reduce the possibility of bottlenecks that can cause a backlog of undelivered messages. In z/OS V1R7, the next phase of console enhancements was made available, including support for deleting unused EMCS consoles, a new AMRF/ORE service routine, disassociating monitor messages from particular consoles, and support for enhanced recovery. In z/OS V1R8, the master console and console switch functions were removed, eliminating them as potential points of failure.

In z/OS V1R10, the final phase of console enhancements is introduced. In this final phase, console processing is designed to reduce serialization contention by reducing the scope of serialization for many operations, from a console class to an individual console. Additionally, support will be provided to increase the maximum number of MCS, SMCS, and subsystem consoles in a sysplex from 99 per sysplex to 99 active consoles per system; also, defining up to 250 consoles per system will be supported (of which up to 99 may be concurrently active), and wildcard support will be added for the DISPLAY CONSOLES command along with improved command response messages.

Reusing ASIDs

In z/OS V1R9, limited support was added for reusing the ASIDs of address spaces with cross-memory connections when they end, so the ASIDs remain available for the system to

assign to new address spaces. ASID reuse is intended to help you prevent planned and unplanned outages by avoiding exhaustion of usable address space slots on the system.

WIth z/OS V1R10, to enable the use of this new function, specify REUSASID(YES) in an active DIAGxx parmlib member. This will allow it to be used when certain options are specified on the START command or the ASCRE (address space create) macro for supported address spaces.

Currently, these z/OS address spaces support ASID reuse: CATALOG, LLA, and VLF. In z/OS V1R10, the following address spaces support ASID reuse:

- z/OS UNIX RESOLVER address space
- TCP/IP address spaces
- DFSMSrmm address space
- TN3270 address space

Before enabling ASID reuse on a production system, it is recommended to first enable it on a test system. For more information about enabling ASID reuse, see *z/OS MVS Extended Addressability Guide*, SA22-7614.

IOS IOSSPOF service

In z/OS V1R10, IOS provides a new IOSSPOF service designed to detect and report single points of failure in the I/O configuration for a single device or common points of failure between a pair of devices. The service is designed to be usable both by functions to perform such checks in real time, and by health checks.

IOS and subchannel set one

IOS is redesigned to allow Metro Mirror secondary devices to be defined in subchannel set one. This can in turn allow subchannels in subchannel set zero previously used for this purpose to be reused to define additional devices. This supplements the support for defining PAV aliases in subchannel set one that was in z/OS V1R7 on z9 EC servers. This helps to alleviate the constraint due to the 64K device limit.

Common storage above the bar

New support is provided for common storage above the 2 GB bar. A new virtual storage area, the high common storage area (HCSA), is defined. Storage management services and RMF support for HCSA are provided. This new support provides the infrastructure required for many users of CSA and ECSA storage to move data above the bar. This is expected to lead to virtual storage constraint relief (VSCR) over time.

System diagnostic work area (SDWA)

The system diagnostic work area (SDWA) is moved into 31-bit storage above the 16 MB line for AMODE(64) functional recovery routines (FRRs) in z/OS V1R10. SQA storage shortages can cause FRRs to be skipped when the SDWA is below the line. Moving them above the line is designed to help avoid one cause of abnormal address space termination.

1.6 JES3 enhancements

Spool Data Set Browse (SDSB)

JES3 now provides a SDSB (Spool Data Set Browse) interface allowing an application to allocate and read a spool data set without requiring the application to first go through SYSOUT Application Programming Interface (SAPI). The benefits are as follows:

- Multiple applications can read the same data set simultaneously.
- An application can look at a data set belonging to a job that is still running.
- An application can read buffered output data for an active job that has not yet been written to a spool data set, provided that the active job and the browsing application are running on the same system.

Unauthorized SSI function code calls

In z/OS V1R10, JES3 provides a new, unauthorized Subsystem Interface (SSI) function code calls for SSI function codes 11, 75, 79, and 80. These new functions are intended to allow programs to be written for user destination validation, sending messages to other users over the network, using the SYSOUT Application Programming Interface (SAPI), and obtaining detailed status information about jobs and SYSOUT in the JES queue.

1.7 JES2 enhancements

In this release, z/OS network availability improvements are planned. z/OS now provides an ability to restart JES2 NJE connections automatically, and introduce a new TCP/IP health check.

JES2 dynamic exits are provided with a new $T EXIT command. Additional commands intended to support refreshing JES2 load modules are also provided. The new $ADD LOADmod, $DEL LOADmod, and $T LOADmod,REFRESH commands are intended to refresh tables in a specified load module, existing exit points, and the list of routines associated with exit points without an IPL or JES2 restart.

New initialization parameters and updated commands are provided to allow installations to specify that NJE connections that terminate unexpectedly should be restarted after a specified interval. This new support is intended to help improve availability for these connections by automating their recovery.

1.8 z/OS UNIX System Services

In z/OS V1R10, z/OS UNIX System Services provides new shell commands.

- The submit command is designed to accept input from a data set, file, or stdin, and is expected to make it easier to initiate batch processing from within z/OS UNIX.
- A new UNIX command, **amblist**, is designed to provide the ability to invoke the AMBLIST program from the UNIX shell.

Support is provided to allow a program object stored in the file system to be loaded to a specified location in common storage (this function is often referred to as a “directed load”).

Currently file system lock recording is done at a module level. In z/OS V1R10, z/OS UNIX provides the ability to perform file system lock recording at a thread level.

Sysplex root

A z/OS UNIX System Services function is provided to allow changing the sysplex root data sets dynamically, without a sysplex-wide IPL. A new MODIFY OMVS,NEWROOT command is expected to eliminate a cause for planned outages and to facilitate migration of sysplex roots from HFS to zFS.

1.9 Workload Manager

With the ability to intelligently manage workloads, reprioritize work, and dynamically reallocate system resources between applications quickly and efficiently, z/OS and System z can handle unexpected workload spikes and improve your system's efficiency and availability, all while meeting application and business priorities.

z/OS Workload Manager (WLM) is unsurpassed in the industry in delivering the management of mixed diverse workloads according to business goals including response time goals. The scope of the Workload Manager and its exploiters extends from managing the incoming TCP/IP and SNA traffic to managing requests for I/O. z/OS middleware like DB2, CICS, IMS, WebSphere MQ, and other WebSphere products can take advantage of WLM to manage the priority and execution of transaction requests across the z/OS system.

For z/OS V1R10 WLM provides the following functional enhancements:

- Enhanced contention management
- Improved management of zIIP workloads
- The ability to manage selected components in service class SYSTEM
- New Performance Block delays

In addition, WLM now provides support for extracting the WLM service definition in XML format as well as installing and activating a WLM service definition in XML format via a CIM Server. Also, WLM supports providing information about the status of a z/OS system using a CIM Server.

Contention management phase 3

WLM contention management is redesigned for z/OS V1R10 to help address chronic or long lasting contention situations. Previously, WLM contention management could promote units of work that held resources requested by waiting units of work for short periods. WLM is now designed to promote units of work identified by exploiters for longer periods of time, and promote them to the priority of the highest-priority units of work waiting for a resource they are holding. This new support is expected to help prevent low-priority work from blocking higher-priority work while still managing the overall system in a way that is consistent with the goals you specify in the WLM policy. RMF support for this function provides information about the service times for workloads that were promoted, contention, and delay states in the RMF Postprocessor Workload activity report.

CPU management of zIIPs

Before z/OS V1R10, WLM algorithms for adjusting the dispatching priorities of work in each service class only considered CPs and zAAPs. To meet the needs of growing zIIP workloads, this processing is planned to be extended to include zIIP processors in z/OS V1R10. This is expected to provide better management for those workloads.

Protection of high consumers of fixed storage

To help prevent critical real storage shortages, new WLM function is provided for z/OS V1R10. New detection functions are designed to track sudden growth in fixed and pageable storage and react quickly to help avoid system outages. This new function is intended to identify fixed storage shortages, identify address spaces with the greatest growth in storage consumption, and issue new messages to identify them. This can help you use new automation methods to terminate address spaces consuming unacceptable amounts of fixed and pageable storage. The system is also designed to logically swap such address spaces and set nonswappable address spaces nondispatchable to avoid further storage consumption. Also, support provides for an ENF signal to allow other products to react to these situations.

Manage selected components in service class SYSTEM

To prevent the inadvertent misclassification of system address spaces, z/OS V1R10 WLM now will manage these address spaces in the SYSTEM service class even if they are differently defined in the WLM policy. The address spaces XCFAS, GRS, SMSPDSE, SMSPDSE1, CONSOLE, IEFSCHAS, IXGLOGR, SMF, and CATALOG are now managed in addition to the *MASTER* and WLM address spaces that were already automatically classified in the SYSTEM service class.

This is intended to prevent system problems from occurring during periods of high system utilization.

New PB delays

WLM provides performance blocks for use in application state (or phase) reporting. In z/OS V1R10, support for representing 10 additional delay states with performance blocks is provided, bringing the number that can be used for reporting to 15. Also, WLM is enhanced to allow applications to specify the delay state names, replacing the default names. These functions are intended to make it easier to determine which application phases are causing the greatest delays.

1.10 z/OS Communications Server

z/OS Communications Server plans to provide enhancements aimed at improving the user experience with installing, configuring, and operating.

With z/OS V1R10, z/OS Communications Server provides virtual storage constraint relief by changing the inbound data path to no longer use ECSA to hold inbound data for processing, or when queueing the data to the application. The TN3270 Server is also changed to reduce its ECSA usage for mapping of active sessions.

Improved network stack performance

z/OS Communications Server will focus on TCP/IP stack performance improvements in multiple areas, with designs intended to reduce CPU consumption and memory access latency, and improve throughput for all TCP/IP workloads.

Virtual storage constraint relief

To provide virtual storage constraint relief, z/OS Communications Server has changed the inbound data path to no longer use ECSA to hold inbound data for processing, or when queueing the data to the application. The TN3270 Server will also be changed to reduce its ECSA usage for mapping of active sessions.

Multiple VLAN support

z/OS Communications Server now allows you to configure multiple Virtual Local Area Networks (VLANs) from the same TCP/IP stack for a single OSA-Express QDIO port. Each TCP/IP stack supports a maximum of eight VLANs per OSA per IP version. This eases OSA port consolidation, for example multiple 1G ports can be consolidated onto a single 10G port, without having to redesign the VLAN definitions in the IP network.

HiperSockets multiple write facility

The IBM System z10 EC provides an enhancement for HiperSockets™ (Internal Queued Direct I/O) which is designed to improve the efficiency of the processing for internal LPAR-to-LPAR communications associated with large messages. The new HiperSockets multiple write facility allows messages that span multiple output buffers to be transferred with a single write operation from the source LPAR. This enhancement will improve HiperSockets throughput while also lowering the CPU cost related to the processing of transferring large messages from the source to the target LPAR. This function is also planned to be made available on z/OS V1R9 with a PTF.

TN3270E Telnet Server

This support provides shared LU name management among a group of servers running on the same system or within the same Telnet sysplex or subplex. Prior to this enhancement, LU names had to be manually partitioned among the TN3270E Telnet Servers to prevent concurrent assignment of the same LU name to clients connected to different servers. With this enhancement, one TN3270E Telnet Server in the group acts as an LU Name Server and allocates shared LU names to other TN3270E Telnet Servers within the group. This allows load balancing across multiple TN3270E Telnet Servers with consistent configurations. High availability of the LU Name Server service is provided with automated takeover and recovery.

Path MTU discovery for Enterprise Extender

This enhancement allows VTAM® to dynamically learn of any maximum transmission unit (MTU) size changes that occur in the underlying IP network associated with IPv4 and IPv6 Enterprise Extender (EE) connections. With this knowledge, VTAM can segment the data to the appropriate size and avoid IP packet fragmentation.

APPN extended border node

The EBN support is enhanced to let you restrict the searching capability of adjacent non-native nodes without having to code a directory services management exit. The AUTHNETS value on the ADJCP statement specifies the list of authorized NETIDs for the adjacent control point. Searches received from that control point will be rejected if the destination LU's NETID is not in the authorized NETID list.

High performance routing start option

z/OS Communications Server introduces a new high performance routing start option (HPRSESLM) to limit the number of sessions that are placed on each Rapid Transport Protocol (RTP) pipe. Once an RTP pipe reaches the specified session limit, another RTP pipe is chosen or created for new sessions. Limiting the number of sessions on a single RTP pipe can result in improved performance on multiprocessor systems by allowing concurrent traffic on multiple RTP pipes.

Load Balancing Advisor and Load Balancing Agent

In z/OS V1R10, the Load Balancing Advisor and Load Balancing Agent functions are enhanced to support the subplex functions introduced in z/OS V1R8. You can configure one Load Balancing Advisor per subplex, and each stack in the subplex will have a Load

Balancing Agent for that subplex. This allows load balancing for applications in one subplex to be independent from load balancing for applications in other subplexes, within a single sysplex.

Configuration Assistant for z/OS Communications Server

Configuration Assistant for z/OS Communications Server is a downloadable tool designed to simplify network and security management. For z/OS V1R10, the Configuration Assistant for z/OS Communications Server adds file import capabilities and support for IP address group definition to make the Configuration Assistant more responsive to networking needs.

The Configuration Assistant (CA) for z/OS Communications Server can help simplify the definition of IP security policies and provides a graphical user interface (GUI) for policy definition management. The CA exports those policy definitions to z/OS systems in the form of text configuration files which z/OS Policy Agent reads and installs into the stack. z/OS Communications Server and the CA are enhanced for z/OS V1R10 to allow the CA to import existing policy text files into the GUI. This allows the CA to learn of and absorb manual changes that the system administrator may have made to the policy configuration text files since the last time they were exported.

z/OS Communications Server policy

The z/OS Communications Server policy definitions include the specification of the IP addresses to which specific rules apply. IP Address Group definitions are used to define rules which apply to more than a single address. CA support for IP Address Groups is being expanded in z/OS V1R10 to support IP Address Group specifications for additional perspectives (for example, AT-TLS, IPSec). Also, the KeyExchangeRule is being enhanced for both the CA and the z/OS Policy Agent to support specification of an IP Address Group.

WebSphere DataPower integration

In z/OS V1R10, the Network Security Services (NSS) function is extended to allow a z/OS NSS Server to provide centralized security services to attached WebSphere DataPower® SOA appliance clients. These clients will be able to access SAF services at the z/OS system acting as the NSS Server to perform SAF based user ID authentication and access control checks without having to define the user IDs and access control rules in the appliance.

Defensive filtering

Defensive filtering support is provided as a mechanism for users to block detected attacks by dynamically installing defensive filters in a TCP/IP stack. Defensive filters are a new kind of deny filter that are always placed in front of IP security filters. The defensive filters can be installed autonomically by an external security information and event manager, or manually by an authorized user.

IPSec RFC Currency

This function implements a number of industry standards (RFCs 4301-4305, 4308) that are required by the US Department of Defense (DoD) for IPv6 certification, and by the National Institute of Standards and Technology (NIST) for general U.S. government use.

AT-TLS enablement for Load Balancing Advisor

The z/OS Load Balancing Advisor and Agent are enhanced to allow users to exploit the AT-TLS feature to secure connections that carry SASP flows. This will optionally allow you to control authentication, access control, and encryption for the load balancing protocols, using AT-TLS policies.

Application security controls

Additional application security controls are provided to let you restrict the use of listening ports and ephemeral ports to only those applications that have the appropriate authority, via SAF resource definitions. Also, new controls restrict the ability of applications to perform rpcbind registration and deregistration.

TCP/IP socket options

Two new socket API options are introduced in this release that allow applications to indicate how long receive and send type socket API calls should block waiting for their operations to complete, to prevent indefinite blocking inside TCP/IP for these types of socket API calls. The socket options, SO_RCVTIMEO and SO_SNDTIMEO, are defined in the POSIX standards, and are included in the Single UNIX Specification (SUS) V3.

New Java class

A new Java class is provided to allow Java programs to invoke the z/OS Communications Server FTP Client. This API support extends the existing z/OS Communications Server FTP Client API to support the Java programming language, and includes a sample Java program.

z/OS FTP support

The FTP Daemon, Server, and Client now provide APPLDATA for the TCP connections they use. This allows you to easily distinguish the TCP connections used by FTP from those used by other applications. APPLDATA is displayed in NETSTAT output, is available in SMF records, and is passed across the Network Management Interface (NMI).

The FTP Server support of Implicit Secure FTP is enhanced to allow the system administrator to decide which protocol to use to establish the implicit secure connection: the z/OS default protocol, or the de facto industry standard protocol. Using the de facto industry standard protocol for Implicit Secure FTP connection activation may result in compatibility with a greater variety of FTP clients.

FTP Server is enhanced to give the administrator greater control over which users can login to the FTP Server. A new keyword can be specified to optionally allow the FTP server to verify that the user has READ authority to the SERVAUTH profile EZB.FTP.systemname.ftpdaemonname.PORTxxxx. If the new keyword is specified, and the user is not authorized, then the user's login fails.

The FTP Server and Client handling of data set contention is improved, to issue messages that help you identify the contention, and to automatically retry the transfer to allow you to resolve the contention before failing the FTP attempt.

The FTP Server is enhanced to restrict the amount of time it spends retrying activation of the data connection to the FTP client. In addition, keep alives are supported on the data connection, to allow it to remain active even during times of inactivity that can occur when using FTP for long-running DB2 queries or jobs.

SMF recording

z/OS Communications Server is enhanced to generate SMF records for IPSec events such as secure tunnel activation, deactivation, and refresh for IKE tunnels, dynamic tunnels, and manual tunnels. These SMF events will be reported over the real-time SMF interface of the Network Management Interface (NMI), as well as over the traditional MVS SMF Exit interface. These unsolicited notifications of key security events can be used in conjunction with the polling NMI for security.

RFC compliant support

z/OS Communications Server is now compliant with RFC 4293, RFC 4292, RFC 4022, and RFC 4133 for the SNMP version-neutral MIBs. These are the MIBs that represent IPv4 and IPv6 objects. Additionally, Communications Server plans to enhance the SNMP generic LinkUp and LinkDown traps as described by RFC 2863 to include the ifName of the interface in the trap.

Traffic Regulation policy

In z/OS V1R10, the configuration of Traffic Regulation (TR) policy as part of the Quality of Service discipline will no longer be supported. All TR functionality will need to be configured under Intrusion Detection Services (IDS) policy, which was first made available in z/OS V1R8. This change only applies to the TR policy configuration. The TR functions themselves remain unaffected.

1.11 IBM Health Checker for z/OS

In z/OS V1R10, IBM Health Checker for z/OS provides not only more checks for RACF®, z/OS UNIX System Services, XCF/XES, and CINET, but it also provides support for log browse and saving data across IPLs, both of which can help improve analysis and problem determination.

New checks are implemented for Sysplex Failure Manager action specifications to improve sysplex availability.

A health check is added for CINET environments to confirm whether the port range defined for use by the OMVS address space has been reserved in the TCP/IP stack definitions.

Log browse service

A new log browse service is designed to enable an application to extract historical data from health check output written to a log stream. This is expected to be useful in helping an application establish historical views of values returned by various health checks. This service does the following:

- Allow checks to save data across IPLs in a data set so it can be accessible to instances of the checks running later, even after IPL. This support is designed to benefit those check writers who need persistent check-related information to be saved across system IPLs.
- In order to avoid a constraint on check-writing due to Health Checker's use of a single 2G data space, Health Checker now runs AMODE 64 and uses an area in extended storage (above 2G) that could then be extended if needed.

XCF and XES health checks

In z/OS V1R10, XCF and XES extend and enhance their existing health checks to provide new and improved checks to detect single points of failure for all types of couple data sets using a new IOSSPOF service to do the following:

- Check for appropriate separation of different types of couple data sets
- Check XCF signaling paths and structure sizes

1.12 Global resource serialization

A new global resource serialization (GRS) ENQ monitor REQTYPE=NCRESERVE filter is added that will allow the monitor to report only on unconverted hardware reserves (those that have not been converted to global ENQs). This is intended to make it easier to eliminate or reduce hardware reserves by helping identify candidates for reserve conversion.

A new function is provided for a migration path from a GRSRNL=EXCLUDE environment to full RNLs without requiring a sysplex-wide IPL for certain environments. GRSRNL=EXCLUDE is a special mode where GRS excludes most SYSTEMS (global) level ENQs to SYSTEM (local) level scope.

GRS display support provides for latch contention, to provide information intended to make it easier to see how long a latch has been held, how long contention has existed for it, and which units of work own or are waiting for a latch. This is expected to make it easier to diagnose latch contention problems on a running system. GRS has improved performance in both GRS Latch and ENQ processing. GRS plans to improve Latch performance through reduced code path and improved hardware cache alignment. ENQ performance and resource consumption improvements are planned through CMSEQDQ lock contention reduction as well as reduce code path lengths for GQSCAN, ISGQUERY, and ENF 51 (contention monitoring) processing.

1.13 DFSMS enhancements

DFSMS provides an enhanced DISPLAY SMS command with new options to display point-in-time PDSE cache information in real time (as opposed to SMF Records) and the overall effectiveness of PDSE caching. DFSMS also displays PDSE caching statistics at the data set level. PDSE members can be cached in a Hiperspace™ to provide enhanced performance for those PDSE data sets that are considered important in a critical performance path. The new command and its displays are designed to help you determine whether changes to cache settings might improve system performance.

In z/OS V1R10, DFSMS supports a new virtual concurrent copy (VCC) function. VCC is designed to use a FlashCopy® relationship rather than a combination of storage control cache and z/OS dataspaces, and to perform point-in-time backup processing for large amounts of frequently-updated data while using less cache and memory resources. This new function is supported during DUMP and COPY operations on DS8000, ESS 800, and other storage controllers that support FlashCopy at a data set level.

In z/OS V1R10, DFSMS supports a new DATACLAS that overrides certain SPACE parameters specified in JCL and in IDCAMS DEFINE commands. These new data class attributes are intended to allow you to specify that the units to be used to allocate primary and secondary space quantities (for example, tracks, cylinders, blocks, or bytes) be set from the attributes you set in the applicable data class. The allocation units and the secondary space quantity will also be made available to ACS routines to allow them to make more intelligent decisions. Additionally, another new data class attribute will allow you to specify that the system determine the block size, thereby overriding any user-provided block size. This new function is expected to help you better manage space in SMS-managed DASD storage, and to make it unnecessary to change a large number of batch jobs to achieve the same result.

DFSMSrmm and DFSMShsm are enhancements to the DFSMSrmm Report Generator to include the ability to generate reports of DFSMShsm processing. Refer to the DFSMSrmm enhancements listed in 1.13, “DFSMS enhancements” on page 14 for more details.

Extended address volume

Using the DS8000 storage subsystems, extended address volume (EAV) is designed to provide a new architectural limit of hundreds of TBs per volume, up from the current limit of approximately 54 GBs per volume (65,520 cylinders). z/OS V1R10 supports a maximum volume size of 223 GB (262,668 cylinders per volume).

With these volumes, they have fully compatible access to data residing on cylinders below 65,520. Also, the existing 3390 device geometry (the track length and number of tracks per cylinder) is maintained on EAV.

In z/OS V1R10, EAV volumes are supported for SMS and non-SMS managed VSAM data sets (ESDS, KSDS, RRDS, and LDS) at any location on an extended address volume. Non-VSAM data sets, catalogs, page data sets, and VSAM data sets with the KEYRANGE or IMBED attribute are restricted to the first 65,520 cylinders. With this initial support, space after the first 65,520 cylinders is intended to provide constraint relief for applications using large VSAM data sets, such as those used by DB2, CICS, zFS file systems, SMP/E CSI data sets, and NFS mounted data sets.

A new dynamic volume expansion function is provided to eliminate the need to copy volumes to increase their size.

This new function is expected to provide substantial, immediate constraint relief for installations with a large number of large VSAM data sets. This is also expected to help improve storage management administration over time, as a relatively small number of large volumes are thought to be simpler to manage than a larger number of smaller ones. IBM recommends the IBM HyperPAV licensed function on the IBM System Storage DS8000 series be leveraged to help manage the number of paths to devices defined as EAV.

DFSMSrmm support

DFSMSrmm provides improved parmlib support for tape library and tape volume partitioning, improved reporting for DFSMShsm activity, and support for new media end-of-life management policies based on media errors, volume usage, and age. These new functions are intended to make tape management easier and improve administrator productivity.

- With z/OS V1R10, exploitation of the SMI-S Storage Library profile by the DFSMSrmm CIM Agent is designed to enable client systems to more easily connect to the DFSMSrmm CIM Agent and also allows IBM TotalStorage Productivity Center to report on volumes managed by DFSMSrmm.
- Enhancements to DFSMSrmm CLI better enable IBM's Integrated Removable Media Manager for the Enterprise on System z (IRMM) to integrate for enterprise-wide tape management. In addition you no longer require WebSphere Application Server to host the DFSMSrmm Web service. These enhancements are also planned to be available for supported z/OS releases with APAR OA23266.
- Enhancements are provided to allow forward recovery of the DFSMSrmm CDS, when no journal backups are available, from the DFSMSrmm audit SMF records.
- Enhanced reporting capability with updates to the DFSMSrmm Report Generator to support keywords for assembler macros from which report types are derived and to add new built-in extract steps. This is in support of new SMF record types from DFSMSrmm and for DFSMShsm and DCOLLECT reporting.
- New run-time options to select which DFSMSrmm records are extracted for reporting, which should help reduce the resources required for tape reporting are provided.
- Support for optional policies to enable tape data sets deleted via normal disposition processing to be fast tracked back to the scratch pool are provided.

- New policies are provided to DFSMSrmm that set expected levels of retention and expiration for tape data sets to help avoid accidental loss of data.

DFSMShsm support

DFSMShsm provides support for a NEWNAME parameter for data set backup. This new function is intended to enable you to create a backup version of the specified data set and make it look like a backup of the data set specified with the NEWNAME keyword. This simplifies the process used to convert backups of an online data set to DFSMShsm backups while preserving the availability of the original data set.

DFSMShsm control data set (CDS) backup processing is enhanced to reduce the delay for starting CDS backup due to the concurrent processing of other DFSMShsm functions. Reducing the delay for starting CDS backup can improve the availability of other DFSMShsm functions, most notably recall. CDS backup improvements are enhanced in z/OS V1R10 for long-running DFSMShsm functions on systems in a sysplex with XCF capabilities by suspending them to allow a CDS backup to proceed, and restarted afterward. This is expected to reduce the time required to complete a CDS backup.

With DFSMShsm, a NEWNAME parameter for data set backup commands is enhanced to allow you to specify certain attributes to be used when creating a backup version of a data set. The new keywords NEWNAME, DATE, and TIME are added to the command to allow you to specify the data set name, date, and timestamp to be used for the backup data set when it is created. This can allow you to create multiple backup versions of the same data set using different names, dates, and timestamps.

DFSMSdss support

DFSMSdss provides enhancements to allow the use of DFSMSdss copy services and exploitation of Fast Replication services provided by DASD subsystems. This is designed to enable almost instantaneous copies of the control data set to be created, reducing recovery time objective.

Object access method

Object access method (OAM) is enhanced to provide support for objects larger than the current maximum of 256M (268,435,456 bytes). The new maximum object size is 2000M (2,097,152,000 bytes), and it is stored, in parts, sequentially to the DASD level of the OAM storage hierarchy only. This is expected to reduce the need to separate large binary strings into multiple objects and to simplify the application interface as the application does not have to materialize the entire object first before it can be accessed.

1.14 XCF and XES

In z/OS V1R10, there are improvements in the way XES and XCF handle CF locking requests. With this support, Coupling Facility locking operations will be queued when I/O resources are not immediately available. This is designed to reduce processor utilization for locking-intensive workloads in CF link- and subchannel-constrained environments.

WLM XCF signaling

Previously, WLM exploitation of XCF signaling was changed in APAR OA20484 for z/OS V1R6-V1R9 in the following ways:

- For asynchronous messages the sender no longer holds the sender latch.

- The monitoring task and some other communication functions are able to detect long waits of asynchronous senders and to terminate these waits.

WLM can now actively find out that another system no longer communicates and is able to initiate recovery for it, as new function integrated in z/OS V1R10.

1.15 z/OS security

Security is a word that is almost synonymous with mainframe. The combination of time-proven z/OS technologies and z/OS system integrity, which is IBM's long-term commitment to protecting key z/OS system resources, means z/OS is a natural choice if you want a platform that can help keep enterprise-wide data and transactions secure.

z/OS technologies, such as RACF to manage authorization and access to z/OS resources, Public Key Infrastructure (PKI) to provide low cost Certificate Authority life-cycle management on z/OS, and DB2 use of z/OS multilevel security (MLS) designed to meet the stringent security requirements of multi-agency access to data, are constantly being updated to meet installation requirements.

Security Server RACF

For z/OS V1R10, the following security updates are provided:

- Z/OS exploitation for RACF password phrases
- Additional RACF integration with IBM Tivoli® Directory Server for z/OS (LDAP)
- Improved RACF administration
- New cryptographic support

System SSL is upgraded with recent SHA-2 technologies.

In z/OS V1R8, RACF password phrase support was introduced, with infrastructure to support password phrases from 14 to 100 characters in length in addition to the long-supported 1-8 character passwords. In z/OS V1R9, the RACF support was extended to include 9-13 character password phrases when an ICHPWX11 user exit is in use (and to provide a REXX-based sample), eliminating the gap between password length and password phrase length. In z/OS V1R10, new support provides for password change logging and enveloping functions for password phrases, and provides password expiration warning for password phrases as is done for passwords, and can exploit password phrases when they have been enabled in RACF user profiles. These exploiting functions include the following:

- TSO/E logon
- z/OS UNIX kernel
- z/OS UNIX Shell and Utilities **`su`** and **`passwd`** commands
- C run-time functions login(), __passwd(), pthread_security_np() and getpass()
- Network Authentication Service support for Kerberos
- IBM Tivoli Directory Server (LDAP) for z/OS SDBM backend support for RACF password phrase envelope search capability and RACF password phrase change logging

  With this enhancement, you can start to implement enterprise-wide password synchronization (using, for example, IBM Tivoli Directory Integrator) where RACF users can now effectively have longer passwords with fewer character restrictions, such as can currently exist on Windows® and UNIX systems.
- The OpenSSH function of IBM Ported Tools for z/OS (5655-M23) when used on z/OS VR10

Custom fields are provided for RACF USER and GROUP profiles, with corresponding administration support using RACF commands, ISPF panels, and LDAP. This support is designed to allow you to add fields using a new RACF CFIELD class to define the new fields to be added to USER or GROUP profiles and the labels you want to use for them. These fields are added to a new CSDATA segment of USER and GROUP profiles. Once the fields have been defined, the RACF commands used for user and group administration and the corresponding LDAP administration support can be used.

RACF password administration is changed to allow more selective authority for resetting passwords to be granted. This support is designed to allow you to grant individuals the capability to reset passwords for one or more users or the users that are members of one or more groups without having the system-wide RACF SPECIAL attribute or access to the system-wide IRR.PASSWORD.RESET profile in the FACILITY class. New RACF support adds the required control to enable the target users of password resets to be scoped by the owner of the RACF user or users that are within a selected group tree. This support is provides better controls for allowing help desk personnel to do password resets without granting them additional authorizations.

RACF now allows messages ICH70001I and ICH70002I to be returned to a RACROUTE REQUEST=VERIFY/X application which specifies MSGRTRN=YES and ACEE=. This relaxes a restriction from earlier releases.

RACDCERT and PKI Services are planned to be able to generate and display the IPv6 type Internet Protocol address (IP address), in addition to the IPv4 format, in the certificate Subject Alternate Name extension.

RACDCERT is now able to generate and display the IPv6 type Internet Protocol address (IP address), in addition to the IPv4 format, in the certificate Subject Alternate Name extension.

The RACF RACDCERT command processor is planned to be modified to replace the BSAFE crypto provider that is presently imbedded in the command with the IBM Crypto Library in C (CLiC). The present BSAFE crypto provider is not capable of supporting RSA key lengths greater than 1024 bits in length. With CLiC, RACDCERT will be able to generate 4096-bit RSA keys through software, in addition to the hardware capability of generating keys with such length.

RACF database integrity

In z/OS V1R10, RACF is changed to help preserve RACF database data integrity. When a new system either IPLs, goes into data sharing mode with an RVARY DATASHARE command, or activates a database with RVARY ACTIVE, RACF command will check for indications of data sharing mode and non-data sharing mode systems using the same database concurrently, and for multiple sysplexes in data sharing mode, using the same database. If a mismatch is detected, a WTOR is issued, asking for direction (for IPL, either FAILSOFT, CONTINUE, or NODATASHARE; for RVARY either CANCEL or CONTINUE). This is designed to help improve availability by eliminating potential causes of database corruption.

PKI Services

Support for additional characters from the UTF8 character set for certificates supported by PKI Services is provided for z/OS V1R10, adding to the support made available in RACF in z/OS V1R9. Both are intended to improve interoperability with certificates created by other certificate authorities (CAs).

PKI Services provides support for three additional Distinguished Name attribute types, Domain Component, Distinguished Name Qualifier, and User ID.

z/OS Cryptographic Services

In z/OS V1R10, updates are provided to the z/OS Cryptographic Services Integrated Cryptographic Services Facility (ICSF) with the functionality introduced in the Cryptographic Support for z/OS V1R7-V1R9 and z/OS.e V1R7-V1R8 Web deliverable. The highlights of the ICSF enhancements to z/OS are 4096-bit RSA key support. Support is provided for 4096-bit RSA support on System z servers.

The servers must have the 4096-bit RSA signature generation and verification support available with feature 0863 installed, and the Crypto Express2 Coprocessor with microcode level MCL006-MCL009 on these servers:

z10 EC - z9 EC - z9 BC

Additional SHA hash algorithms are provided: SHA-224, SHA-384, and SHA-512. SHA-224 is supported on all hardware supported by z/OS V1R10. SHA-384 and SHA-512 are only available with System z10 servers.

Additional clear AES key algorithms are provided: AES-192 and AES-256. These algorithms are only available with System z10 servers.

ISO Format-3 PIN Block support that meets the ISO 9564-1 banking standard is provided. Feature 0863 must be installed, and the Crypto Express2 Coprocessor with microcode level MCL006-MCL009 on these servers:

z10 EC - z9 EC - z9 BC

Long random number callable service is provided. The service can create random numbers that are longer than 8192 bytes in length. This service is available on all hardware versions supported by z/OS V1R10. Optimum performance is available when feature 0863 is installed and the Crypto Express2 Coprocessor with microcode level MCL006-MCL009 on these servers:

z10 EC - z9 EC - z9 BC

Note: The Cryptographic Support for z/OS V1R7 through z/OS V1R9 and z/OS.e V1R7 through z/OS.e V1R8 Web deliverable is available. This Web deliverable supports z/OS V1R7 through z/OS V1R9 and z/OS.e V1R7 through z/OS.e V1R8. To obtain this Web deliverable, see

http://www.ibm.com/server/eserver/zseries/zos/downloads

The RACF RACDCERT command processor is enhanced to replace the BSAFE crypto provider that is presently imbedded in the command with the IBM Crypto Library in C (CLiC). The present BSAFE crypto provider is not capable of supporting RSA key lengths greater than 1024 bits in length. With CLiC, RACDCERT will be able to generate 4096-bit RSA keys through software, in addition to the hardware capability of generating keys with such length. The software support requires z890, z990, or a later server with feature 3863 installed. The hardware support will exploit the 4096-bit RSA key generation function available on z10 EC, z9 EC, and z9 BC servers with feature 0863 installed with the latest Crypto Express2 Coprocessor with microcode level 3.30, which is provided by MCL006 - MCL009.

System SSL

With z/OS V1R10, support is available to utilize hardware support for RSA digital signature generate and verification and RSA encrypt and decrypt available on z10 EC, z9 EC, and z9 BC servers with feature 0863 installed with the latest Crypto Express2 Coprocessor with microcode level 3.30, which is provided by MCL006 - MCL009.

The `list` command support in the command line version of gskkyman is to help you determine when certificates in key data base (kdb) files are due to expire and to obtain other information about the set of certificates in a key data base.

z/OS V1R10 completes the SHA-256 support, along with support for SHA-224, SHA-384, and SHA-512. This support is designed to extend the prior Secure Hashing Algorithm support to include import, display, export, and key data base storage for certificates using SHA-224, SHA-256, SHA-384, and SHA-512 algorithms.

IBM Tivoli Directory Server

In z/OS V1R10, new support is provided for an additional IBM Tivoli Directory Server for z/OS extended operation to support group access checking in addition to user access checking. This new function is planned to be made available on z/OS V1R8 and z/OS V1R9 with the PTF for APAR OA23078.

The IBM Tivoli Directory Server for z/OS provides enhancements for IBM Tivoli Directory Server compatibility and support of new z/OS Security Server RACF function. The following planned enhancements provide additional compatibility with the IBM Tivoli Directory Server for z/OS:

- A new plug-in support that allows configured plug-ins to be used to extend the capabilities of the IBM Tivoli Directory Server for z/OS. Pre-operation, post-operation, and client operation plug-ins are supported. HCD exploits this plug-in support for reading/updating IODF data.
- Improved handling of SHA- and MD5-based user password attributes for better compatibility with IBM Tivoli Directory Server for z/OS.

The following enhancements provide support of the z/OS Security Server RACF function:

- The IBM Tivoli Directory Server for z/OS SDBM backend support for the RACF custom user and group fields of the RACF CSDATA segment.
- The z/OS Security Server RACF password phrase, as follows:
  - Support for specifying a RACF password phrase for a simple bind or both native authentication and authentication to an IBM Tivoli Directory Server for z/OS SDBM backend. SDBM backend is also planned to support the RACF password phrase in the RACF Kerberos (KERB) segment.
  - Enhanced support in the IBM Tivoli Directory Server for z/OS SDBM backend is provided for RACF password phrase envelope search capability and RACF password phrase change logging.
- Support for making SASL external binds to an IBM Tivoli Directory Server for z/OS SDBM backend.

z/OS Communications Server

z/OS security capabilities do not stop at the server, but extend into the network as well. For z/OS V1R10 the z/OS Communications Server plans to provide improvements to its policy-based networking components, NSS, IPSec, and AT-TLS. Building on its history of Intrusion Detection Services (IDS), the z/OS Communications Server also introduces new defensive filtering capability. Defensive filters are evaluated ahead of configured IP filters, and can be created dynamically, for added protection and minimal disruption of services in the event of an attack.

1.16 Resource Measurement Facility

Resource Measurement Facility (RMF) Monitor III provides new reports about spin and suspend locks. The Spin Lock report will be designed to display the address spaces which hold locks and which are suspended. The Suspend Lock report displays how often global locks are held and who is spinning. Reporting of lock statistics is intended to help you analyze lock contention in the system.

The RMF Distributed Data Server is enhanced to support IPv6 connections from clients requesting RMF Monitor III performance data.

RMF is enhanced to save the actual device capacity in SMF 74 subtype 1 Device Activity records, including records involving extended address volumes (EAVs).

1.17 Hardware Configuration Manager

Hardware Configuration Manager (HCM) usability improvements include support for configuration packages similar to those supported by HCD, to allow a subset of a configuration to be created. For example, this support is designed to allow you to create and send a configuration package for a sysplex or a single site to another location.

There is support for importing and exporting I/O configuration data, similar to that provided by HCD. This function is designed to allow you to create I/O configuration statements for processor, switch, or operating system configurations of an IODF, and to migrate existing data sets containing I/O configuration statements into an IODF.

There is an enhancement to the Named Views capability of HCM to save lists of selected objects, the current zoom factor, and the scrolling positions. This is designed to allow the same part of a diagram to be displayed when restoring a view as it appeared when saved.

Using configuration packages with HCM

A configuration package is a subset of a centrally maintained master IODF that is extracted from the master IODF and distributed for activation at specified target systems. In z/OS V1R10, HCM implements the concept of configuration packages, as it is currently available in HCD. New dialogs are implemented to help HCM users to define, edit, transmit, and delete configuration packages, corresponding to the existing HCD functionality.

Create and migrate I/O configuration statements

In z/OS V1R10, HCM implements new dialogs that let users perform the following tasks, which previously have been available in HCD only:

- Export/build IOCP input data set
- Export/build I/O configuration statements
- Import/migrate IOCP input data or I/O configuration statements into an IODF

For exporting, the intent is to allow users to select where to store the resulting output data; for importing, to specify where the source data is located (either on the host or on the workstation).

1.18 Parallel Sysplex clustering and GDPS

Beyond single system availability is z/OS Parallel Sysplex clustering and GDPS disaster recovery. Parallel Sysplex is designed to provide your data sharing applications and data with not only continuous availability for both planned and unplanned outages, but also near-linear scalability and read/write access to shared data across all systems in the Parallel Sysplex for data sharing applications. z/OS V1R10 provides enhancements in support of Parallel Sysplex, as follows:

- Load Balancing Advisor support of subplexes
- XCF improvements
- RACF improved data integrity, and more

This release of z/OS (and all other supported releases) has updated support for GDPS V3.5 as well.

1.19 Resource Recovery Services

In z/OS V1R10, Resource Recovery Services (RRS) is designed to improve availability by allowing an application to request that RRS syncpoint processing be ended without completion. A new FORGET request will be supported by the ATRSRV function to allow another resource manager on any system in the sysplex to instruct RRS to discard an SDSRM's interest in a transaction. This is intended to allow the transaction to proceed to completion once other interests have been satisfied, which in turn can help you avoid SDSRM restarts.

RRS now writes to an optional archive log stream when a transaction is completed. To improve RRS performance, some installations disable the archive log. In z/OS V1R10, a new SETRRS ARCHIVELOGGING command is designed to enable you to specify whether the archive log is to be used dynamically, without an RRS restart. This is expected to help you avoid planned outages for subsystems and applications that use RRS services.

1.20 Hardware Configuration Definition

Hardware Configuration Definition (HCD) usability improvements include support for multiuser update access to IODFs. This multiuser access capability is designed to allow you to specify that work IODFs be shared and provide serialization to allow different parts of them to be updated by different people using concurrent HCD sessions.

1.21 Capacity Provisioning Control Center

In z/OS V1R10, IBM introduces the Capacity Provisioning Control Center, which is a tool for managing capacity provisioning for System z10 servers. It is designed to manage provisioning policies and domain configurations. Provisioning policies specify the criteria for capacity increases and decreases, while domain configurations specify systems to be observed and servers to be managed.

The Capacity Provisioning Control Center (CPCC) is planned to be available on z/OS V1R9 with PTF for APAR OA20824 (only on System z10 servers) and z/OS V1R10, when available.

Initial support is planned for a policy definition application which requires a workstation running Windows XP.

Specifically, the Capacity Provisioning Control Center will provide the following functions:

- Create and edit Capacity Provisioning policies
- Create and edit Capacity Provisioning domain configurations
- Connect to the Provisioning Manager; Display the status of the Provisioning Managers
- Install Capacity Provisioning policies and domain configurations into the Provisioning Manager

1.22 SMP/E

SMP/E is enhanced to help simplify the task of verifying required software fixes identified in Preventive Service Planning (PSP) buckets. PSP buckets identify required software fixes for new hardware devices, toleration and coexistence of new software releases, and for enabling new functions. More specifically, IBM will consolidate the lists of required fixes from PSP buckets and produce SMP/E-consumable meta-data in the form of HOLDDATA to identify those fixes. SMP/E will use the new HOLDDATA to identify what fixes are missing in a current software environment. In addition, SMP/E will help to simplify the task of selecting and installing the required fixes identified by the HOLDDATA.

Additional ease of use functions for SMP/E include the following:

- Providing UNIX utility input enhancements.
- Helping to simplify the review of SMP/E operations by consolidating and reducing the HOLDDATA report output, and by reducing the number of warning conditions that must be investigated.

1.23 System Logger

Improvements are implemented for the System Logger administrative data utility to help you set up System Logger resources and in problem determination involving log stream data sets including when a log stream was defined, the amount of space used in each offload data set (high-used RBA), and the oldest timestamp in each offload data set. For example, it will be possible to continue execution of the utility after specification errors have been encountered, so that subsequent problems may also be seen and corrected, and to allow duplexing-related parameters created by list output to be specified for DASD-only log streams.

A new enhancement provides increased availability of System Logger log streams by allowing updates to duplexing attributes to be put into effect without noticeable disruption to the log stream exploiters.

1.24 Language Environment

Language Environment® (LE) support in z/OS V1R10 now allows the following:

- Verification of the syntax of CEEPRMxx members
- Batch support to check one or more members using a new CEEPRMCC program

- Support for a CEEPRMCK CLIST
- XL C support for mixed addressing modes (32 and 64) for the METAL option
- Standard C++ Library enhancements including support for a subset of the ISO/IEC TR 19768, Technical Report on C++ Library Extensions
- New support for application IDs to login, password, and pthread security functions.
- The Language Environment IPCS formatter is planned to be enhanced to format additional C run-time control blocks.

Performance enhancements

- I/O performance for multi-threaded C/C++ programs that use a single thread for I/O operations to any particular file.
- New HEAPPOOLS support for applications using 31-bit memory allocation (__malloc31) services in an AMODE 64 environment is provided. This enhancement is being made available on z/OS V1R7, V1R8, and V1R9 with the PTF for APAR PK41618, APAR PK47298, and the PTF for APAR PK49427, when available.
- Additional decimal floating point support for the new hardware-based instruction processing on System z10 servers. This support is planned to be made available on z/OS V1R8 and later releases with the PTF for APAR PK54438 and its prerequisites.
- New prefetch built-in functions.
- Improvements to Standard C++ Library particularly with global placement new operators and the implementation of vector template class.
- Reduced debugger bring-up time.

CEEROPT module

In z/OS V1R10, the functions provided for Language Environment options by the CEEROPT module are extended to batch and AMODE 64 processing, in addition to the CICS and IMS environments.

WebSphere for z/OS (5655-N01)

WebSphere for z/OS (5655-N01) 64-bit users now have the capability to define very large heaps. Current processing dumps the heap ahead of the system and Language Environment thread stacks needed to debug. 64-bit storage is planned to be assigned a dump priority which allows the stacks to be dumped ahead of the heap.

1.25 Common Information Model

A new z/OS specific function is added for the Common Information Model (CIM) Server, which allows the CIM Server configuration to be changed using the MODIFY system command, so operators no longer need to go to the z/OS UNIX System Services command prompt for making configuration changes.

In z/OS V1R10, the following enhancements have been added:

- There is an upgrade to the z/OS CIM component to OpenPegasus 2.7, DMTF CIM Schema 2.13, and CIM Client for Java version 2 to keep z/OS at the latest level of the CIM/WBEM standard.
- A new z/OS-specific function for the CIM Server is added that allows the CIM Server configuration to be changed using the MODIFY system command, so that operators no

longer need to go to the z/OS UNIX System Services command prompt to make configuration changes.

- Enable the CIM Server to write audit log messages using SMF records.
- Enable the CIM Server for logon with pass-tickets instead of real world passwords. With this support for custom pass-tickets for z/OS UNIX applications it will be possible to have pass-tickets generated specific for the CIM Server rather than having to use the general OMVSAPPL APPLID. z/OS UNIX applications are now enabled to use their own, unique APPLID with security token validation C/C++ API functions like __passwd(), __login(), and pthread_securtiy_np(). An APPLID is used within RACF for validation of PassTickets, and for making finer, granular authorization decisions.

1.26 ISPF enhancements

The following ease of use functions are enhanced for ISPF with z/OS V1R10:

- The ability to specify multiple targets for the ISPF EDIT move and copy line commands. This can help to eliminate repetitious use of these commands when copying or moving lines that will have multiple destinations.
- A new z/OS UNIX interface to ISPF and TSO/E commands allows them to be issued from the Shell.
- A new ISPF service, DIRLIST, is available to display the z/OS UNIX directory. Also, a new ISPF UDLIST command allows a directory to be listed.
- Enhanced Screen Swapping is available when using more than two logical screens. Navigation among the screens can now be supplemented by Point and Shoot fields at bottom of the screen. This new support is intended to make it easier to navigate within an ISPF session and to supplement the existing SWAP LIST and SWAP NEXT support.
- The ISPF Data Set List panel (Option 3.4) is enhanced to support block commands. This is intended to enable you to act on multiple data sets using fewer line commands; for example, using paired “DD” commands will specify that a block of data sets is to be deleted.
- The VARY command is enhanced to support a new device attribute, UNAVAILABLE, for tape devices. This support, which supplements the OFFLINE status that can already be set for any device, allows you to specify that UNAVAILABLE devices be excluded from recovery allocation processing and from message IEF877E, which lists eligible devices that might be used to satisfy an allocation request.

1.27 AMBLIST processing

Several enhancements were added to the AMBLIST program to allow it to be more easily used and to provide higher quality information for diagnosing problems found when developing or deploying program objects.

LISTLOAD operations

For LISTLOAD operations, the DLL Import/Export information from the B_IMPEXP class (information built by the binder in section IEWBCIE) is now formatted with MODLIST output. This new format displays the symbols, their attributes, and the DLLs they belong to, making it possible to determine dynamic bind problems which were not recognized when the module was initially built. This information contained in this section is the same information that Language Environment uses to complete dynamic linking at run-time.

For LISTLOAD operations, the information contained in the binder built module-level section X'0001', will be written with MODLIST output. This provides information about symbols that are referenced but are not used to update text, IDRU records (created by the user via the binder APIs), and the contents of the B_MAP class which contains general information about all the symbols within the module.

For LISTLOAD operations, the XREF output cross-reference information (both numerical and alphabetical) will be completely redesigned. This new design will align more closely to binder XREF output, and present the data in a much simplified, more intuitive format.

1.28 IPCS and dump processing

To help verify optimal dump reading configurations, IPCS is planned to issue additional messages when a dump is initialized, checking both the control interval size and available space of the dump directory in use. Additionally, IPCS is planned to report on the amount of time taken by the dump process.

System Trace buffers are planned to move to 64-bit storage, allowing for significantly increased system trace capacity.

AutoIPL support

AutoIPL support will provide the capability to request that the system automatically IPL a stand-alone dump, z/OS, or both, when a disabled wait state is requested by a system component. This function is designed to be under the control of new parmlib parameters and a new wait state action table (WSAT), and together, they specify the actions, if any, to be taken for various disabled wait states. In a sysplex environment, the Sysplex Failure Manager (SFM) policy can result in actions that load disabled wait states on systems to be partitioned out of the sysplex, which can also trigger AutoIPL processing. New options on the **VARY XCF** operator command will allow you to request a SADMP, z/OS IPL, or both, after the indicated system has been removed from the sysplex. AutoIPL capability is intended to help achieve faster failure data capture and recovery after system failures.

1.29 z/OS XML System Services

Support for 19 additional code pages

This extends XML System Services processing to accommodate the character sets used in many additional languages. This function is also available on z/OS V1R7, V1R8, and V1R9 with PTF for APAR OA22777.

Source offset support

This is designed to make it easier to locate or extract specific data from within an XML document.

Enterprise COBOL V4.1 (5655-S71)

This support provides for a new XMLPARSE compiler option to allow COBOL programs to use XML System Services for XML processing and take advantage of zAAP specialty processors. For more information, refer to Software Announcement 207-339, dated December 11, 2007.

1.30 Network File System

The Network File System (NFS) is a distributed file system designed to provide transparent processing capability for data and information on worldwide and heterogeneous networks. The z/OS Network File System (NFS) provides the implementation that allows the z/OS platform to participate in these networks.

In z/OS V1R10, NFS provides interoperability, continued NFS V4 support, constraint relief, and serviceability enhancements which build on previous support delivered in z/OS V1R7, V1R8, and V1R9, as follows:

- Expanded platform support for z/OS NFS Clients and Servers to interoperate with Linux on System z NFS Servers and Clients.
- Continued NFS Server and Client support for the NFS V4 standard (RFC3530). The NFS Client support provides stronger authentication and network transmission protection for NFS data via the use of the RPCSEC_GSS security authentication flavor.
- NFS Client and Server support for byte-range file locking using services is provided by z/OS UNIX System Services for z/OS UNIX file systems. The NFS V4 locking protocol provides some improvements over NFS V2 and V3 as it incorporates the locking operations into the same protocol as other file access operations (open, close, read, write) and allows for files, file systems, or servers to be defined to use advisory locking rules.
- NFS Client and Server support for remotely managing ACLs via the NFS V4 protocol is added to display and modify ACL values via the ACL attribute. The NFS Server will map ACL requests between the z/OS UNIX ACL definition and the NFS V4 protocol definition. The NFS Client will use UNIX APIs to manage ACLs on remote NFS servers via the NFS V4 protocol and will map ACL requests between the z/OS UNIX ACL definition and the NFS V4 protocol definition.
- Improved storage constraint relief, via 64-bit support, is provided for the NFS client enabling utilities (mvslogin, mvslogout, and showattr) on other platforms (AIX®, Sun™, Linux on POWER5™, and Linux on System z).
- Enhanced NFS Server message support with Japanese National Language Support (NLS).

1.31 Server Message Block

When you start Server Message Block (SMB), the SMB server can now validate the syntax of the SMB environment variables. This can help to avoid errors and ensure that SMB uses the configuration options you intend.

A new environment variable, _IOE_SMB_TRANSPORTS, enables SMB clients to specify which port will handle SMB calls. The server is designed to respond on the enabled ports; the client software can chose to attempt one protocol prior to the other or both in parallel.

1.32 TotalStorage Productivity Center for Replication BE

z/OS V1R10 announces a new product, TotalStorage Productivity Center for Replication Basic Edition (BE) for System z V3.4. A new Basic HyperSwap capability (to be enabled by TotalStorage Productivity Center for Replication BE) plans to provide a low-cost, single-site, high-availability disk solution that allows the configuration of disk-replication services using an

intuitive GUI from z/OS. A future release of TotalStorage Productivity Center for Replication for System z intends to provide Basic HyperSwap administration as well.

The new Basic HyperSwap capability allows the configuration of disk-replication services using an intuitive GUI from z/OS. The intention is that with Basic HyperSwap function enabled, seamlessly swapping between primary and secondary disk volumes in the event of planned and unplanned outages such as hardware maintenance, testing, or device failure can be accomplished from z/OS.

Basic HyperSwap, enabled through TotalStorage Productivity Center for Replication BE, is intended to help you eliminate single disk failures as a source of application outages by allowing you to specify a set of storage volumes to be synchronously mirrored. For example, in the event of a permanent I/O error, I/O requests can be automatically switched to the secondary copy, thereby masking the failure from the application and minimizing the need to restart the application (or system) after the failure. You can also initiate a planned failover to a secondary for the purpose of initiating hardware maintenance on primary storage controllers, or simply to periodically test the function. You can switch back to your preferred configuration via the GUI or operator commands.

Required prerequisites

- z/OS V1R9 with appropriate service, or z/OS V1R10.
- A new planned, no-charge product, IBM System Services Runtime Environment for z/OS, which is an environment that provides native Web services for z/OS. You may use WebSphere Application Server 6.1.0 for the IBM System Services Runtime Environment for z/OS.
- DB2 V8 (or later). Customers without DB2 may use Apache Derby (available with TotalStorage Productivity Center for Replication BE).
- IBM storage controller with the Advanced Copy Feature Metro Mirror. This includes IBM ESS Models 800, DS6000™, and DS8000.

GDPS/PPRC HyperSwap Manager

GDPS/PPRC HyperSwap Manager (GDPS/PPRC HM) provides a robust continuous availability disk management solution, as well as an entry level disaster recovery solution when used across multiple sites. Basic HyperSwap will not be a replacement of GDPS/PPRC HyperSwap Manager. Customers desiring the comprehensive high-availability, multi-site, disaster-recovery capabilities of GDPS are still advised to investigate one of the GDPS solutions.

2

Installation considerations

This chapter provides some planning information for a migration to z/OS V1R10.

When planning to install this new system, an objective is to make it functionally compatible with the previous system. When considering the migration, the applications and resources on the new system need to function the same way (or similar to the way) they did on the old system or, if that is not possible, in a way that accommodates the new system differences so that existing workloads can continue to run. Migration does not include exploitation of new functions except for new functions that are now required.

During this migration, consider what needs customizing and what programming is necessary to take advantage of (exploit) the enhancements available in the new release.

This chapter describes some of the considerations needed to successfully migrate from either of the two releases that are supported for direct migration to z/OS V1R10:

- z/OS V1R9
- z/OS V1R8

If you want to migrate to z/OS V1R10 from any other release, contact your IBM representative to find out what alternatives are available.

2.1 Installing z/OS V1R10

The program number for z/OS Version 1 Release 10 is 5694-A01. When ordering this program number, remember to order all the optional features that you were licensed for in previous releases of z/OS.

In z/OS V1R10 there are only two export controlled unpriced features, as follows:

- z/OS Security Level 3, and Communications
- z/OS Security Level 3 that contains the subelements, as follows:
  - IBM Tivoli Director Server Security Level 3
  - Network Authentication Service Level 3
  - OCSF Security Level 3
  - System Secure Sockets Layer (SSL) Security Level 3

Ordering z/OS V1R10

z/OS V1R10 can be ordered beginning on September 12, 2008. General availability for z/OS V1R10 is September 26, 2008 and it becomes generally availability via ServerPac, CBPDO and SystemPac®. This is the first date for ordering z/OS V1R10 ServerPac, SystemPac, and CBPDO using CFSW configuration support, or ShopzSeries, the Internet ordering tool.

Typically, when one new z/OS release becomes orderable in ServerPac, SystemPac, and CBPDO, the previous release is orderable for only a month. Due to this short overlap, it is very important that you order the z/OS release you need for migration and coexistence while it is still available for ordering. Note that z/OS V1R9 ServerPac is no longer orderable, while SystemPac (a fee deliverable) is still available.

Web delivery

Sometimes enhancements are provided as Web deliverables, as follows:

- Cryptographic support for z/OS V1R7-R9 and z/OS.e V1R7-R8

  The Cryptographic support for the z/OS V1R7-R9 and z/OS.e V1R7-R8 Web deliverable contains FMID HCR7750. This FMID is included in z/OS V1R10 Cryptographic Services ICSF.
- System REXX Support for z/OS V1R8 & z/OS.e V1R8

These two Web deliverables were generally available in 2007. z/OS and z/OS.e Web deliverables are available from:

`http://www.ibm.com/eserver/zseries/zos/downloads/`

They are packaged as two files that you download:

- A readme file, which contains a sample job to uncompress the second file, transform it into a format that SMP/E can process, and invoke SMP/E to RECEIVE the file. This file must be downloaded as text.
- A pax.z file, which contains an archive (compressed copy) of the FMIDs to be installed. This file needs to be downloaded to a workstation and then uploaded to a host as a binary file.

2.2 Elements, features, and FMID changes in z/OS V1R10

There are no new elements added in z/OS V1R10. No elements have been withdrawn from z/OS V1R10.

In z/OS V1R10, the SDSF element added a new FMID JJE775S which contains SDSF JES2 support. This new FMID installs in new libraries.

If you plan to use z/OS V1R10 SDSF with z/OS V1R10 JES2, FMID JJE775S must be installed with the JES2 FMID HJE7750 and the SDSF FMID HQX7750.

In z/OS V1R10, the NFS element added a new FMID JDZ1AJ0 that provides Japanese support.

SMP/E for z/OS V3R5 (5655-G44)

Beginning with z/OS V1R2 (which was SMP/E V3 R1), SMP/E is non-exclusive because of the introduction of the SMP/E stand-alone product. SMP/E for z/OS is available under its own product number and also remains a base element of z/OS. This allows customers who are licensed for a currently supported release of z/OS to order and install the latest release of SMP/E without having to upgrade their entire operating system. The advantage is that products other than z/OS can exploit the packaging and installation enhancements in SMP/E without having to install the prerequisites for a new level of the operating system.

In addition, since SMP/E plays a key role in Internet delivery of software, it allows IBM to exploit the Internet delivery and installation technologies in SMP/E sooner without having to wait for customers to migrate to new levels of the operating system. SMP/E for z/OS is available at no additional charge to customers. It is intended for customers who have a license for z/OS V1 (5694-A01).

You may want to consider ordering this separate product early (in a CBPDO) in order to take advantage of the new functions in SMP/E right away.

2.3 Target and driving system

Driving system requirements for installing z/OS by way of ServerPac or dump-by-data-set SystemPac are, as follows:

- An operating system that includes any of the following:
  - z/OS V1R8 or later, or z/OS.e V1R8. If V1R8, DFSMSdfp PTF UA34953 (for APAR OA20599) is also required.
  - z/OS V1R9
- The Customized Offerings Driver V2.4.1 (5655-M12) or later

If you are migrating to z/OS V1R10 from z/OS V1R9 or you will have a different product set than your previous release, you will see an increased need for DASD, as shown in Figure 2-1 on page 32. How much more depends on what levels of products you are running. Keep in mind what the DASD required for your z/OS system includes (per the z/OS Policy). That is, it includes *all* elements, *all* features that support dynamic enablement, regardless of your order, and *all* unpriced features that you ordered. This storage is in addition to the storage required by other products you might have installed. All sizes include 15% freespace to accommodate the installation of maintenance.

The total storage required for z/OS data sets is listed in the space tables in the z/OS Program Directory.

	z/OS V1R9	z/OS V1R10
Target	6400	6400
DLIB	8900	9000
File system	2900	2900

Figure 2-1 DASD space requirements for z/OS V1R10

z/OS V1R10 DASD space

For z/OS V1R10, the space requirements are as follows:

- The total storage required for all the target data sets is 6400 cylinders on a 3390 device.
- The total storage required for all the distribution data sets listed in the space table is 9000 cylinders on a 3390 device.
- The total file system storage is 2,900 cylinders on a 3390 device for the ROOT file system, 50 cylinders for the /etc file system and 50 cylinders for the VARWBEM file system (for CIM element).
- The total storage required for the SMP/E SMPLTS is 0 3390 cylinders (there are no load modules in z/OS V1R10 that are both cross-zone and use CALLLIBs, thus the SMPLTS is not needed for permanent storage).

z/OS and z/VM

z/OS can run as a guest of z/VM®. Requirements are:

- If your server is a z9 EC or z9 BC server, the z/VM PTFs for the following APARs must be installed:
  - On z/VM V5 (5741-A05): PTFs for APARs OA11650, VM63646, VM63721, VM63722, VM63740, VM63743, and VM63744.
  - On z/VM V4R4 (5739-A03): PTFs for APARs VM63124, VM63646, VM63721, VM63740, and VM63743. Also, if z/VM is using QDIO Guest LAN to deploy guest LANs/VSWITCHs using VLANs, the PTF for APAR VM64359 must be installed on the z/VM V4R4 system.
- If your server is a z990 or z890 server, the z/VM PTF for APAR VM63124 must be installed. Also, if z/VM is at V4R4 using QDIO Guest LAN to deploy guest LANs/VSWITCHs using VLANs, the PTF for APAR VM64359 must be installed on the z/VM V4R4 system.
- If your server is a z900 or z800 server, there are no special requirements.

2.4 Coexistence-Migration-Fallback policy

Coexistence and fallback are important parts in planning for migration to z/OS V1R10. Coexistence occurs when two or more systems at different software levels share resources. The resources could be shared at the same time by different systems in a multi-system configuration, or they could be shared over a period of time by the same system in a single-system configuration.

Examples of coexistence are:

- Two different JES releases sharing a spool
- Two different service levels of DFSMSdfp sharing catalogs
- Multiple levels of SMP/E processing SYSMODs packaged to exploit the latest enhancements
- An older level of the system using the updated system control files of a newer level (even if new function has been exploited in the newer level).

Coexistence for z/OS V1R10

z/OS V1R8, z/OS V1R9, and z/OS V1R10 are supported for coexistence, fallback, and migration. z/OS V1R8 is the lowest release supported for coexistence, fallback and migration with z/OS V1R10, as shown in Figure 2-2:

- You can migrate from z/OS V1R8 and z/OS V1R9 to z/OS V1R10.
- You can fallback from z/OS V1R10 to z/OS V1R9 and z/OS V1R8.

JES releases with z/OS V1R10

You should migrate to the JES2 or JES3 that comes comprehensively-tested with z/OS V1R10 at the same time you migrate to the rest of z/OS V1R10, or as soon as possible thereafter. In this way, you benefit directly from the new functions in the z/OS V1R10 level of JES2 or JES3 and enable other elements and features to benefit from this level. However, because such a migration is not always practical, certain prior levels of JES2 and JES3 are supported with z/OS V1R10 so that you can stage your migration to z/OS V1R10 (that is, migrate your JES2 or JES3 later).

Use lower releases of JES2 and JES3 with z/OS V1R10, as follows:

- z/OS V1R8 JES2 and z/OS V1R9 JES2 can be used with z/OS V1R10
- z/OS V1R8 JES3 and z/OS V1R9 JES3 can be used with z/OS V1R10

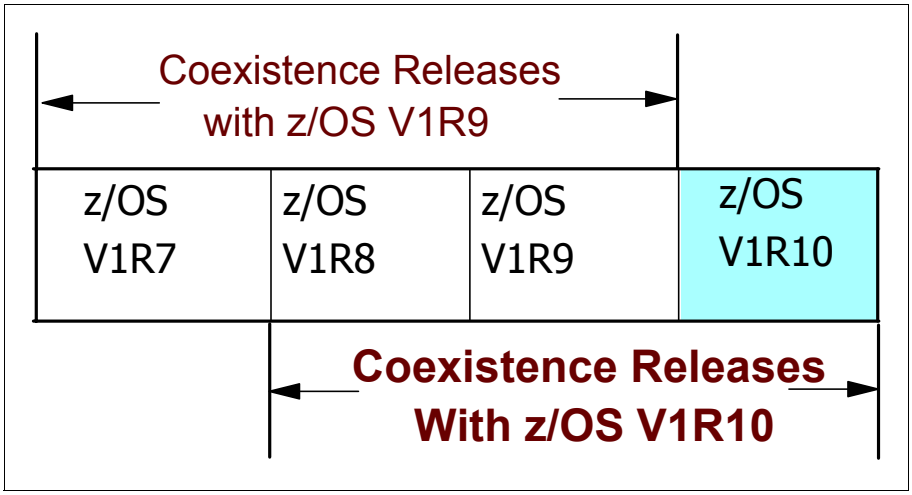


Figure 2-2 Coexistence with z/OS V1R10

JES2 V1R10 and SDSF

Starting with z/OS V1R9, the SDSF release level must be the same as the JES2 release level. With z/OS V1R10, the SDSF release level must be the same as the JES2 release level; see Table 2-1 on page 34.

Table 2-1 Allowable BCP, JES2, and SDSF combinations

BCP Release	JES2 Supported Release Allowed	SDSF Supported Release Allowed
z/OS V1R10	z/OS V1R8	z/OS V1R8
z/OS V1R10	z/OS V1R9	z/OS V1R9
z/OS V1R10	z/OS V1R10	z/OS V1R10

JES3 V1R10 and SDSF

With z/OS V1R10, SDSF is now supported with JES3. Before z/OS V1R10, SDSF was supported only with JES2. The JES3 and SDSF releases allowed with z/OS V1R10 BCP are shown in Table 2-2.

Table 2-2 Allowable BCP, JES3, and SDSF combinations

BCP Release	JES3 Supported Release Allowed	SDSF Supported Release Allowed
z/OS V1R10	z/OS V1R8	Not allowed with JES3
z/OS V1R10	z/OS V1R9	Not allowed with JES3
z/OS V1R10	z/OS V1R10	z/OS V1R10

2.4.1 Coexistence maintenance

There are many PTFs needed for coexistence and fallback service that must be installed specifically on a z/OS V1R9 system for coexistence with z/OS V1R10. Following are the coexistence and fallback service PTFs needed on a z/OS V1R9 system.

BCP coexistence PTFs

- UA39716 (APAR OA17252) allows a z/OS V1R9 system to tolerate the WLM policy changes and related control block changes that have been introduced in z/OS V1R10.
- UA39319 (APAR OA20962) causes the z/OS V1R9 program management binder to reject a PO5 PMAR level with the new z/OS V1R10 PO sublevel.
- UA39760 (APAR OA18204) allows a z/OS V1R9 system to tolerate changes made in z/OS V1R10 to console data that is passed between members of a sysplex.
- UA39874 (APAR OA23153) allows a z/OS V1R9 system to tolerate changes made in z/OS V1R10 to resource recovery services (RRS) as RRS saves data during startup.

DFSMSdfp coexistence PTFs

- APAR OA22247 is required because in z/OS V1R10, multiple lock structures are supported. The PTF prevents a z/OS V1R9 system from opening a data set that z/OS V1R10 has already opened to a secondary lock structure. Current sharing can still be performed by using the current lock structure, IGWLOCK00. The PTF also provides toleration of new shared structures and communication that is done between SMSVSAM instances.
- UA39934 (APAR OA22026) causes a z/OS V1R9 system to do the following:
  - Fail OSREQ DELETE and OSREQ RETRIEVE requests for objects greater than 268,435,456 bytes.
  - Skip OSMC processing of objects greater than 268,435,456 bytes.
  - Recognize the new P value in the FULL column of the TAPEVOL and VOLUME tables as permanently marked full.

  - Recognize ONLYIF statements as valid in the CBROAMxx member of parmlib.

- APAR OA22400 allows the new (in z/OS V1R10) P volume full value to be displayed as “yes” in the volume full column on the Mountable Optical Volume List ISMF panel (DGTLGP31) of a z/OS V1R9 system.
- APAR OA22449 is needed because in z/OS V1R10, the LSPACE macro's parameter list has been extended to support extended address volumes (EAVs). The PTF allows a z/OS V1R9 system to tolerate the change.
- APAR OA21487 allows a DEVSERV QDASD command issued on z/OS V1R9 to report the actual size of an extended address volume even though the volume cannot be brought online. Extended address volume support is introduced in z/OS V1R10.

2.5 Functions withdrawn in z/OS V1R10

Figure 2-3 on page 36 lists items that are withdrawn in z/OS V1R10. These items were last shipped in z/OS V1R9. You should take this into account as you plan your migration to z/OS V1R10. The removal of these functions may have migration actions which you can perform now, in preparation for z/OS V1R10. See *z/OS Migration*, GA22-7499 for the migration actions. A description of this function withdrawal is as follows:

- English and Japanese ISPF panels in DFSORT were withdrawn in z/OS V1R10. There will be no replacement for this limited interactive facility. For many years these panels were not enhanced, and they no longer matched the new functions added to DFSORT. Support for JCL to sort, copy, or merge will continue to be available.
- Traffic Regulation (TR) policy as part of Quality of Service (QoS) policy type was withdrawn from Communications Server in z/OS V1R10. Migrate from Quality of Service (QoS) TR policies to Intrusion Detection Services (IDS) TR policies. The TR policy function is still available but only as part of the Intrusion Detection Services (IDS) policy type. Note that this change is only for the TR policy configuration. The TR policy functions themselves remain unaffected and continue to run.
- RMF LDAP backend was withdrawn in z/OS V1R10. The RMF LDAP interface currently allows access to RMF performance data from application programs. In its place you can use the Common Information Model (CIM) monitoring interface, which has been provided since z/OS V1R7.
- IBM has removed support for CPU affinity as of z/OS V1R10. Beginning with z/OS V1R10, any attempt to assign CPU affinity is ignored.

DFSORT ISPF English and Japanese panels	Priced Feature – No replacement for this limited interactive facility	z/OS V1R10
Traffic Regulation (TR) policy as part of Quality of Service (QoS) policy type	Communications Server element – In z/OS V1R10, TR policy function is available only as part of Intrusion Detection Services (IDS) TR policies. Migrate to IDS TR policies	z/OS V1R10
RMF LDAP backend	Optional priced feature – Use CIM monitoring interface (available since z/OS V1R7)	z/OS V1R10
CPU affinity	BCP element – In z/OS V1R10, CPU affinity is ignored	z/OS V1R10

Figure 2-3 Functions withdrawn in z/OS V1R10

2.6 Functions to be withdrawn in the future

Figure 2-4 on page 37 lists the items that IBM has announced it intends to remove in a future z/OS release. You are encouraged to consider these removals when making your plans for system upgrades. These statements represent IBM's current intentions. IBM development plans are subject to change or withdrawal without further notice.

Network Database

z/OS V1R10 is planned to be the last release in which z/OS Communications Server will support the Network Database (NDB) function. If you currently use or plan to use the NDB function, you should investigate the distributed data facility (DDF) provided by z/OS DB2, and the DB2 Run-Time Client.

Bind DNS 4.9.3

z/OS V1R10 is planned to be the last release in which z/OS Communications Server will support BIND DNS 4.9.3. If you use BIND DNS 4.9.3, you should implement BIND DNS 9.2.0 as a replacement. BIND DNS 9.2.0 is included in z/OS beginning with z/OS V1R4.

Boot Information Negotiation Layer

z/OS V1R10 is planned to be the last release in which z/OS Communications Server will support Boot Information Negotiation Layer (BINL). If you use this function, consider using IBM Tivoli Provisioning Manager for OS Deployment V5 (5724-Q99).

Dynamic Host Configuration Protocol

z/OS V1R10 is planned to be the last release in which z/OS Communications Server will support Dynamic Host Configuration Protocol (DHCP) server. If you use this function, you should investigate using a DHCP server on Linux for System z if you want to continue to run the DHCP server on System z hardware.

zFS multi-file system aggregates

In a future release, IBM plans to withdraw support for zFS multi-file system aggregates. When this support is withdrawn, only zFS compatibility mode aggregates will be supported. (A zFS compatibility mode aggregate has a single file system per data set.)

Integrated Security Services LDAP Server

If you are currently using Integrated Security Services LDAP Server, it is recommended that you migrate to IBM Tivoli Directory Services (IBM TDS) because IBM TDS is designed to be an improvement over LDAP Server and because IBM is enhancing only IBM TDS. This support is planned to be removed in a release after z/OS V1R10.

Some VSAM data set attributes

No supported release of z/OS honors the IMBED, REPLICATE, and KEYRANGE attributes for new VSAM data sets. IBM recommends that you stop using IMBED and REPLICATE, and that you minimize or eliminate your use of KEYRANGE. Striped data sets provide much better performance than KEYRANGE and should be viewed as a candidate for any existing KEYRANGE data sets. The recommendation to migrate from IMBED, REPLICATE, and KEYRANGE was originally made in the z/OS V1R6 time frame.

Network Database (NDB) function (from Communications Server)	Base element – Use the distributed data facility (DDF) provided by z/OS DB2, and the DB2 Run-Time Client	Release after z/OS V1R10
Boot Information Negotiation Layer (BINL) **(from Communications Server)**	Base element – investigate using IBM Tivoli Provisioning Manager for OS Deployment	Release after z/OS V1R10
Bind DNS 4.9.3 (from Communications Server)	Base Element – implement BIND DNS 9.2.0 as a replacement (available since z/OS V1R4)	Release after z/OS V1R10
DHCP server (from Communications Server)	Base Element – investigate using a DHCP server on Linux for System z	Release after z/OS V1R10
zFS multi-file system aggregates (from DFS)	Base Element – use zFS compatibility mode aggregates	Future release
Integrated Security Services LDAP Server	Base Element – recommend migrating to IBM Tivoli Directory Server for future enhancements	Release after z/OS V1R10
VSAM data sets that contain IMBED, REPLICATE, KEYRANGE **Attributes (DFSMS)**	Base Element - IBM recommends that you stop using IMBED, REPLICATE, KEYRANGE since z/OS V1R6	Future release

Figure 2-4 Functions to be withdrawn in the future

2.7 Migrating to a System z10 server

The IBM System z10 server is a follow-on to the IBM System z9 servers (z9 EC [formerly z9-109] and z9 BC) and IBM eServer™ zSeries® servers (z990, z890, z900, and z800). The System z10 server builds on the inherent strengths of the System z platform, delivers new technologies that offer dramatic improvements in price and performance for key new workloads, and enables a new range of hybrid solutions.

Table 2-3 on page 39 indicates which z/OS releases are supported for the new system functions and that can be exploited with the z10 EC server, as follows in the next sections.

2.7.1 HiperDispatch

A new HIPERDISPATCH=YES|NO parameter in the IEAOPTxx parmlib member, and on the SET OPT=xx command, controls whether HiperDispatch is enabled or disabled for the system. The value can be changed dynamically. HiperDispatch defaults to disabled on z/OS V1R7 through V1R9. Thus, by default, your environment is not changed from a HiperDispatch perspective when migrating from a pre-System z10 server to a System z10 server. Once migration has completed, you can exploit the HiperDispatch function of the System z10 server. Because HiperDispatch improves the performance of a System z10 system, a new health check (SUP_HIPERDISPATCH) was added to verify that HiperDispatch is enabled. The new health check is only added on System z10 systems. WLM goal adjustment might be required when using this function. Review and update your WLM policies as necessary. You might need to turn off and on HiperDispatch while adjusting your WLM goals.

2.7.2 Capacity provisioning

Connectivity to the managed systems is by TCP/IP. The function requires the CIM Server to be active. For more information about exploiting this enhancement, see *z/OS MVS Capacity Provisioning User's Guide*, SC33-8299.

2.7.3 Large page (1 MB) support

Large page support is not enabled without the software support. Page frames will be allocated at the current 4 K size without the large page support. There is a new LFAREA=xx%|xxxxxxM|xxxxxxG parameter in the IEASYSxx parmlib member, which specifies the amount of online real storage at IPL that is to be used to back large pages. This parameter cannot be changed dynamically.

2.7.4 C/C++ ARCH(8) and TUNE(8) options

The ARCHITECTURE option of the XL C/C++ compiler selects the minimum level of machine architecture on which your programs will run. Certain features provided by the compiler require a minimum architecture level. ARCH(8) exploits instructions available on System z10 servers.

The TUNE compiler option allows you to optimize your application for a specific machine architecture within the constraints imposed by the ARCHITECTURE option. The TUNE level must not be lower than the setting in the ARCHITECTURE option. For more information, refer to the TUNE compiler option in:

z/OS XL C/C++ User's Guide, SC09-4767

ARCH(7) level

ARCH(7) is the minimal level required to exploit the Decimal Floating Point support. The resultant program objects can execute on z9 EC and z9 BC servers, as well as on the new z10 EC servers. Also note that on z9 servers, the ability to use the hardware decimal floating point facilities is dependent on the MCL level of the server.

ARCH(8) level

Once programs exploit the ARCH(8) or TUNE(8) options, those programs can only run on System z10 EC servers, or an operation exception will occur. This is a consideration for programs running on different server levels (System z9 and zSeries) during development, test, and production, as well as during fallback or disaster recovery.

ARCH(8) is provided with z/OS V1R9. Instructions are provided on the Web enabling z/OS V1R8 to use ARCH(7) or ARCH(8) to exploit decimal floating point, as well as use ARCH(8)/TUNE(8) on z/OS V1R8 to generate code to use instructions only available on a z10 EC service and optimize the code for a z10 EC server.

z/OS V1R8 XL C/C++ does not support Decimal Floating Point Math, although it does support a smaller subset of ARCH(7).

You must have at least the z/OS V1R9 XL C/C++ compiler to use this function. Information about how to copy the z/OS V1R9 XL C/C++ compiler to a z/OS V1R8 system is provided at the following Web site:

http://www.ibm.com/software/awdtools/czos/

Restriction: For exploitation, this restriction applies once programs exploit the ARCH(8) or TUNE(8) option. The programs can only run on System z10 servers; otherwise, an operation exception will occur. This is a consideration for programs that will run on different server levels (System z9 and zSeries) during development, test, and production, as well as during fallback or disaster recovery.

Note: ARCH(7) is the minimum level required to exploit Decimal Floating Point Math support. The resulting program objects can run on z9 EC and z9 BC servers (depending on the MLC installed) as well as on System z10 servers.

Table 2-3 System z10 functions exploited by z/OS V1R8, z/OS V1R9, and z/OS V1R10

New System z10 functions that you can exploit	**z/OS V1R8**	**z/OS V1R9**	**z/OS V1R10**
HiperDispatch	YES	YES	YES
>128 GB Support	YES	YES	YES
>54 CPs for a single LPAR		YES	YES
Capacity Provisioning		YES	YES
Large Page (1 MB) Support		YES	YES
HiperSockets Multiple Write Facility		YES	YES
zIIP-assisted z/OS Global Mirror (XRC)		YES	YES
ARCH(8), TUNE(8)	YES***	YES	YES

Where:

*** Only if instructions at the following Web site are followed for using the R9 XL C/C++ compiler on an R8 system:

http://www.ibm.com/software/awdtools/czos/

3

Extended address volume

The extended address volume (EAV) is the next step in providing larger volumes for z/OS environments. This new support in z/OS V1R10 is implemented in DFSMS, and it also requires the DS8000 Release 4.0 Licensed Internal Code for EAV support.

In past releases of z/OS, IBM has grown volumes by increasing the number of cylinders and thus GB capacity. However, the existing track addressing architecture has limited the growth to relatively small GB capacity volumes which is causing problems for the 4-digit device number limit. With EAV in z/OS V1R10, a new architecture is implemented that provides capacities of hundreds of terabytes (TBs) for a single volume and includes an architectural maximum of 268,434,453 cylinders. The first release of EAV in z/OS V1R10 is limited to a volume with 223 GB or 262,688 cylinders.

This chapter describes the following aspects of this new EAV support:

- zArchitecture data scalability
- ESS, DS8000, and PAV
- Extended address volume
- VTOC considerations for user programs
- EAV volume selection
- Managing EAV volumes
- User program accesses to the VTOC
- SMF records and user programs
- Migration considerations for using EAV volumes
- EAV migration assistance tracker
- Data migration to EAV volumes
- Using EAV volumes
- EAV summary

3.1 zArchitecture data scalability

In the past decade, as processing power has dramatically been increasing, great care and appropriate solutions have been deployed so that the amount of data that is directly accessible can be kept proportionally equivalent. Over the years DASD volumes have increased in size by increasing the number of cylinders and thus GB capacity.

However, the existing track addressing architecture has limited growth to relatively small GB capacity volumes. This has placed increasing strain on the 4-digit device number limit and the number of UCBs that can be defined. The largest available volume is one with 65,520 cylinders or approximately 54 GB, as shown in Figure 3-1 on page 42.

Rapid data growth on the z/OS platform is leading to a critical problem for some customers, with 37% compound rate of disk storage growth between 1996 and 2007. The result is that this is becoming a real constraint for customers to growing data on z/OS. Business resilience solutions (GDPS, HyperSwap, and PPRC) that provide continuous availability are also driving this constraint.

Serialization granularity

Since the 1960s, shared DASD could be serialized through a sequence of RESERVE/RELEASE CCWs that are today under the control of GRS; see Figure 3-1 on page 42. This was a useful mechanism as long as the volume of data so serialized (the granularity) was not too great. But whenever such a device grew to contain too much data, bottlenecks became an issue.

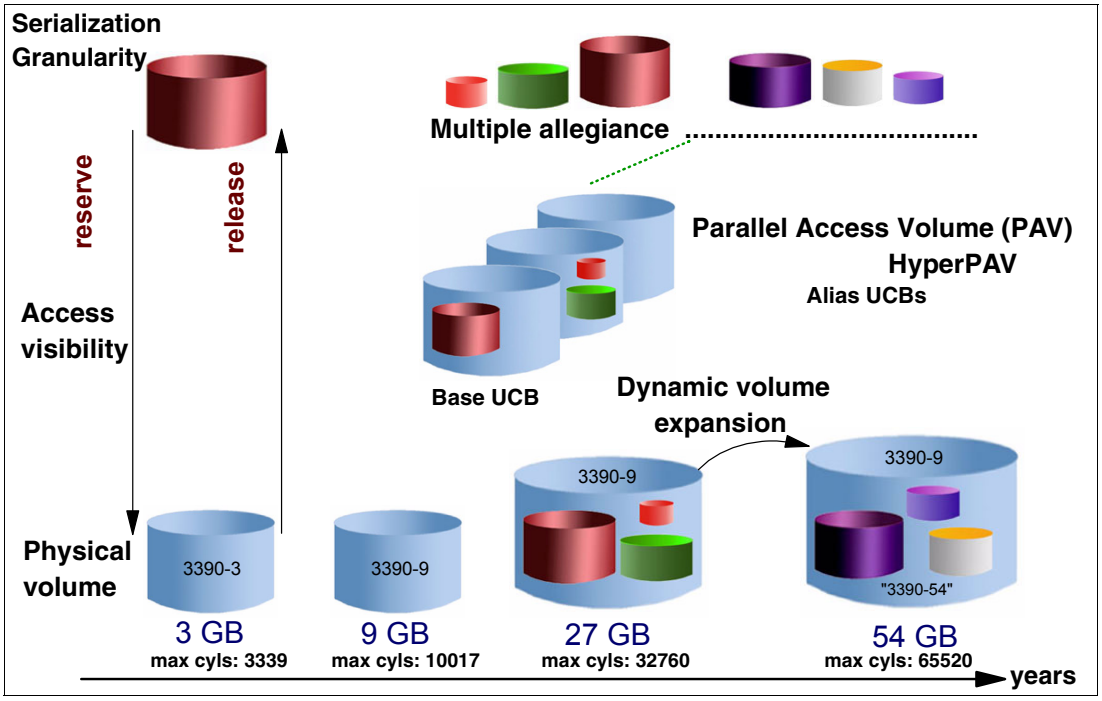


Figure 3-1 zArchitecture data scalability

DASD virtual visibility

Traditional S/390® architecture does not allow more than one I/O operation to the same S/390 device because such devices can only handle, physically, one I/O operation at a time. However, in modern DASD subsystems such as ESS, DS6000, and DS8000, the device (such as a 3390) is only a logical view. The contents of this logical device are spread in HDA

RAID arrays and in caches. Therefore, it is technically possible to have more than one I/O operation towards the same logical device. Changes have been made in z/OS (in IOS code), in the channel subsystem (SAP), and in ESS, DS6000, and DS8000 to allow more than one I/O operation on the same logical device. This is called parallel I/O, and it is available in two types:

- Multiple allegiance
- Parallel access volume (PAV)

Multiple allegiance

Multiple allegiance (MA) was introduced to alleviate the following constraint. It allows a serialization on a limited amount of data within a given DASD volume, which leads to the possibility of having several (non-overlapping) serializations held at the same time on the same DASD volume. This is a useful mechanism on which any extension of the DASD volume addressing scheme can rely.

In other terms, multiple allegiance provides finer (than RESERVE/RELEASE) granularity for serializing data on a volume. It gives the capability to support I/O requests from multiple systems, one per system, to be concurrently active against the same logical volume if they do not conflict with each other. Conflicts occur when two or more I/O requests require access to overlapping extents (an *extent* is a contiguous range of tracks) on the volume, and at least one of the I/O requests involves writing of data.

Requests involving writing of data can execute concurrently with other requests as long as they operate on non-overlapping extents on the volume. Conflicting requests are internally queued in the DS8000. Read requests can always execute concurrently regardless of their extents. Without the MA capability, DS8000 would generate a busy indication for the volume whenever one of the systems issues a request against the volume. This would cause the I/O requests to be queued within the channel subsystem (CSS).

However, this concurrency can be achieved as long as no data accessed by one channel program can be altered through the actions of another channel program.

3.2 ESS, DS8000, and PAV

The ESS and DS8000 support concurrent data transfer operations to or from the same 3390/3380 devices from the same system. A device (volume) accessed in this way is called a parallel access volume (PAV).

To implement PAV, IOS introduces the concept of *alias addresses*. Instead of one UCB per logical volume, an MVS host can now use several UCBs for the same logical volume, as shown in Figure 3-2 on page 44. Apart from the conventional base UCB, alias UCBs can be defined and used by z/OS to issue I/Os in parallel to the same logical volume device.

Currently, the Enterprise Storage Server® (ESS) allows for concurrent data transfer operations to or from the same volume on the same system using the optional feature, parallel access volumes (PAV). With ESS, you can define alias device numbers to represent one physical device, which allows for multiple I/O operations to be started at one time. However, the z/OS operating system does not allow more than one I/O operation at a time to the same device and queues additional I/O operations to the physical device.

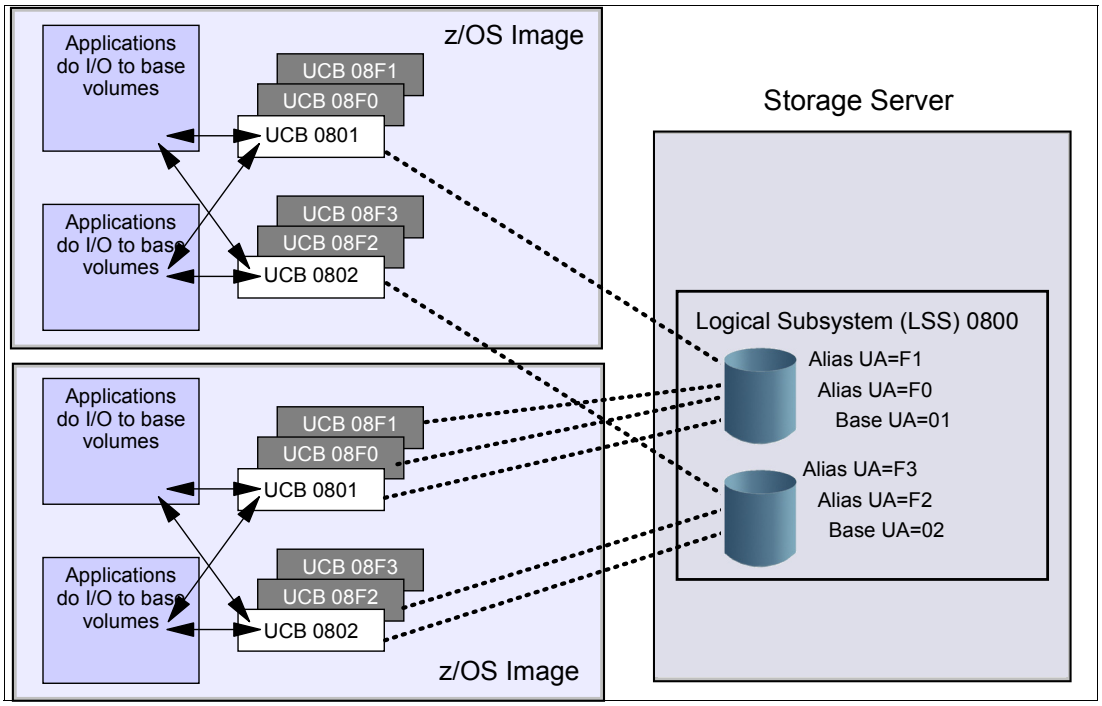


Figure 3-2 Parallel access volume

3.2.1 Parallel access volume

Parallel access volume (PAV) allows concurrent I/Os to originate from the same z/OS image. Using PAV can provide significant performance enhancements in IBM System z environments by enabling simultaneous processing for multiple I/O operations to the same logical volume.

PAV was introduced in z/OS V1R6 as a new option that allows you to specify PAV capability as one of the volume selection criteria for SMS-managed data sets assigned to a storage class.

The PAV capability option available with the storage class provides you with the following benefits:

- It eliminates I/O operation queue time for SMS-managed data sets.
- It improves performance for some kinds of data, especially DB2 data.
- It ensures that SMS-managed data sets that require high performance are automatically allocated on a volume that is using the PAV capability option.

The IOS component of z/OS systems map a device in a unit control block (UCB). Traditionally this I/O device does not support concurrency, being treated as a single resource, serially used. High I/O activity towards the same device can adversely affect performance. This contention is worse for large volumes with many small data sets. The symptom displayed is extended IOSQ time, where the I/O request is queued in the UCB. z/OS cannot attempt to start more than one I/O operation at a time to the device.

Multiple allegiance and PAV

Multiple allegiance and PAV allow multiple I/Os to be executed concurrently against the same volume, as follows:

- With multiple allegiance, the I/Os are coming from different system images.
- With PAV, the I/Os are coming from the same system image with two different implementations:
  - Static PAV: Aliases are always associated with the same base addresses.
  - Dynamic PAV: Aliases are assigned up front, but can be reassigned to any base address as need dictates by means of the dynamic alias assignment function of the Workload Manager as a reactive alias assignment.

Dynamic PAV

Dynamic PAV required WLM to monitor the workload and goals. It took some time until the WLM detected an I/O bottleneck. Then WLM had to coordinate the reassignment of alias addresses within the sysplex and the DS8000. All of this took some time, and if the workload was fluctuating or had a burst character, the job that caused the overload of one volume could have ended before WLM had reacted. In these cases, the IOSQ time was not eliminated completely.

3.2.2 HyperPAV feature

With the IBM System Storage DS8000 Turbo model and the IBM server synergy feature, the HyperPAV together with PAV, multiple allegiance can dramatically improve performance and efficiency for System z environments.

With HyperPAV technology:

- z/OS uses a pool of UCB aliases.
- As each application I/O is requested, if the base volume is busy with another I/O:
  - z/OS selects a free alias from the pool shown in Figure 3-3 on page 46, quickly binds the alias device to the base device, and starts the I/O.
  - When the I/O completes, the alias device is used for another I/O on the LSS or is returned to the free alias pool.

If too many I/Os are started simultaneously:

- z/OS will queue the I/Os at the LSS level.
- When an exposure frees up that can be used for queued I/Os, they are started.
- Queued I/O is done within assigned I/O priority.

For each z/OS image within the sysplex, aliases are used independently. WLM is not involved in alias movement so it does not need to collect information to manage HyperPAV aliases.

These implementations, built upon prior technologies, were implemented in part to help reduce the pressure on running out of device numbers. They include PAV, HyperPAV, and space-efficient FlashCopy (SEFC).

With HyperPAV, WLM is no longer involved in managing alias addresses. For each I/O, an alias address can be picked from a pool of alias addresses within the same LCU. This capability also allows different HyperPAV hosts to use one alias to access different bases. This reduces the number of alias addresses required to support a set of bases in a System z environment with no latency in targeting an alias to a base. This functionality is also designed to enable applications to achieve better performance than is possible with the original PAV feature alone, while also using the same or fewer operating system resources.

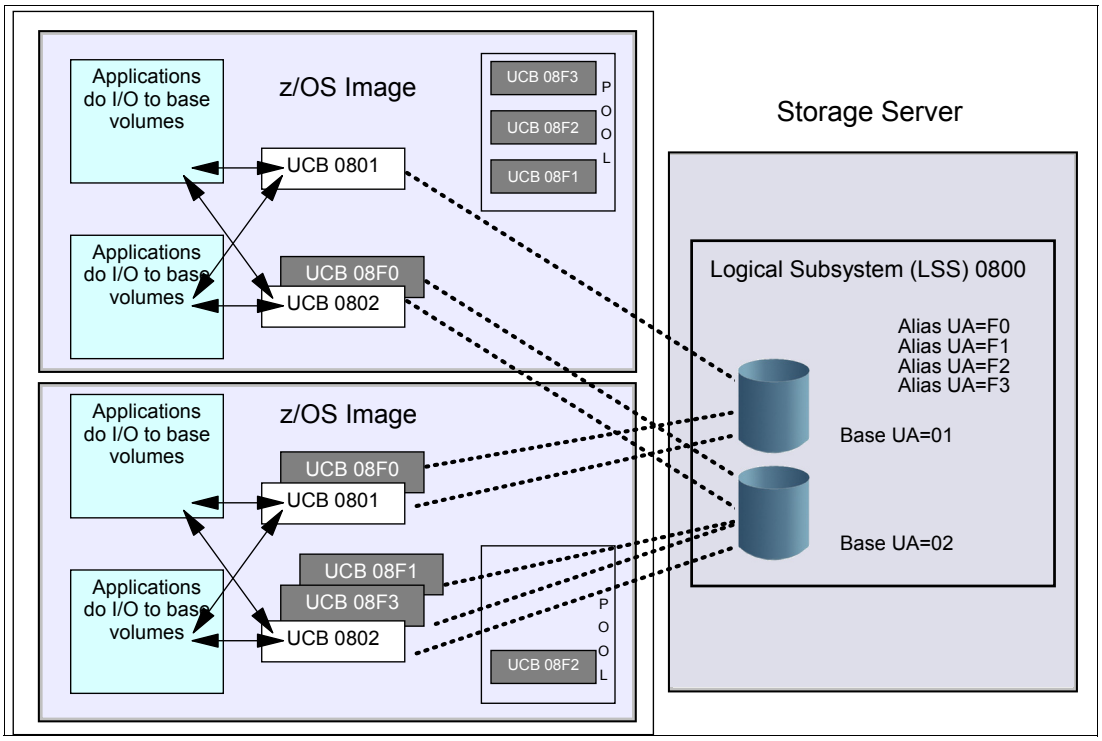


Figure 3-3 HyperPAV

Benefits of HyperPAV

HyperPAV has been designed to provide an even more efficient parallel access volume (PAV) function. When implementing larger volumes, it provides a way to scale I/O rates without the need for additional PAV alias definitions. HyperPAV exploits FICON® architecture to reduce overhead, improve addressing efficiencies, and provide storage capacity and performance improvements, as follows:

- More dynamic assignment of PAV aliases improves efficiency.
- Number of PAV aliases needed might be reduced, taking fewer from the 64 K device limitation and leaving more storage for capacity use.

Physical volumes

Historically, the term *physical volume* referred to many, quite different, hard disk technologies. Such is the case for 3390-3 and 3390-9 (for 3 GB and 9 GB). Other 3390s, though officially called 3390-9 as well, have colloquially been known as “3390-27” and “3390-54” for 3390-9s of 27 GB and 54 GB in size, as shown in Figure 3-1 on page 42.

Today, however, a physical volume is only the appearance, in accordance with the ECKD™ architecture, that a controller such as the IBM DS8000, gives of a DASD volume out of SCSI devices.

Through the years, different methods have pushed the limits of the physical size of a volume, such as with multi-volumes data sets.

More recently, Dynamic Volume Expansion is a function (available at the IBM DS8000 console) that allows you to increase the size of a given existing volume. This function triggers a signal to the z/OS system, but it is still a manual operation for the system programmer to expand the VTOC size.

3.2.3 FlashCopy SE

Functionally, IBM FlashCopy SE is not very different from the standard FlashCopy. The concept of “space efficient” regarding IBM FlashCopy SE relates to the attributes or properties of a DS8000 volume. As such, a space efficient volume could be used like any other DS8000 volume. However, the intended and only recommended use is as target volume in a FlashCopy relationship.

When a normal volume is created, it occupies the defined capacity on the physical drives. A space efficient volume does not occupy physical capacity when it is initially created. Space gets allocated when data is actually *written* to the volume, which allows the FlashCopy target volume capacity to be thinly provisioned (in other words, smaller than the full capacity of the source volume).

Therefore, when you plan for FlashCopy, you can provision less disk capacity when using FlashCopy SE than when using standard FlashCopy. This can help to lower the amount of physical storage needed by many installations.

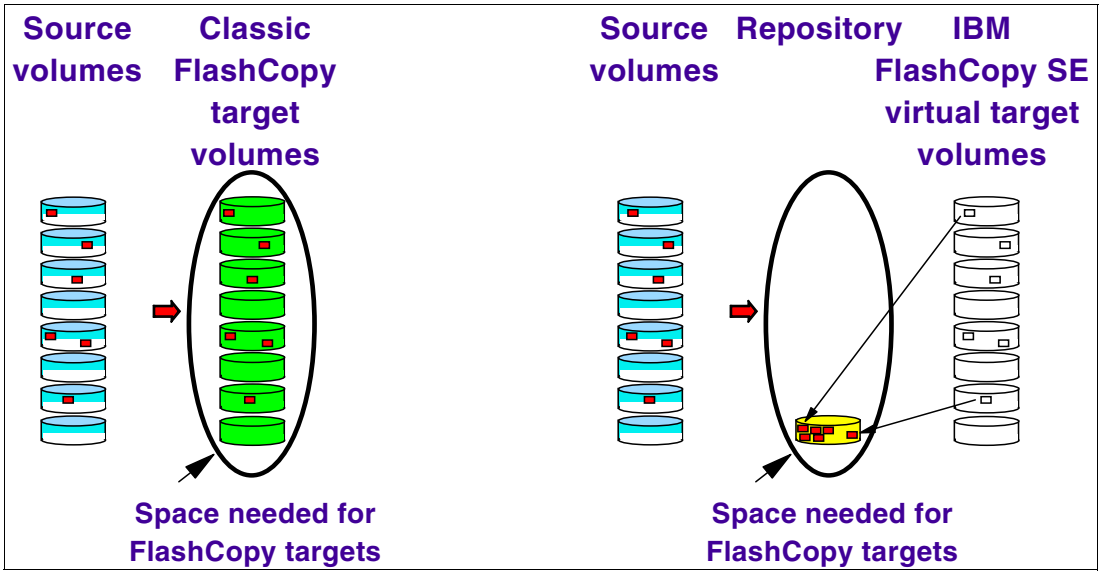


Figure 3-4 IBM FlashCopy SE concept

FlashCopy SE accomplishes space efficiency by pooling the physical storage requirements of many FlashCopy SE volumes into a common repository. A mapping structure is created to keep track of where the FlashCopy SE volume's data is physically located within the repository.

Refer to Figure 3-4 on page 47, as we take a closer look at the architecture of FlashCopy SE. The FlashCopy SE target volume itself is a *virtual* volume. It has no actual storage allocation, which is why it appears as transparent in the picture.

However, FlashCopy SE is not “free” storage. The repository is an object within an extent pool. In some sense, the repository is similar to a fully provisioned volume within the extent pool. The repository has a physical size and a logical size. The physical size of the repository is the amount of space that is allocated in the extent pool. The intended overall benefit of FlashCopy SE is that the repository is smaller than the full capacity of the FlashCopy source volumes.

The repository provides the physical space requirements for space efficient volumes. As tracks are destaged for space efficient volumes, storage for the tracks is obtained from the

segments that are assigned in the repository. The data for a space efficient volume is stored in the repository, but it is only accessible from the space efficient volume.

Important: The host operating systems do not have access to the repository.

3.3 Extended address volume

An extended address volume (EAV) is a volume with more than 65520 cylinders. An EAV increases the amount of addressable DASD storage per volume beyond 65520 cylinders by changing how tracks on ECKD volumes are addressed.

With z/OS V1R10, the IBM solution to this problem is to provide larger volumes (by about four times) by increasing the number of cylinders beyond 65520 cylinders. Volumes are grown with a larger number of cylinders (hundreds of millions) by implementing a new track addressing architecture.

EAV benefit

The benefit of this support is that the amount of z/OS addressable disk storage is significantly increased. This helps customers who are approaching the 4-digit device number limit by providing constraint relief for applications using large VSAM data sets, such as those used by DB2, CICS, zFS file systems, SMP/E CSI data sets, and NFS mounted data sets.

The extended address volume is the next step in providing larger volumes for z/OS. This support is provided with z/OS V1R10. Over the years, volumes have grown by increasing the number of cylinders and thus GB capacity. However, the existing track addressing architecture has limited the possible growth to relatively small GB capacity volumes which has put pressure on the 4-digit device number limit. The largest available volume is one with 65520 cylinders or approximately 54 GB. Access to the volumes includes the use of PAV, HyperPAV, and FlashCopy SE (space-efficient FlashCopy), as previously mentioned.

3.3.1 Extended address volume overview

With EAV volumes, an architecture is implemented that provides capacities of hundreds of terabytes for a single volume. However, the first release is limited to a volume with 223 GB or 262,668 cylinders.

3390 Model A

A volume of this size has to be configured in the DS8000 as a 3390 Model A. However, a 3390 Model A is not always an EAV. A 3390 Model A is any device configured in the DS8000 to have from 1 to 268,434,453 cylinders. 3390 Model A support is provided in DS8000 3.1 versions. The EAV support is provided in the DS8000 4.0 version. Figure 3-5 on page 49 illustrates the 3390 device type.

Note: With the 3390 Model A, the model A refers to the model configured in the DS8000. It has no association with the 3390A notation in HCD that indicates a PAV-alias UCB in the z/OS operating system. The Model “A” was chosen so that it did not imply a particular device size as previous models 3390-3 and 3390-9 did.

How an EAV is managed by the system allows it to be a general purpose volume. However, EAVs should work especially well for applications with large files. PAV and HyperPAV technologies help in both these regards by allowing I/O rates to scale as a volume gets larger.

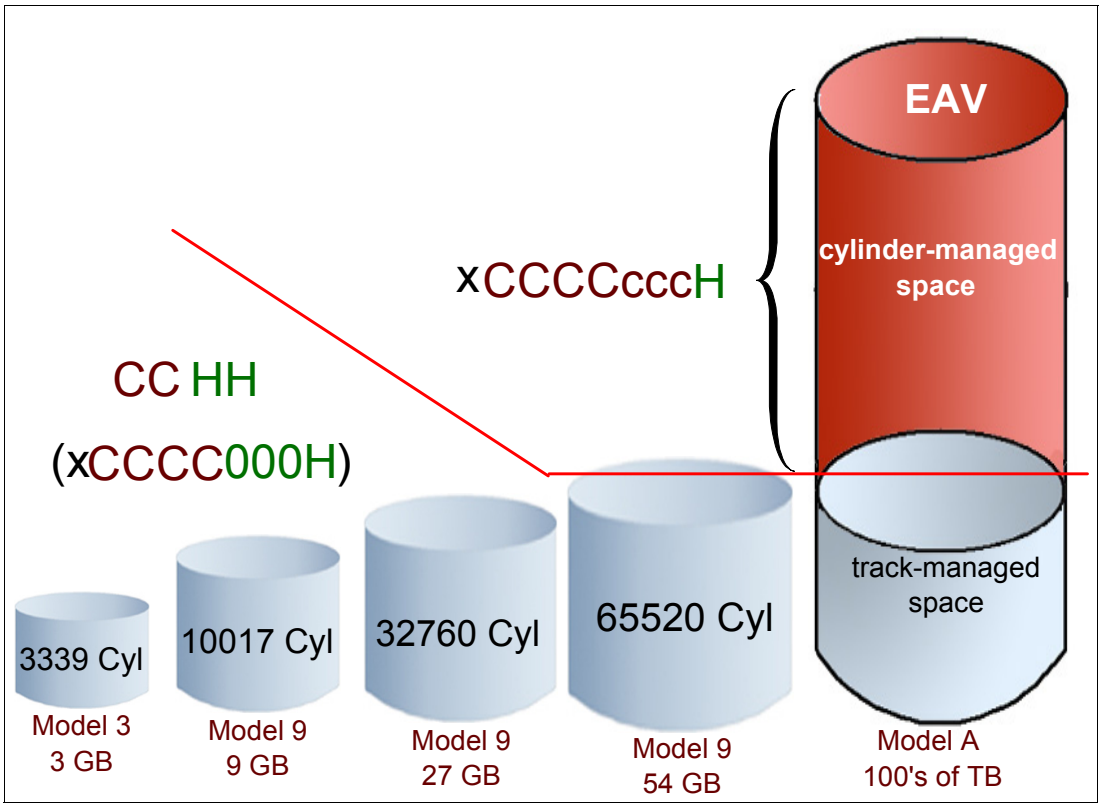


Figure 3-5 Device type 3390

EAV basic definitions

Note the following points regarding extended address volumes.

- Only 3390 Model A devices can be EAV.
- EAV is supported only in z/OS V1R10.
- The size is limited to 223 GB (262,668 cylinders).

Important: The 3390 Model A as a device can be configured to have from 1 to 268,434,453 cylinders on an IBM DS8000. It becomes an EAV if it has more than 65520 cylinders defined. With z/OS V1R10, this maximum size is currently not supported.

3.3.2 EAV key design points

One of the more important EAV design points is that IBM will maintain its commitment to customers that the 3390 track format and image size, and tracks per cylinders, will remain the same as previous 3390 model devices. An application using data sets on an EAV will be comparable to how it runs today on 3390 “numerics” kind of models.

The extended address volume has two managed spaces: the track-managed space and the cylinder-managed space.

Track-managed space

The track-managed space is the space on a volume that is managed in track and cylinder increments. All volumes today have track-managed space. Track-managed space ends at cylinder address 65519. Each data set occupies an integral multiple of tracks. The

track-managed space allows existing programs and physical migration products to continue to work. You can perform physical copies from a non-EAV to an EAV and have those data sets accessible.

Cylinder-managed space

The cylinder-managed space is the space on the volume that is managed only in multicylinder units (MCUs). Cylinder-managed space begins at cylinder address 65520. Each data set occupies an integral multiple of multicylinder units. Space requests targeted for the cylinder-managed space will be rounded up to the next multicylinder unit. The cylinder-managed space exists only on EAV volumes. A data set allocated in cylinder-managed space may have its requested space quantity rounded up to the next MCU.

Data sets allocated in cylinder-managed space are described with a new type of data set control blocks (DSCB) in the VTOC. Tracks allocated in this space will also be addressed using the new track address. Existing programs that are not changed will not recognize these new DSCBs and therefore will be protected from seeing how the tracks in cylinder-managed space are addressed.

Multicylinder unit

A multicylinder unit (MCU) is a fixed unit of disk space that is larger than a cylinder. Currently, on an EAV volume, a multicylinder unit is 21 cylinders and the number of the first cylinder in each multicylinder unit is a multiple of 21.

The 21-cylinder value for the MCU is derived from being the smallest unit that can map out the largest possible EAV volume and stay within the index architecture (with a block size of 8192 bytes), as follows:

- It is also a value that divides evenly into the 1 GB storage segments of an IBM DS8000.
- These 1 GB segments are the allocation unit in the IBM DS8000 and are equivalent to 1113 cylinders.
- These segments are allocated in multiples of 1113 cylinders starting at cylinder 65520.

Figure 3-6 on page 51 illustrates the EAV and multicylinder units.

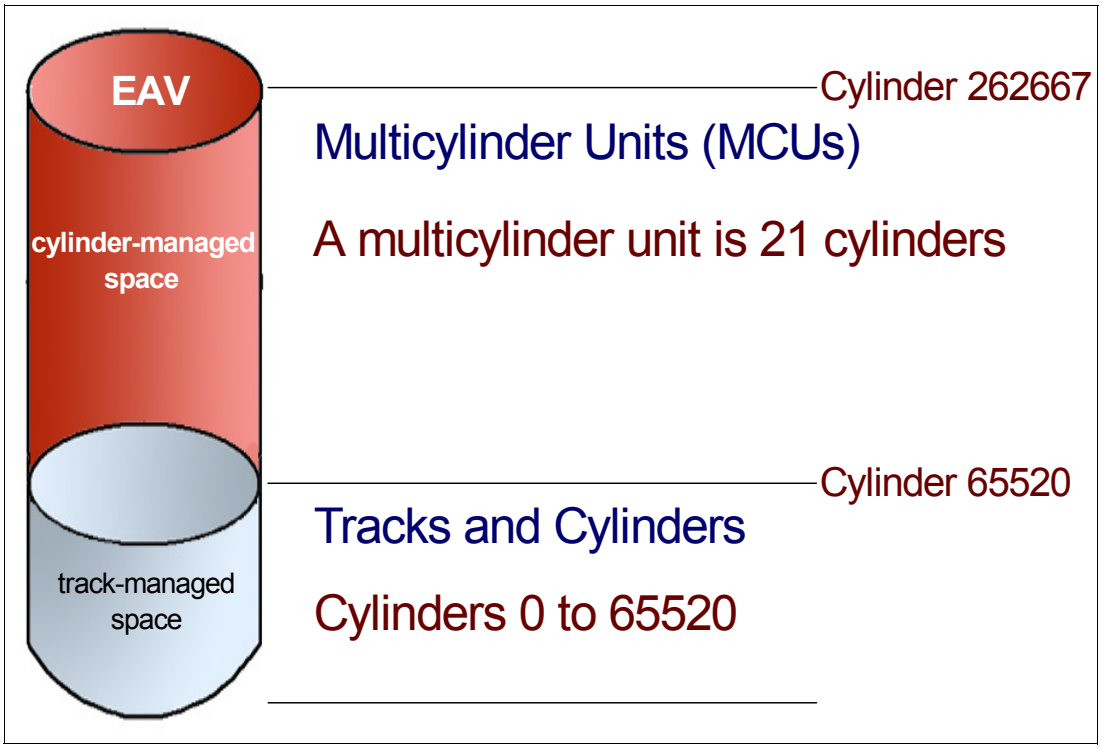


Figure 3-6 EAV and multicylinder units

3.3.3 DASD track address format

As explained, an EAV is defined to be a volume with more than 65520 cylinders. A volume of this size has to be configured in the DS8000 as a 3390 Model A. However, a 3390 Model A is not always an EAV. A 3390 Model A is any device configured in the DS8000 to have from 1 to 268,434,453 cylinders. 3390 Model A support is provided in DS8000 3.1 versions. EAV support is provided in version 4.0.

3.3.2, “EAV key design points” on page 49 discusses cylinder-managed space and track-managed space and describes how space is managed. The following sections describe how the disk is addressed using its new track address format.

Addressing extended address volumes

The extended address volume has two addressing spaces. To distinguish them from virtual storage, the term “address space” is not used.

- The base addressing space

  The base addressing space is the area on an EAV located within the first 65,536 cylinders, as shown in Figure 3-7 on page 52. Tracks are addressed in this area with 16-bit cylinder numbers, described with the CCHH notation. The CC represents 16 bits for a cylinder address. The HH represents 16 bits for a track address, of which only the low order 4 bits are used. This is how all disks are addressed today.

- The extended addressing space (EAS)

  The extended addressing space (EAS), shown in Figure 3-7 on page 52, is the area on a EAV located above the first 65,536 cylinders. Tracks are addressed in this area with 28-bit cylinder numbers, described with the CCCCcccH notation. The CCCC represents the low order 16 bits of a 28-bit number. The ccc represents the high order 12 bits of a 28-bit number. The H represents a 4-bit track number. This addressing is comparable to all

16-bit cylinder addressing. This area is similar to the cylinder-managed spaces, but is a subset of it. We often interchange the terminology of EAS and cylinder-managed space. This area is similar to the track-managed space but has a larger set of cylinders.

28-bit cylinder addressing

The 28-bit cylinder addressing architecture allows existing programs to address tracks in the base addressing space. The extended addressing space provides you with a method to protect existing programs from accessing tracks in the EAS. This is done with the new data set control blocks (DSCBs) in the VTOC, which are discussed in 3.4.2, “New format DSCBs” on page 60.

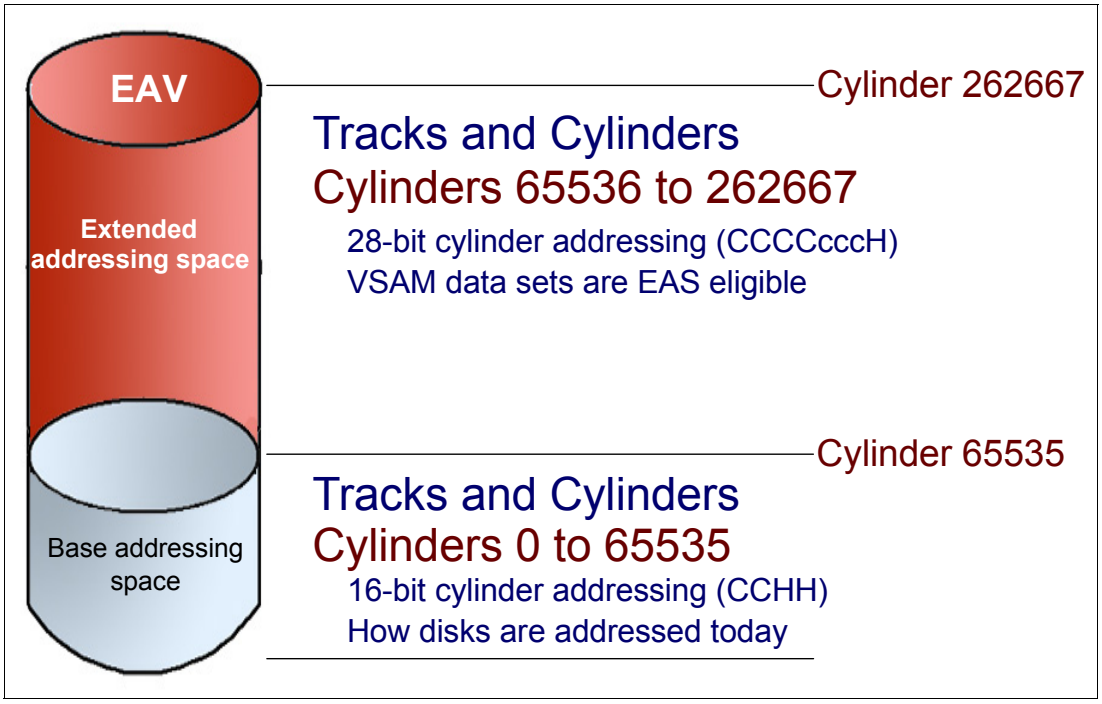


Figure 3-7 New track addressing for EAV volumes

Old track address

The base addressing space is the area on an EAV located within the first 65536 cylinders. As shown in Figure 3-8 on page 53, tracks are addressed in this area with 16-bit cylinder numbers described with the CCHH notation, as follows:

- CC represents 16 bits for a cylinder address.
- HH represents 16 bits for a track address, of which only the low order 4 bits are used.

This is how all disks are addressed today. We generally refer to a track address using the CCHH notation. However, now you can show a track address using the CCCCHHHH notation. This track address is a 32-bit number that addresses each track within a volume. Each cylinder and track number uses a 16-bit number. For the track number only the low order 4 bits are used. The high order 12 bits of the track number are not used.

Note: The maximum supported volume size today is 65,520 cylinders, which is near the 16-bit theoretical limit of 65535 cylinders.

Thus to handle cylinder numbers greater than 65,520, a new format for the track address is required.

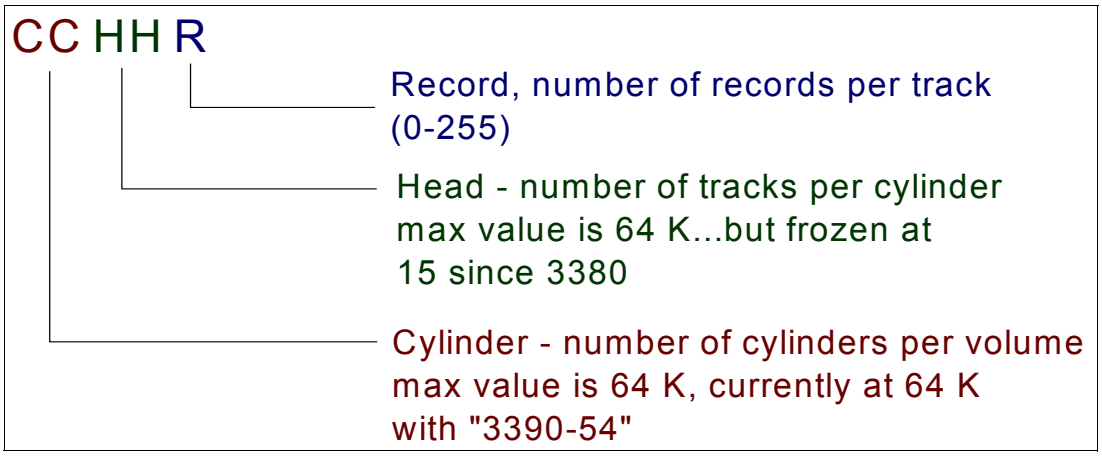


Figure 3-8 Old track address

New track address for extended address volume

The extended addressing space (EAS) is the area on an EAV located above the first 65536 cylinders. Tracks are addressed in this area with 28-bit cylinder numbers, described with the CCCCcccH (showing hex digits) notation, as follows:

- CCCC represents the low order 16 bits of a 28-bit number.
- ccc represents the high order 12 bits of a 28-bit number.
- H represents a 4-bit track number.

This addressing is comparable to all 16-bit cylinder addressing. This area is similar to the cylinder-managed spaces, but is a subset of it. We often interchange the terminology of EAS and cylinder-managed space. This area is similar to the track-managed space but has a larger set of cylinders.

The 28-bit cylinder addressing architecture allows existing programs to address tracks in the base addressing space. The extended addressing space provides you with a method to protect existing programs from accessing tracks in the EAS. This is done with the new data set control blocks (DSCBs) in the VTOC discussed in “New format DSCBs” on page 60.

For compatibility with older programs, the ccc portion is hexadecimal 000 for tracks in the base addressing space. This track address method is referred to as a 28-bit cylinder number. This format preserves the 3390 track geometry. Track addresses for space in track-managed space will be comparable to today’s track addresses. However, track addresses for space in cylinder-managed space will *not* be comparable to today’s track addresses.

Note: For compatibility reasons, the 32 bits in each track address on an EAV will be in this format: CCCCcccH. The 12 high order bits of the cylinder number are in the high order 12 bits of the two old HH bytes. This format might be written as CCCCcccH, as shown in Figure 3-9 on page 54.

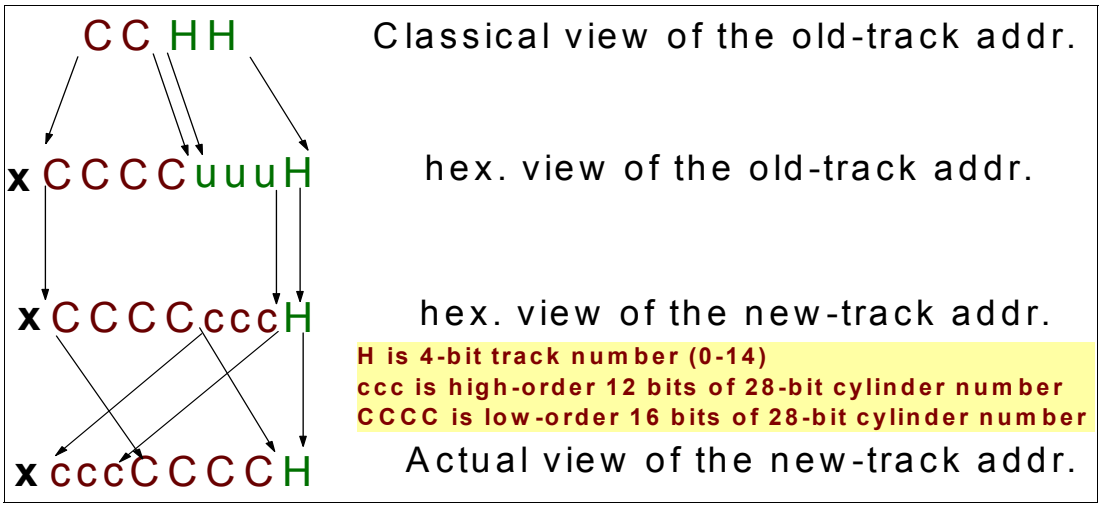


Figure 3-9 From old track address to new track address

Key points

The cylinder number is in a non-contiguous form. Reading this new track address, you must rearrange the hex digits, as shown in Figure 3-10 on page 54. This format preserves the 3390 track geometry.

Track addresses in existing channel programs and extent descriptors in DSCBs and elsewhere are in the form of CCHH, where CC is the 16-bit cylinder number and HH is the 16-bit track number in that cylinder. If the volume is an EAV, the cylinder number in these four CCHH bytes will be 28 bits and the track number will be four bits. For compatibility reasons, the 32 bits in each track address on an EAV are in this format: CCCCcccH.

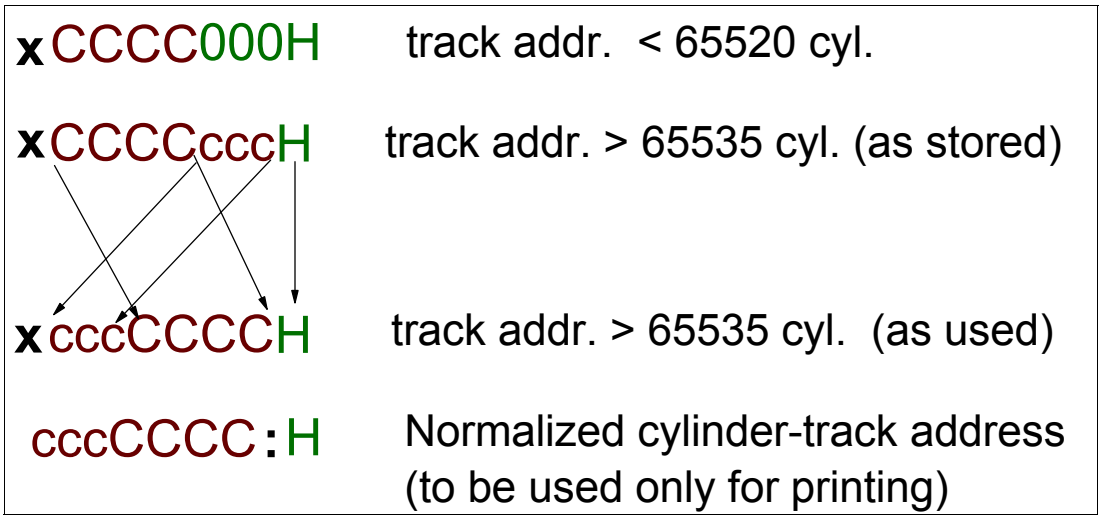


Figure 3-10 EAV: the two types of track address

3.3.4 Extended address volume attributes

In z/OS V1R10, each extended address volume, as shown in Figure 3-11 on page 55, has either all or none of the following attributes. However, in a later release, they might be independent of each other. The new volume attributes are provided in the VTOC's format 4 DSCB and the UCB's device class extension. Each of these attributes should be tested

independently. An exception is that a volume with more than 65520 cylinders requires format 8 and 9 DSCBs.

An extended address volume has all of the following three attributes. A non-extended address volume has none of these attributes.

- The volume supports an extended addressing space.
- The volume supports cylinder-managed space.
- The volumes supports extended attribute DSCBs.

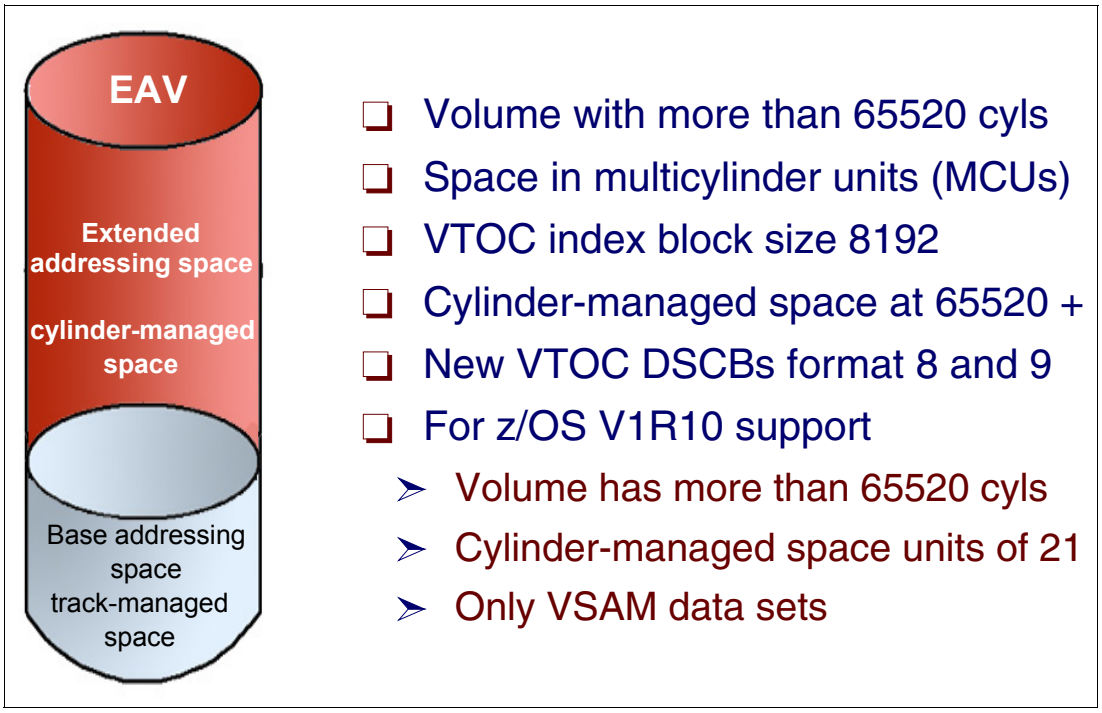


Figure 3-11 EAV volume attribute summary

Note: It is possible that in a future release a volume may just have a lower level attribute. For example, we may have a volume that supports format 8 and 9 DSCBs (extended attribute DSCBs) and the volume is smaller than an extended address volume.

3.3.5 EAS-eligible data sets

EAS-eligible data sets are defined to be those that can be allocated in the extended addressing space, which is the area on an EAV located above the first 65536 cylinders. This is sometimes referred to as cylinder-managed space. All of the following data sets can be allocated in the base addressing space of an EAV:

- SMS-managed VSAM (all types)
- Non-SMS VSAM (all types)
- zFS data sets (which are VSAM LS)
  - zFS aggregates are supported in an EAV environment
  - zFS aggregates/file systems can reside in track managed space or cylinder managed space (subject to any limitations that DFSMS may have)

  - zFS still has an architected limit of 4 TB for the maximum size of a zFS aggregate
- Database (DB2, IMS) use of VSAM
- VSAM data sets inherited from prior physical migrations or copies

Note: Only VSAM data sets that are allocated with compatible control areas (CA), for non-striped VSAM, and minimum allocation units (MAU), for striped VSAM, can reside or be extended in cylinder-managed space. A compatible CA or MAU size are those that divide evenly into the multicylinder unit of value of cylinder-managed space.

The following CA sizes and MAU are compatible because they divide evenly into the multicylinder unit of 21 cylinders (315 tracks):

1, 3, 5, 7, 9, 15 Tracks

The system ensures that, for all new allocations on all volume types, a compatible CA or MAU is selected.

EAS non-eligible data sets

An EAS-ineligible data set is a data set that may exist on an EAV but is not eligible to have extents (via Create or Extend) in the cylinder-managed space. The exceptions to EAS eligibility are as follows:

- Catalogs (BCS (basic catalog structure) and VVDS (VSAM volume data set)
- VTOC (continues to be restricted to within first 64 K-1 tracks)
- VTOC index
- Page data sets
- VSAM data sets with imbed or keyrange attributes
- Non-VSAM data sets
- VSAM data sets with incompatible CA sizes

Note: In a future release, some of these data sets may become EAS eligible. All data set types, even those listed here, can be allocated in the track-managed space on a device with cylinder-managed space on an EAV volume.

Eligible EAS data sets can be created and extended anywhere on an EAV. Data sets that are not eligible for EAS processing can only be created or extended in the track-managed portions of the volume.

3.3.6 Extended address volume free space

Currently, free space on a volume today is kept, reported, and processed by the system on a total volume basis. For a device that supports cylinder-managed space, free space information is kept, reported, and processed by the system using the total volume free space statistics and also free space statistics pertaining to the track-managed space. These two sets of free space statistics are computed and maintained in the index data set by the system for quick access for functions like the LSPACE macro for volumes that have cylinder-managed space.

Because all volumes have track-managed space, free space statistics from track-managed space are now reported on all volumes and is new with z/OS V1R10. Track-managed space statistics are the same for volumes that do not support cylinder-managed space.

The two sets of free space statistics that are returned are reported by the following functions:

```
LSPACE, ISPF, ISMF, IEHLIST, DITTO, FILEMANAGER, ADRDDSSU
```

They are recorded in an SMF type 19 record.

Free space using ISPF

By using ISPF option 3.4, entering `V` for volume on the command line, and specifying a volser (`MLDC65`), we displayed Figure 3-12 on page 57. The new information shown with z/OS V1R10 for an extended address volume is the Track Managed space shown in bold.

```
                      VTOC Summary Information
Volume . : MLDC65
Command ===> ____________________________________________________________

Unit . . : 3390              Free Space

 VTOC Data                   Total              Tracks         Cyls
 Tracks  . :       900       Size  . . :       801,746       53,327
 %Used . . :         6       Largest . :       116,115        7,741
 Free DSCBS:    42,624       Free
                             Extents . :     862

 Volume Data                 Track Managed      Tracks         Cyls
 Tracks . :     1,051,785    Size  . . :       734,966       48,875
 %Used  . :            23    Largest . :       116,115        7,741
 Trks/Cyls:            15    Free
                             Extents . :     861
```

Figure 3-12 Free space for an extended address volume using ISPF

Free space using ISMF panels

Using the VOLUME LIST panel with ISMF, Figure 3-13 on page 57 displays the available free space on the volume.

```
------------------------------------------------------------------------------
                                        VOLUME LIST
Command ===>                                                Scroll ===> HALF
                                                         Entries 1-1 of 1
Enter Line Operators below:                              Data Columns 3-8 of 43

     LINE       VOLUME FREE       %     ALLOC      FRAG   LARGEST    FREE
     OPERATOR   SERIAL SPACE      FREE  SPACE      INDEX  EXTENT     EXTENTS
    ---(1)----  -(2)-- ---(3)---  (4)-  ---(5)---  -(6)-  ---(7)---  --(8)--
                MLDC65 44364537K    76  13836972K    415   6425332K      862
    ----------  ------ -----------  BOTTOM  OF  DATA  -----------  ------  ---
```

Figure 3-13 Free space for an extended address volume using ISMF panels

Free space using IEHLIST LISTVTOC

We also ran an IEHLIST job for volume MLDC65; at the end of the output is the following information about the volume:

```
THERE ARE     53326 EMPTY CYLINDERS PLUS  1841 EMPTY TRACKS ON THIS VOLUME
THERE ARE 48874 EMPTY CYLINDERS PLUS  1841  EMPTY TRACKS FROM THE TRACK-MANAGED
```

```
THERE ARE   42623 BLANK DSCBS IN THE VTOC ON THIS VOLUME
THERE ARE   1315 UNALLOCATED VIRS IN  THE INDEX
```

Free space using ADRDSSU

Following is the output of a DEFRAG for volume MLDC65 using the new DEFRAG Version 2:

```
2008.241 14:04:12 BEGINNING STATISTICS ON MLDC65:
 *STATISTICS*                     *VOLUME*    *TRKAREA*
 FREE CYLINDERS                   00053326     00048874
 FREE TRACKS                      00001841     00001841
 FREE EXTENTS                     00000862     00000861
 LARGEST FREE EXTENT (CYL,TRK) 00007741,00  00007741,00
 FRAGMENTATION INDEX                0.415        0.431
 PERCENT FREE SPACE                    76           74
```

3.4 VTOC considerations for user programs

Be sure to check the correct EAV volume attributes in user-written programs. If these programs need to know if a volume supports extended attribute DSCBs, then check the DS4EADSCB flags and not the old volume size being x'FFFE'.

The DS4MCU field describes the minimum allocation size in cylinders for cylinder-managed space. Each extent in this space must be a multiple of this value. This is referred to as the MCU, and is the smallest unit of disk space in cylinders that can be allocated in cylinder-managed space. The field names are DS4MCU, DCEMCU, and DVAMCU.

The DS4LCYL field indicates the first cylinder address / 4095 where space is managed in multicylinder units. Cylinder-managed space begins at this address. It is valid when CYLMG is set. The field names are DS4LCYL, DCELCYL and DVALCYL.

The DS4EADSCB field indicates that extended attribute DSCBs, format 8 and format 9 DSCBs, are allowed on this volume. The field names are DS4EADSCB, DCEEADSCB, and DVAEADSCB.

The DS4CYLMG fields indicates that cylinder-managed space exists on the volume and begins at LCYL in multicylinder units of the MCU. The label names are DS4CYLMG, DCECYLMG, and DVACYLMG. Flag DS4EADSCB will also be set.

The volume size information has been relocated to new four-byte fields labeled as DCEUDCY, DS4DCYL, and DVAICYL. These will be set for all volumes. For an EAV, the number of user data cylinders in DCE (16-bit DCEUDCYL) and format 4 DSCB (16-bit DS4DSCYL) will be set to x'FFFE' (65534). This can be used to identify the volume as being an EAV. An EAV will have the DS4CYLMG and DS4EADSCB flags set.

Note: The details of these mapping changes can be found in *z/OS DFSMSdfp Advanced Services*, SC26-7400, or directly from their mappings, IECSDSL1, IECDDCE, and IHADVA.

3.4.1 Using EAV track addresses

As a consequence of this new scheme for track addresses, the following examples show how the new track address is determined:

- x'FFF0000E' is for - Cylinder 65520 - track 14
- x'0000001E' is for - Cylinder 65536 - track 14

TRKADDR macro

IBM recommends using the new TRKADDR macro for all track address comparisons and calculations. Programs should not need to perform 28-bit manipulation themselves. Use TRKADDR for all track address computations, even those not directly affected by this support for VSAM data sets using EAS, because this prepares you for additional support.

This macro is available only in z/OS V1R10, but the expansion will run equally well on downlevel systems if the high order 12 bits of the track number are zero (0). The TRKADDR macro supports these functions:

ABSTOREL This calculates the relative track number on the volume from the passed 'cchh' track address.

COMPARE This compares two track addresses in the new CCCCcccH format.

EXTRACTCYL This extracts the 28-bit cylinder number from the passed 'cchh' track address.

EXTRACTTRK This extracts the 4-bit track number from the passed 'cchh' track address.

NEXTTRACK This increments the track address by one track and increments the cylinder number if necessary from the passed 'cchh' track address.

NORMALIZE This reverses the 16-bit and 12-bit portions of the cylinder number from the passed 'cchh' track address. CCCCcccH becomes cccCCCCH. This could be used to subsequently perform unsigned comparisons of track addresses.

NORMTOABS This reverses the 12-bit and 16-bit portions of the cylinder from the track address. cccCCCCH becomes CCCCcccH. Use this to convert a normalized track address to an absolute 28-bit cylinder address.

RELTOABS This converts a relative track number on the volume (counting across cylinder boundaries) to a 28-bit cylinder address.

SETCYL This converts a cylinder number to a 28-bit cylinder address and sets the track portion to zero (0).

Conversions of relative track addresses to and from absolute format (MBBCCHHR) are supported automatically in the existing routines pointed to by CVTPRLTV and CVTPCNVT.

New IECTRKAD routine

Programs that are written in a high level language such as C, C++, COBOL, or PL/I can call a new routine named IECTRKAD that performs the same functions as the TRKADDR macro. This routine is available only on z/OS V1R10 but programs linked with it can run on downlevel releases.

Normalized cylinder-track address

To assist in understanding of 28-bit cylinder numbers, a normalized cylinder-track address may be used for printing. This is where the bits are rearranged to a more readable format and a linear 28-bit cylinder number. The presence of a colon (:) before the last hex digit identifies the track address as being normalized.

Note: Not all messages or reports will normalize the output. For example, IDCAMS LISTCAT reports in the native format. Others do the same. Some reports have shifted their output in order to display the larger number of cylinders. IEHLIST LISTVTOC is such an example.

3.4.2 New format DSCBs

Data set control blocks (DSCBs) are volume table of contents (VTOC) entries that describe data set attributes and allocated extent information. This extent information describes allocated space using beginning and ending track addresses. These are called *extent descriptors*. These extent descriptors may contain 28-bit cylinder number for their track addresses. DSCBs also contain metadata in the format 1 DSCB that are the characteristics or attributes of the allocated data set. There is no more space available in the format 1 DSCB to add additional attributes.

Extended attribute DSCBs

There are new DSCB types that provide a method of protecting existing programs from seeing unexpected track addresses (28-bit cylinder numbers) and new format DSCBs, as follows:

Format 8 DSCB This DSCB is equivalent to a format 1 DSCB. It contains a chain pointer to a format 9 DSCB

Format 9 DSCB This DSCB provides attribute data and a list of pointers to each possible format 3 DSCB. It contains a chain pointer to the possible next format 9 or format 3 DSCB. These attributes are maintained only for the first volume. There is only one format 9 DSCB in z/OS V1R10.

Format 4 DSCB

The format 4 DSCB contains the following new information:

- The number of cylinders on a volume.
- The existing 2-byte field DS4DSCYL in the F4 DSCB will contain the value of x'FFFE' (65 534). This identifies the volume as an EAV.
- DS4EAV is defined as a constant value of X'FFFE'.
- A new four-byte field DS4DCYL will contain the number of cylinders on the volume.
- A new allocation unit for cylinders above 65520.
- A new 2-byte field DS4LCYL will contain a code value of x'0010' to indicate that the cylinder-managed space after the first 65520 cylinders must be allocated in units that are larger than one cylinder.
- This value represents 65520 cylinders divided by 4095. For a non-EAV, this will be zero (0).
- The new field DS4MCU (minimum allocation unit) will contain the number of cylinders that each extent in the cylinder-managed area must be a multiple of. For an EAV this will be 21. For a non-EAV, this will be zero (0). It is valid only when the value in DS4DSCYL is DS4EAV.

Format 8 DSCB

This DSCB is identical to the format 1 DSCB with the following exceptions:

- The format identifier (DS1FMTID) will be x'F8' instead of x'F1'. New symbols defined:

- DS1IDC constant value of X'F1' in DS1FMTID.
- DS8IDC constant value of X'F8' in DS1FMTID.
- Track addresses in the extent descriptors starting in DS1EXT1 will use new track address format (that is, they may contain 28-bit cylinder numbers).
- The "next DSCB" address (DS1PTRDS) will always point to an F9 DSCB (a new type of DSCB) instead of a possible first F3 DSCB.

Format 9 DSCB

The format 9 DSCB will exist only for EAS-eligible data sets (VSAM, in the first release), it will contain the following:

- The format identifier will be x'F9'.
- A subtype field.
- In the first EAV release, the subtype will be 1.
- In future releases, additional subtypes may be added.
- Track addresses which point directly to up to ten F3 DSCBs.

Accessing extended attribute DSCBs

To access extended attribute DSCBs, the system requires the specification of a new permission keyword on system services that read DSCBs. By specifying EADSCB=OK, the invoking program is indicating to the system service that it understands extended attribute DSCBs and the 28-bit cylinder numbers that could be present in the data set's extent descriptors.

The EADSCB=OK keyword has been added to the following services:

- OBTAIN (CAMLST macro)
- CVAFDIR
- CVAFFILT
- CVAFDSM
- CVAFSEQ
- OPEN (DCBE macro) - opening VTOC or VSAM data set with EXCP access.

Not specifying EADSCB=OK causes these services to fail if issued to a data set that supports extended attribute DSCBs or a volume that supports cylinder-managed space (CVAFDSM and OPEN). Code this keyword when your application supports EADSCB. IBM recommends specifying it on all invocations of each service, regardless of whether the application runs on pre-z/OS V1R10 systems or accesses volumes that do not support extended attribute DSCBs. Macros on earlier releases do not recognize EADSCB but you can assemble with EADSCB=OK on z/OS V1R10 and run on downlevel releases, where EADSCB=OK will have no effect.

Specifying EADSCB=OK indicates the program understands 28-bit cylinder numbers and format 8 and format 9 DSCBs.

Format 9 DSCB vendor fields

Prior to z/OS V1R10, some vendor products used fields in the format 1 DSCB that IBM reserved for IBM use.

Format 9 DSCB is mapped by the IECSDSL1 macro, and it contains a 20-byte field DS9ATRV1 that IBM is reserving for vendors. IBM makes rules on DSCB updates but does not want to enforce content rules here. For this field, IBM is recommending the format as illustrated in Figure 3-9 on page 54.

```
Subfields that begin with the following two-byte header:
  +0       Flags.
      xxxx ....   Reserved
      .... xxxx   Number of bytes following two-byte header
     +1       Vendor code.  IBM will maintain a vendor code list
              specific for this field.
```

Figure 3-14 Format 9 DSCB vendor fields

Note: For information about VTOC usage and the VTOC DSCBs, see *z/OS V1R10 DFSMSdfp Advanced Services*, SC35-0428.

3.4.3 DEVTYPE macro

A user program can issue the DEVTYPE macro to learn whether a device supports ECKD. Use the DEVTYPE macro instruction to request information relating to the characteristics of an I/O device, and to cause this information to be placed into a specified area. The IHADVA macro maps the data returned by the DEVTYPE macro.

The information includes the number of cylinders among others. Use this macro in the following ways:

- The DEVTYPE macro issued without the INFOLIST parameter returns a 2-byte value for the number of cylinders.
- The DEVTYPE macro issued with the INFOLIST parameter (INFO=DASD) returns a different format of the device characteristics information. There is a 2-byte field is mapped by DVACYL. In addition, five new fields are returned. Use this macro call to return the following information:
  - A 4-byte value for the number of cylinders, which is mapped to field DVAICYL in the shipped mapping macro IHADVA
  - MCU value
  - First cylinder address where cylinder-managed space begins
  - Cylinder-managed space supported indicator
  - Extended attribute DSCBs supported indicator
  - Block size of index data set

Figure 3-15 on page 63 displays the 16 bytes returned when issuing the DEVTYPE macro with INFO=DASD.

```
&SDASD   DS    OCL16     INFOLIST POINTS TO LIST WITH INFO=DASD
DVAICYL  DS    FL4       NUMBER OF CYLINDERS, EXCLUDING ALTERNATES
DVAITRK  DS    FL4       TRACKS PER CYLINDER
DVAIFLAG  DS   0XL1      FLAGS
DVAIFLAG1 DS   X         FLAGS WITH PREFERRED SYMBOLS
DVAECKD1 EQU   X'80'     ECKD SUPPORTED, ALSO ON FOR VIO DATA SETS
DVALRE1  EQU   X'40'     LOCATE RECORD EXTENDED IS SUPPORTED
DVACACHE1 EQU  X'20'     THE DEVICE IS CACHED
DVAIXVLD EQU   X'10'     DVACYLMG, DVAEADSCB, DVAVIRSZ valid
DVACYLMG EQU   X'08'     Cylinder-managed space exists on
*                        this volume and begins at DVALCYL
*                        in multicylinder units of DVAMCU.
*                        DVAEADSCB is also set with this
*                        flag on. Valid when DVAIXVLD is set.
DVAEADSCB EQU  X'04'     Extended attribute DSCBs, Format 8
*                        and 9 DSCBs, are allowed on this
*                        volume.  Valid when DVAIXVLD is set.
DVAMCU   DS     XL1      Minimum allocation size in
*                        cylinders for cylinder-managed
*                        space.  Each extent in this space
*                        must be a multiple of this value.
*                        space.  Also referred to as the
*                        multicylinder unit (MCU). This is
*                        the smallest unit of disk space in
*                        cylinders that can be allocated
*                        in cylinder-managed space.
*                        Valid when DVACYLMG is set.
*                        This field is zero on releases
*                        before z/OS 1.10 or if the status
*                        is not yet known.  In these two
*                        cases DVAIXVLD is not set.
DVALCYL  DS    XL2       First cylinder address divided by
*                        4095 where space is managed in
*                        multicylinder units. Cyl-managed
*                        space begins at this address.
*                        Valid when DVACYLMG is set. This
*                        field is zero on releases before
*                        z/OS 1.10 or if the status is not
*                        yet known.  In these two cases
*                        DVAIXVLD is not set.
DVAITSET DS    XL1       TRACK SET SIZE
         DS    XL1       Reserved.  DEVTYPE currently
*                        returns zeroes but could return
*                        something different in a future
*                        release.
DVAVIRSZ DS    XL2       Block size of the index data set.
```

Figure 3-15 The DEVTYPE macro return area when INFO=DASD is specified

3.5 Extended address volume selection

The system performs volume selection for extended address volumes in the following ways:

- SMS requests will not use EAV volumes if the USEEAV setting in the IGDSMSxx parmlib member is set to NO. See “EAV installation parmlib changes” on page 98.

  Specific allocation requests are failed. For non-specific allocation requests (UNIT=SYSDA), EAV volumes are not selected. Messages indicating no space available are returned when non-EAV volumes are not available.
- For non-EAS eligible data sets, all volumes (EAV and non-EAV) are equally preferred (or they have no preference). This is the same as today, with the exception that extended address volumes are rejected when the USEEAV parmlib value is set to NO.
- During the volume selection process, each candidate volume is assessed for its capability with respect to attributes that are pertinent to volume selection. Among the key attributes is whether a volume can remain below its high threshold for free space after the requested space has been allocated. This is the ‘THRESHOLD’ requirement. Volumes that cannot meet this requirement are less preferred than volumes that do meet this requirement. For VSAM allocations, the primary and secondary thresholds are applied as described in “Allocation and migration thresholds”, and the volumes are ranked.

Allocation and migration thresholds

The allocation and migration thresholds in the system have an affect on how HSM conducts space management and SMS selects volumes. For SMS processing, two thresholds are used to determine whether a volume could be put on the list for selection:

Primary threshold This is the more important threshold; however, both the primary and secondary thresholds must be met.

When requested space >= BreakPointValue, the EAV volume meets the primary threshold if cylinder-managed space has sufficient below-threshold space.

When requested space < BreakPointValue, the EAV volume meets the primary threshold if track-managed space has sufficient below-threshold space.

A non-extended address volume meets the primary threshold if the entire volume has sufficient below-threshold space.

Secondary threshold An extended address volume or non-extended address volume meets the secondary threshold if the entire volume has sufficient below-threshold space.

For volumes with cylinder-managed space, there are two sets of values provided, as follows:

- Total volume free space thresholds, which is available in current systems
- Track-managed space and free space thresholds, which is new with z/OS V1R10

For HSM processing, a track-managed threshold was added to ensure that track-managed space gets space-managed even when the overall volume has not exceeded the threshold value. The two sets of thresholds that are considered by HSM are the entire volume space and track-managed space, as explained here:

- Specified in storage group or HSM parameters, and is recorded in DCOLLECT storage group fields (Record Type ‘SG’), reported by ISMF.
- The track-managed threshold was added to ensure that track-managed space gets space-managed even when overall volume has not exceeded the threshold value.

3.5.1 EAV space considerations

Give additional consideration to how space is allocated for a device that supports cylinder-managed space. Space is rounded up to the next MCU if the entire extents are allocated in cylinder-managed space.

Individual extents always start and end on an MCU boundary in cylinder-managed space. This applies to the creation of data sets, extending them with more space and also releasing unused space (partial release). For partial release, some or all of the unused space may not be released because extents must end on a MCU boundary.

VSAM striping

For VSAM striping, all stripes must have a common “release point”, which is a common RBA based on MCU or CA boundaries. A given extent is contained in a single managed space. This means that an extent cannot straddle where cylinder-managed space begins. The exact space is obtained if some portion of the space is allocated in track-managed space.

Current algorithms in the search for space continue to apply. The system selects the first free space extent from the start of the preferred managed space that can satisfy the requested quantity. When the space cannot be satisfied in one extent, additional extents may be used to satisfy the request where the used extents are ranked from the largest to smallest in the preferred managed space.

When the request cannot be satisfied from the preferred managed space (as determined by the BreakPointValue), or the exact space is not available for a VSAM striped data set, then the available free space extents from the entire volume are ranked and used. Space could be from the non-preferred area or from both track-managed and cylinder-managed spaces.

Considerations for VSAM striping

When the system extends an SMS VSAM data set and the secondary space must be returned in multiple extents due to volume fragmentation, prior to z/OS V1R10 the system extends the last current extent only if the first new extent happens to be contiguous. In z/OS V1R10, the system extends the current last extent if any of the new extents is contiguous with it.

Because VSAM striped data sets require all stripes to be the same size, the system will attempt to honor the exact requested space by using a combination of cylinder-managed and track-managed spaces, as follows:

- It allows stripes in a mixed environment.
- If the exact requested space cannot be obtained for any stripe, the system will round up the requested space to next MCU for all stripes.

> **Note:** An exact preferred request may return two extents (where it used to be one).

To help consolidate extents, the current last extent will be enlarged if any of the newly acquired extents are contiguous to it.

VSAM extended format

For VSAM extended format, the system looks at all newly acquired extents for an extent contiguous to the current last extent. Current code only looks at the first returned extent from the extend of space.

Overall, the following rules apply:

- Exact space will be obtained if allocated in track-managed space.
- A given extent is contained in a single managed space.
- Extents cannot straddle where cylinder-managed space begins.

3.5.2 Allocating data sets

The introduction and use by the system of an EAV is determined by adding the volumes to SMS storage groups and/or adding them to non-SMS managed storage pools (specific or esoteric volumes). Volumes could be in their dedicated storage group or pool, or mixed with other volumes. Adding and enabling them to these pools allows the system to allocate VSAM data sets in cylinder-managed space if the USEEAV parmlib setting is set to YES. When USEEAV is set to NO, no new data set creates (even non-VSAM) are allowed on any EAV in the system.

As described earlier, VSAM and non-VSAM data set types may reside in the track-managed space. VSAM data sets, however, can be allocated anywhere on the volume. The preference on how the system determines which managed space should be used is based on the derived BreakPointValue for the target volume of the allocation.

EAV volumes can be SMS-managed or non-SMS managed, as follows:

- SMS storage groups and non-SMS managed volume pools may be defined with all EAV, all non-EAV or a mix of EAV and non-EAV volumes.
- Including or excluding EAVs in particular storage groups or pools controls whether VSAM data sets could reside in cylinder-managed space.

Selecting a volume

For EAS-eligible data set allocation, the volume selected is based on the following criteria:

- Specific, non-specific requests have no volume preference.
- SMS ranks volumes with all other volume preference attributes.
  - Volumes that meet primary and secondary thresholds are ranked higher.
  - EAV volumes are preferred over non-EAV volumes where space >= BreakPointValue (there is no EAV volume preference where space < BreakPointValue).
  - Striping allocations, SMS prefers EAV volumes that can provide the exact requested space for each stripe (when stripe size >= BreakPointValue and MCU).

SMS selects volumes with the higher ranking or weighting.

VSAM control area

All VSAM data sets created in z/OS V1R10 may have different CA sizes from what would have been received on prior releases. The reason for this change is that a CA must be compatible with the multicylinder unit (MCU) for cylinder-managed space. Each extent allocated must be a multiple of the CA size. The CA size must be compatible with the MCU for cylinder-managed space, as follows:

- These compatible CA sizes are 1, 3, 5, 7, 9 and 15 tracks.
- The MCU (315 tracks) is a multiple of these CA sizes.
- This allows the VSAM data set to be EAS-eligible.

The list of CA sizes allows any VSAM data set to be EAS-eligible that otherwise would not have been. All these are divisors of 315 tracks (21 cylinders).

VSAM data sets allocated with compatible CAs on a non-EAV are eligible to be extended to an additional volume that may be one which supports cylinder-managed space.

Note: VSAM data sets physically copied from a non-EAV to an EAV may have an incompatible CA and thus would not be EAS-eligible. This means extends for additional space would not use cylinder-managed space.

The system may adjust the primary and secondary quantity to select a CA. The system may adjust the CA size and the primary and secondary space amounts, as follows:

- TRK(24,4) will have a CA of 5 tracks, with a primary and secondary amount of 25 and 5 tracks.

This is a consideration when migrating to z/OS V1R10.

3.6 Managing extended address volumes

With this new, additional support for extended address volumes, there are functions within DFSMSdss and DFSMShsm that have new considerations regarding how to use the functions with the new and larger volumes.

3.6.1 DEFRAG and CONSOLIDATE performance improvement

DEFRAG Version 2 is new with z/OS V1R10. The DEFRAG function consolidates the free space on a volume to prevent out-of-space conditions on new allocations. DEFRAG accomplishes this by relocating data set extents on a DASD volume to reduce or eliminate free space fragmentation. It also prints a report about free space and other volume statistics.

The CONSOLIDATE function consolidates data set extents on a volume to improve performance when reading data sets. CONSOLIDATE accomplishes this by combining contiguous extents of a data set into one extent, and by relocating multiple non-contiguous data set extents into contiguous space. It prints a report of volume statistics.

As volumes have increased in size, the time that it takes to process them with DEFRAG has also increased. The larger volume size inherently requires more time to process the increased number and size of data sets. This increased processing time affects most customers because they have limited time in which to run maintenance tasks on their systems. CONSOLIDATE, being a function that is extracted from DEFRAG, shares the same performance problems.

DEFRAG changes

The changes introduced in z/OS V1R10 support will begin to address this issue. Performance improvements are made in these functions by replacing the method of maintaining the locations of data set extents and free space with a more efficient method.

With the new DEFRAG, Version 2, DFSMSdss obsoletes the CONSOLIDATE keyword. If CONSOLIDATE is specified, the user receives an informational messages indicating that the keyword is no longer supported, as follows:

```
ADR109I (R/I)-RI01 (01), 2008.241 12:57:48 INITIAL SCAN OF USER CONTROL
STATEMENTS COMPLETED
```

```
ADR145I (R/I)-RI03 (01), OBSOLETE KEYWORD CONSOLIDATE SPECIFIED. THE
CONSOLIDATE COMMAND WILL BE RUN PRIOR TO DEFRAG
```

DEFRAG will then run the new CONSOLIDATE command against all eligible data sets after EXCLUDE and/or BY filtering. After CONSOLID processing completes, DEFRAG, Version 2, will run normally. The new **CONSOLID** command will try to reduce the number of extents of a data set as much as possible, even when the entire data set cannot be reduced to one extent. Using CONSOLID also enables you to selectively specify the data sets that you might want CONSOLID to focus on.

Note: You can run DEFRAG and CONSOLIDATE functions on a volume at any time. However, these operations lock the VTOC (through the RESERVE macro) and the VVDS, if it exists on the volume. They also serialize on data sets through ENQ or dynamic allocation. These activities might cause excessive wait time for other jobs to update the VTOC. Therefore, times of low system activity are best for DEFRAG and CONSOLIDATE runs.

DFSMSdss can use FlashCopy during a DEFRAG or CONSOLIDATE operation if the device is in an ESS that supports data set FlashCopy. FlashCopy is much faster than traditional data movement methods, especially when you are moving large amounts of data.

New performance keywords

Four new keywords have been added to provide better performance with DEFRAG:

MMOVPCT(n,p) This keyword stops the DEFRAG run when n% contiguous tracks on the volume are assembled as free. If n% contiguous tracks already exist as free tracks, the DEFRAG function tries to further reduce the fragmentation of the volume, but no more than n% tracks are relocated. If more than n% tracks must be relocated, no DEFRAG is performed.

- **n** This is the percentage of tracks on the volume that DFSMSdss is to try to assemble as free tracks in a contiguous area.
- **p** This is the number of passes DFSMSdss is to make in attempting to assemble the tracks.

Note: Using the MMOVPCT keyword is recommended instead of using MAXMOVE when running DEFRAG on an EAV. MMOVPCT applies separately to the track-managed space and the cylinder-managed space. MAXMOVE and MMOVPCT are mutually exclusive.

MAXTIME(nummins) This keyword specifies the maximum number of minutes that the DEFRAG function should be allowed to process. This allows the user to control the amount of the time that the job will run. MAXTIME will be checked after processing each data set. If the MAXTIME has passed, the DEFRAG function will end.

nummins is a decimal number (0-9999) that specifies the maximum number of minutes the DEFRAG function will run. A value of 0 is ignored.

PASSDelay This keyword specifies the time delay between the passes (p) specified in MAXMOVE(n,p) or MMOVPCT(n,p) to allow access to the volume between passes.

VERSION1 This keyword specifies which version of DEFRAG is executed. During DEFRAG operations, you might want to execute the pre-z/OS

V1R10 version of DFSMSdss DEFRAG. VERSION1 only supports volumes that are 65,520 cylinders or less. Any new function added to DEFRAG with z/OS V1R10 or later will not be available.

Note: The elapsed time of the operation may be slightly longer than the MAXTIME specified because the MAXTIME is checked after each data set has been processed. If the CONSOLIDATE keyword is specified, MAXTIME will be ignored.

DEFRAG Version 2

This new DEFRAG Version 2 provides the following enhancements:

- Performance for extended address volumes and non-extended address volumes.
- The CONSOLIDATE keyword on DEFRAG results in calling a new **CONSOLIDATE** command first. The command supports the same type of data set filtering as COPY, DUMP, and RESTORE functions.

  The new command performs both extent consolidation and reduction. It also allows you to specify which data sets are to be processed. This command tries to reduce the number of extents of a data set as much as possible even when the entire data set cannot be reduced to one extent.

  The **CONSOLIDATE** command can be restricted by a new RACF FACILITY-class profile STGADMIN.ADR.CONSOLID. It allows a CONSOLIDATE operation without having READ access to the data sets that are moved.
- An informational message is returned.
- IBM plans to remove the CONSOLIDATE keyword from the DEFRAG function in a future release

Note: Using the MMOVPCT keyword is recommended over using MAXMOVE when running DEFRAG on an extended address volume. To have the DEFRAG function perform in the shortest period of time, or to create the largest single freespace extent, perform only the first pass. Do so by coding the MAXMOVE(n) parameter using a very high value for n, or code the MMOVPCT(n) parameter using a high percentage amount.

When the value is higher than what the DEFRAG function can assemble, the process stops at the end of the first pass; for example:

```
DEFRAG DYNAM(388002) MAXMOVE(9999) - or
DEFRAG DYNAM(388002) MMOVPCT(75)
```

3.6.2 DFSMShsm space management

Space management will be performed on both the total volume (existing) and track-managed (new) space. When the track-managed space threshold is exceeded and the total volume threshold is not, only data sets with one or more of its first three extents in track-managed space will be processed. The purpose of this support is to ensure that track-managed space gets space-managed even when the overall volume has not exceeded its volume threshold.

DFSMShsm performs a space check on each DFSMShsm volume that is being managed. Interval migration of SMS-managed EAVs having the AUTOMIGRATE=I attribute in Storage Groups will be performed if either the midpoint between the high and low volume threshold values is met or if the midpoint between the high and low track-managed threshold values is reached or exceeded.

The track-managed threshold is set via the following:

- SMS: Storage Group
- Non-SMS: **ADDVOL** command

DFSMShsm examines both volume level thresholds and track-managed thresholds for EAV L0 volumes. If either threshold has been exceeded, then migration eligibility will be performed for data on the volume. If track-managed threshold values are not specified, the default is to use the volume-level threshold values.

Data set eligibility will be further qualified by the location of the first three extents of the data set. If any of the first three extents are allocated in track-managed space, the data set will be eligible to be processed when the track-managed space or entire volume space is managed. If none of the first three extents are allocated in track-managed space, then the data set will be eligible to be processed when the entire volume space is being managed.

Table 3-1 lists and explains the DFSMShsm space management thresholds.

Table 3-1 DFSMShsm space management thresholds

Track-managed threshold exceeded	Volume threshold exceeded	Data set selection
Yes	Yes	All data sets.
No	Yes	All data sets.
Yes	No	Only data sets with one or more of the first three extents in track-managed space.
No	No	Data sets are not selected.

Migrate data sets

After putting the candidate data sets in priority order, DFSMShsm compares the occupied space on the volume to the low threshold for the volume. If the occupied space is more than the low threshold, DFSMShsm migrates the highest priority data set.

Data set size is a determining factor for the order in which data sets are migrated. Managing an entire extended address volume (EAV) with one threshold is not sufficient because average data set size in the cylinder-managed space will likely exceed the average data set size in track-managed space. This will result in data sets allocated in the cylinder-managed space being migrated at a higher rate than data sets allocated in the track-managed space, potentially causing no space in the track-managed space being freed.

To prevent this, a unique threshold for the track-managed space (track-managed threshold) is used to manage the free space separate from the free space on the entire volume (volume-level threshold).

ADDVOL command

The DFSMShsm **ADDVOL** command has an optional keyword TRACKMANAGEDTHRESHOLD(high low). TRACKMANAGEDTHRESHOLD (TMT) can be used to define the high and low thresholds for the track-managed space of a non-SMS EAV PRIMARY volume.

If TMT is not specified, then the corresponding values for THRESHOLD will be used. If no threshold values are specified, then all threshold values will use the default values. If TMT is specified for a non-extended address volume, the **ADDVOL** command will fail; for example:

```
ADDVOL PRIM01 PRIMARY UNIT(3390) TRACKMANAGEDTHRESHOLD(80 40) THRESHOLD(85 60)
```

In this example, the parameter TRACKMANAGEDTHRESHOLD(80 40) defines a High Track-Managed Threshold of 80% and a Low Track-Managed Threshold of 40%. If TMT is not specified, then the corresponding values for THRESHOLD will be used.

If no THRESHOLD keyword values are specified, then all threshold values will use the default values. The default value for High is 100, and the default value for Low is 0. Therefore, a primary volume has no valid default thresholds in this case.

If the attributes of a volume are being changed but track-managed space threshold values are not specified, the track-managed space thresholds are not changed. If, however, the new high track managed threshold is less than or equal to the low track managed threshold, or is equal to 100, the volume threshold values will be used as track managed threshold values.

Small Data Set Packing data sets and extended address volumes

Because Small Data Set Packing (SDSP) data sets can be allocated in cylinder-managed space, extended address volumes can be used as ML1 volumes. SDSPs are eligible for EAV because they are VSAM data sets. Any DFSMShsm-managed data set can be an EAV, but only ML1 volumes with an SDSP data set are able to exploit the cylinder-managed space.

A recommended usage is to define an EAV volume such that the cylinder-managed space is just large enough to allocate an SDSP data set. That enables all of the track-managed space to be used for non-SDSP migration data sets.

For ML1 volumes, migration copies can only be written to track-managed space (migration copies are non-VSAM data sets). The threshold for ML1 volumes only applies to the track-managed space.

3.6.3 VTOC index data set

The VTOC contains several kinds of DSCBs. The first record in every VTOC is the VTOC DSCB (format 4). The record describes the device the volume resides on, the volume attributes, and the size and contents of the VTOC data set.

The next DSCB in the VTOC data set is a free-space DSCB (format 5), even if the free space is described by format 7 DSCBs. The third and subsequent DSCBs in the VTOC can occur in any order.

One or more DSCBs in the VTOC define a data set on each volume on which the data set resides. The number of DSCBs needed to define a data set is determined by the number of extents that the data set occupies and by whether it has a format 9 DSCB. One or more format 3 DSCBs are required for data sets with more than three extents.

When a data set has several extents and is an EAS-eligible data set, it requires a format 8 and a format 9 DSCB, plus an additional format 3 DSCB.

Figure 3-16 on page 72 illustrates an example of the relationship of a VTOC to its index.

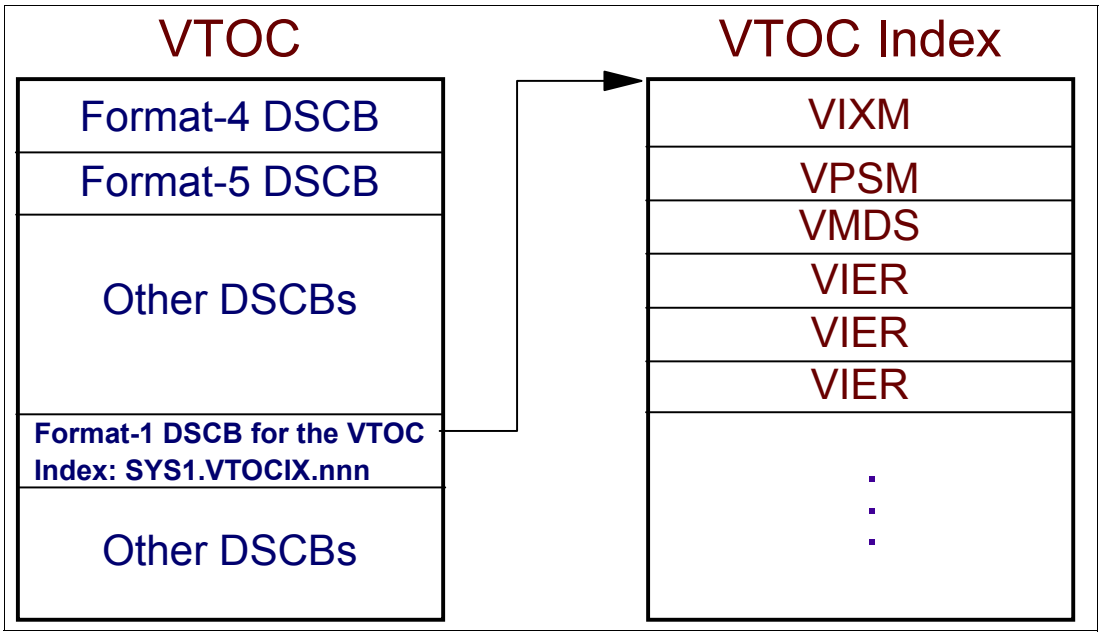


Figure 3-16 Example of the relationship of a VTOC to its index

VTOC index

The VTOC index enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC. It consists of an index of data set names in format 1 DSCBs contained in the VTOC and volume free space information.

With z/OS V1R10, the index block size is increased from 2048 bytes to 8192 bytes for devices with cylinder-managed space. The new block size is recorded in the format 1 DSCB for the index. It is needed to allow for scaling to the largest-sized volumes. The DEVTYPE INFO=DASD macro can be used to return the actual block size, or it can be determined from examining the format 1 DSCB of the index data set.

To refresh a volume VTOC and INDEX in its current format, use the ICKDSF **`REFORMAT`** command with the RVTOC keyword. To optionally extend the VTOC and INDEX, use the ICKDSF **`REFORMAT`** command with the EXTVTOC and EXTINDEX keywords.

Note: Device Support Facilities (ICKDSF) initializes a VTOC index into 2048-byte physical blocks, or 8192-byte physical blocks on an extended address volume, named VTOC index records (VIRs). The DEVTYPE INFO=DASD macro can be used to return the actual block size, or it can be determined from examining the format 1 DSCB of the index data set

VTOC Index Records

VTOC Index Records (VIRs) equal VPSMs plus VMDSs plus VIXMs. VIRs are limited to 255 records. Volumes with cylinder-managed space will have an index record size of 8 K.

A VTOC index contains the following kinds of VIRs, as shown in Figure 3-16 on page 72:

- The VTOC index entry record (VIER) identifies the location of format 1 and format 8 DSCBs and the format 4 DSCB.
- The VTOC pack space map (VPSM) identifies the free and allocated space on a volume.
- The VTOC index map (VIXM) identifies the VIRs that have been allocated in the VTOC index.

- The VTOC map of DSCBs (VMDS) identifies the DSCBs that have been allocated in the VTOC.

A format 1 DSCB in the VTOC contains the name and extent information of the VTOC index. The name of the index must be 'SYS1.VTOCIX.volser', where 'volser' is the serial number of the volume containing the VTOC and its index. The name must be unique within the system to avoid ENQ contention, and it must conform to standard data set naming conventions.

VTOC index map

The VTOC index map (VIXM) has a new bit, VIMXHDRV, to indicate that new fields exist in the new VIXM extension. The VIXM contains a new field for the RBA of the new large unit map and new space statistics.

The VIXM also contains a new field for the 'minimum allocation unit' in cylinders for the cylinder-managed space. Each extent in the cylinder-managed space must be a multiple of this on an EAV. There is an extended header in VIXM for free space statistics.

Note: If the VTOC index size is omitted when formatting a volume with ICKDSF and does not preallocate the index, the default prior to this release was 15 tracks. With z/OS V1R10 and EAV support, the default size for EAV and non-EAV volumes is calculated and may be different from earlier releases.

VTOC pack space map

The VTOC pack space map (VPSM) shows the allocated space on a volume and the space that remains free. The space within the first 65,520 cylinders is called the track-managed space. A 2 K record size can map 999 cylinders in one VPSM. An 8 K record can map 4,071 cylinders in one VPSM. A VPSM that maps only cylinders can map up to 1,368,864 cylinders.

The VPSMs for this area contains bit maps of the cylinders and tracks on the volume. A value of 1 indicates that the cylinder or track has been allocated. A value of zero (0) indicates that the cylinder or track is unallocated. The space after the first 65,520 cylinders is called the cylinder-managed space. The VPSMs for this area contains bit maps of only multicylinder units. Individual cylinders and tracks are not mapped in this type of VPSM. A value of 1 indicates that the multicylinder unit is allocated. A value of zero (0) indicates that the multicylinder unit is unallocated.

For the MCUs:

- In the new map, each bit will represent 21 cylinders instead of each bit representing 1 cylinder.
- A new field will define the number of cylinders per large unit map.
- The system rounds the size of each extent that happens to be satisfied in the cylinder-managed space, up to a multiple of 21 cylinders. Programs must allow for the possibility that any extent might be larger than requested.
- Contents of index map records increased proportionally (more bits per record, offsets changed)
- Track-managed space. Small (tracks and cylinders) and large (cylinders) unit map.
- Cylinder-managed space. Each bit in large unit map represents 21 cylinders (an MCU).
- Programs that access index maps must use existing self-describing fields in the map records.

3.7 User program accesses to the VTOC

You can use either DADSM or common VTOC access facility (CVAF) macros to access a VTOC and its index. The DADSM macros and tasks covered here include the following:

- OBTAIN reads 1 or more DSCBs from the VTOC.
- The CVAFDIR macro can be used to read or write 1 or more DSCBs.
- LSPACE provides information on volume size, free space on the volume, free space on the VTOC and INDEX, volume fragmentation, and VTOC status. Also provided is information about the size of the track-managed space and its free space statistics.

OBTAIN and CAMLST have a new option to specify the number of DSCBs you want to read. CVAFDIR provides a new keyword MULTIPLEDSCB=YES to indicate to CVAFDIR processing to use the multiple buffers passed in the buffer list.

OBTAIN search processing will store an additional 2 bytes in the caller's return area right after the 101 bytes that is set by OBTAIN on prior releases. Ensure that your program provides the minimum 140-byte return area. These 2 bytes are set to the total number of consecutive 140-byte areas that are needed to read all the DSCBs for the data set.

3.7.1 OBTAIN macro

To read 1 or more DSCBs into virtual storage, use the OBTAIN and CAMLST macro instructions. Identify the DSCB to be read using the name of the data set associated with the DSCB, or the absolute track address of the DSCB. Provide a 140-byte data area in virtual storage to contain the DSCB. On a request to read multiple DSCBs, specify the NUMBERDSCB= number_dscbs on the OBTAIN or CAMLST macro and provide consecutive 140-byte return areas in virtual storage to contain this number of DSCBs.

When you specify the name of the data set, an identifier (format 1, format 4, or format 8) DSCB is read into virtual storage. To read a DSCB other than a format 1, format 4, or format 8 DSCB, specify an absolute track address.

EADSCB=OK parameter

Code the EADSCB=OK on the OBTAIN or CAMLST macro when your program supports DSCBs that describe data sets with format 8 and format 9 DSCBs. The extent descriptors in DSCBs for a data set described with these formats may have 28-bit cylinder track addresses. Use the TRKADDR macro or IECTRKAD service to manipulate 16-bit or 28-bit cylinder track addresses.

For SEARCH and SEEK requests, number_dscbs is an absolute expression with a value between 0 and 255 that designates the number of consecutive 140-byte return areas that are provided in wkarea_relexp. The system treats a value of 0 as a 1. Currently the system does not support a chain of more than 12 DSCBs for one data set, but it is valid for you to provide an area that is longer than currently needed. The system verifies that the provided area is valid.

When you provide an area that is long enough to contain more than one DSCB, obtain processing will return DSCBs for the requested data set name in logical VTOC order until all 140-byte return areas are used. The logical VTOC order is a format 1 DSCB, followed by zero (0) or more format 3 DSCBs or a format 8 DSCB, followed by one or more format 9 DSCBs, followed by zero (0) or more format 3 DSCBs. No absolute maximum number of DSCBs for a data set should be assumed.

Note: For programs run on an older level of the system that does not support this keyword, the NUMBERDSCB value will be treated as if it were 1.

3.7.2 CVAFDIR processing

Using CVAFDIR, a new specification MULTIPLEDSCBS=YES|NO is allowed. This specification indicates that the calling program requests to read or write multiple DSCBs to or from a buffer list that contains more than one buffer list entry. This flag will resolve to a new indicator in the CVPL, CV4MULTD, to be set on.

Multiple DSCB processing for reads and writes is requested by specifying the MULTIPLEDSCBS=YES keyword and providing a buffer list that contains more than one buffer list entry (BFLHNOE>1). MULTIPLEDSCBS=NO is a specification indicating that the calling program requests that only 1 DSCB should be processed. This is the default for MF=L and MF=I forms of the CVAFDIR macro.

When the MULTIPLEDSCBS keyword is not specified on the MF=E form, the existing setting of CV4MULTD is left unchanged. When MULTIPLEDSCBS=NO is specified or defaulted, only the first available buffer list entry is processed.

VTOC order

Only the first buffer list entry, seek or search, argument will be used. This will provide orientation to the data set from which the subsequent data set DSCBs will be read. As each one is read, the DSCB argument (BFLEARG) in each buffer list entry will be set in the format specified by the caller in each buffer list entry. The buffer list argument will be indicated as updated with the flag, BFLEAUPD, set on. Reading the data set DSCBs will be in the logical VTOC order, and will continue as long as buffer list entries are available to return the DSCB.

A new field in the buffer list header (BFLHNOEN) will be set by CVAFDIR read processing to indicate the number of buffer list entries that are needed to read the entire set of DSCBs for the data set. This number is set in the header of the first buffer list. Its setting is independent of the specification of the MULTIPLEDSCBS= keyword, the target volume type, and whether the number of provided buffer list entries, BFLHNOE, is short.

The logical VTOC order would be either:

- Format 1 -> Format 3s
- Format 8, -> Format 9s -> Format 3s

Buffer list entries

Buffer list entries other than the first must provide a 140-byte buffer. The first buffer list entry buffer size will follow the same rules as today. That is, with the seek option, provide either a 96-byte or 140-byte buffer. For the search option, provide a 96-byte buffer. All other buffer list entry processing flags as described in the buffer list entry flag byte (BFLEFL) should continue to be supported. They include data in buffer modified, skip, I/O error, no key verify and argument format qualifiers. The DSCB argument (BFLEARG) returned in each buffer list entry is in the format determined by the argument format qualifiers (BFLECHR or BFLETTR) in each buffer list entry.

The buffer list header must indicate the number of buffer list entries passed. Only buffer list entries without the skip flag on will be processed. The order in which the DSCBs are passed in the buffer list entries must correspond to the logical VTOC order as described previously.

A new flag in the buffer list header can be set by the caller to indicate that the logical order in which the DSCBs appear in the buffer list must be written in reverse order (BFLHWREV).

For write processing the first buffer list entry, like today, can be a 96-byte buffer if the DSCB to write is a format 1. The same will hold true for a format 8 DSCB. A 140-byte buffer can also be provided for these DSCBs as long as the BFLEARG points to the actual DSCB that needs to be written.

Buffer list entries that do not describe format 1 or format 8 DSCBs must provide a 140-byte buffer, and its buffer address (BFLEARG) must point to the actual DSCB that needs to be written. The caller must also specify the buffer list entry argument (BFLEARG) as a CCHHR for these buffer list entries. A new flag in the buffer list entry could be set for entries where a format 0 DSCB verify before a write is not needed (BFLENVER). This will override the VERIFY=YES setting.

3.7.3 LSPACE macro

You can use the LSPACE macro to get free space, volume fragmentation, and volume table of contents (VTOC) status information for a direct access storage device (DASD) volume. The LSPACE macro returns status information (such as LSPACE subfunction, return code, and reason code) in the parameter list. The LSPACE macro also returns the return code in register 15.

For volumes that are configured with more than 9999 cylinders, you can use the EXPMSG option to create an expanded message return area that the LSPACE macro needs. For volumes that are configured with more than 65 520 cylinders, you can use the XEXPMSG option to create an extended expanded message return area that the LSPACE macro needs.

The expanded data return area (EXPDATA) will return binary data of free space information for volumes with more than 65520 cylinders. You can have LSPACE return additional information such as the format 4 DSCB, the total number of free extents on the volume, or the fragmentation index.

The new LSPACE keywords are listed and explained here:

XEXPMSG XEXPMSG=addr or (reg) or 0 specifies the address of a caller-provided 95-byte extended expanded message return area into which LSPACE returns either a free space message or, for unsuccessful requests, status information. Specify this keyword if you want to obtain free space information in the message return area for volumes that are configured with more than 65520 cylinders. The returned free space will include space for the total volume and space from the track-managed space on a volume. For volumes with 65520 or fewer cylinders, both sets of free space information will be returned but they will be the same.

EXPDATA EXPDATA=addr or (reg) or 0 specifies the address of a caller-provided expanded data return area into which LSPACE returns expanded free space and volume information. Specify this keyword if you want to obtain free space information in the LSPACE data return area for volumes that are configured with more than 65520 cylinders. The returned free space will include space for the total volume and space from the track-managed space on a volume. For volumes with 65520 or fewer cylinders, both sets of free space information will be returned but they will be the same.

DATATYPE DATATYPE= (VOLUME,VTOC,INDEX,FRAGINDEX,ALL) is only allowed when the DATA or EXPDATA keyword is specified. Only the information specified will be returned to the caller. DATATYPE is valid for both non-EAV

and EAV. This keyword will eliminate unnecessary I/O required to retrieve free space information that is not be required by the caller.

- VOLUME– Provide free space information for the VOLUME
- VTOC – Provide free space information related to the VTOC
- INDEX – Provide free space information related to the VTOC INDEX
- FRAGINDEX – Provide the fragmentation index
- ALL – Provide all available LSPACE statistics (default)

PLISTVER PLISTVER=plistver | IMPLIED_VERSION | MAX defines the version of the LSPACE parameter list that should be generated for the MF=L form of the LSPACE macro. The value for plistver is a byte input decimal value in the "1-2" range that specifies the version of the LSPACE parameter list that should be generated. The macro keys associated with each supported version of the macro are listed here. This PLISTVER= keyword is required for any macro keys associated with version 2 or larger to be specified:

- VERSIONKEY
- 1MSG
- EXPMSG
- DATA
- SMF
- F4DSCB
- 2XEXPMSG
- EXPDATA
- DATATYPE

When MAX is specified, the generated parameter list is the largest size currently supported. This size may grow from release to release, thus possibly affecting the amount of storage needed by your program. If your program can tolerate this, IBM recommends that you always specify MAX when creating the list form parameter list because that will ensure that the list form parameter list is always long enough to hold whatever parameters might be specified on the execute form.

When IMPLIED_VERSION is specified, the generated parameter list is the lowest version, which allows all of the parameters on the invocation to be processed.

When PLISTVER is omitted, the default is the lowest version of the parameter list, which is version 1.

3.8 SMF records and user programs

Programs that build channel programs for VSAM data sets must be changed if the data set has cylinders with numbers greater than 65535. This is because it is unlikely that such a program would work without a change to support 28-bit cylinder numbers for the track address. In a later z/OS release, IBM expects to support non-VSAM data sets that will find this problem.

It is possible to control an I/O device directly while processing a data set with almost any data organization without using a specific access method. The EXCP (execute channel program), EXCPVR, and XDAP macros use the system function that provides for scheduling and queuing I/O requests, efficient use of channels and devices, data protection, interruption procedures, error recognition, and retry.

Two kinds of program issue an OPEN macro and then issue EXCP, EXCPVR, or XDAP. Both types of programs can be found by reading source code and by examining SMF records. The types are:

- The DCB has MACRF=E, which signifies that the DCB is for EXCP. This signifies that the DCB is only for EXCP and bit 0 (DCBMRECP) in DCBMACRF is on.
- The DCB has a MACRF value other than E, which signifies that the DCB is for a regular access method.

3.8.1 SMF type 14 and type 15 records

To facilitate migration of programs that build channel programs, the SMF type 14 and SMF type 15 records have a new bit that indicates whether one or more instances of EXCP, EXCPVR, or XDAP were issued for a non-EXCP DCB. The XDAP macro results in issuing EXCP so they are indistinguishable.

Attention: For programs that do not issue and OPEN, but instead issue EXCP, EXCPVR, or XDAP, these macros require a DCB and a DEB. If the application program builds its own DEB, it can do that only if it is authorized.

The operating system cannot detect these types of instances when they are directed to an extent that has 28-bit cylinder numbers. In other words, the system cannot tell whether the program has been upgraded to handle 28-bit cylinder numbers. It is necessary for a programmer to search source code and run tests to find these programs.

Finding user programs

To find instances of these EXCP DCBs, the programmer can search source code and also can examine SMF type 14 and type 15 records. One of these records is written when the user makes the transition to another volume or issues a CLOSE macro for a non-VSAM data set. The 2-byte SMFDCBMF field contains a copy of the DCBMACRF field. If bit 0 is on, it means that the DCB is only for EXCP (MACRF=E). The JFCB will contain the data set name unless the OPEN is for a partitioned concatenation.

If the program issues an OPEN for an EXCP DCB, OPEN can determine whether the program as been upgraded to handle EAV because the program must supply a DCBE with the EADSCB=OK keyword coded. In z/OS V1R10, if an attempt is made to open an EAS-eligible data set (VSAM) where a format 8 DSCB exists, or open the VTOC and this EADSCB=OK keyword is not coded, then a new ABEND and message IEC142I 113-44 is issued. In a future release where other data set types will be EAS-eligible, this condition will occur for them also.

SMF type 14 record changes

- A new flag, SMF14EADSCB, indicates whether a program specified EADSCB=OK on the DCBE macro. There is a migration aid in z/OS V1R10 to identify programs that open the VTOC or VSAM data sets with EXCP (MACRF=E).

  In z/OS V1R10 or later, this data set may have a format 8 DSCB. IBM recommends that you upgrade the program to handle 28-bit cylinder numbers and code EADSCB=OK.

  When SMF14EADSCB is off, it identifies programs that may need to be upgraded to handle 28-bit cylinder numbers and have EADSCB=OK coded in preparation for future release.

If SMF14EADSCB in the SMF type 14 or 15 record is zero (0), then the program did not specify EADSCB=OK on the DCBE macro. If you expect to use this program with a data set that has a format 8 DSCB in z/OS V1R10 or later, then IBM recommends that you upgrade the program to handle 28-bit cylinders and code EADSCB=OK.

- A new flag, SMF14EXCPBAM, indicates that a program has used a non-EXCP OPEN DCB (BSAM, QSAM, BPAM) and has issued EXCP or XDAP.

  If SMF14EXCPBAM is not set, then EXCP or XDAP was not issued while it was opened for BAM processing.

  No changes are required in this case because BAM internally will handle the 28-bit cylinder numbers (in a future release). If it is set, then an EXCP or XDAP was specified while it was opened for BAM processing and this program needs to be upgraded to handle 28-bit cylinder numbers. In a future release, this instance will be failed if a data set has a format 8 DSCB and EADSCB=OK was not specified on the DCBE macro.

These two new flags indicate that the EADSCB=OK keyword on the DCBE macro may need to be specified only if at some point this data set will be EAS-eligible. In z/OS V1R10, EAS-eligible data sets are VSAM data sets. The VTOC is not EAS-eligible but is a data set that may contain format 8 and format 9 DSCBs. For data sets that are currently not EAS-eligible, the specification or absence of the EADSCB=OK keyword will have no effect.

Note: Upgraded programs will continue to operate correctly on downlevel systems. The VTOC cannot be in cylinder-managed space but opening one on a volume that supports extended attribute DSCBs requires EADSCB=OK because the VTOC may contain format 8 and format 9 DSCBs.

Other SMF enhancements

The following SMF records have also been enhanced:

- SMF record type 19 (volume statistics)
  - LSPACE statistics recorded are expanded with track-managed free space statistics and total volume and track-managed space sizes.
- SMF Type 60, 61, 64, 65, 66 (various VSAM events)
  - Each of these SMF records contains a copy of a catalog record for a VSAM data set. They may contain extent descriptors with cylinder numbers greater than 65535.
- SMF Type 74, subtype 1 (RMF device activity)
  - New device capacity field saved.

Non-EXCP OPEN and Issue EXCP or XDAP

The DCB is for an access method other than EXCP. In this case, bit 0 of DCBMACRF and in SMFDCBMF are not 1. z/OS DFSMSdfp Advanced Services states that it is valid to issue EXCP but not EXCPVR in this case. An EXCPVR would result in abend 400.

Programmers should search source code for these instances of EXCP and XDAP. In addition, the following data in SMF will be recorded:

- The new SMF14/15 flag SMF14EXCPBAM is defined to help find programs that issue a Non-EXCP OPEN and issue EXCP or XDAP. This flag will be set on when the access method of BSAM, QSAM, or BPAM is used and the user program issues one or more instances of the EXCP or XDAP macro since the DCB was opened.
- If the flag SMF14EADSCB in this SMF record is zero (0), then the program did not specify EADSCB=OK on the DCBE macro. Upgrade the program to handle 28-bit cylinders and code EADSCB=OK.

Changing programs that issue EXCP, EXCPVR, or XDAP

The simplest way to handle 28-bit cylinder numbers is to use the TRKADDR macro. It will work equally well with track addresses that contain 28-bit and 16-bit cylinder numbers, and with any DASD data set. It also will work correctly on systems before EAV is supported in z/OS V1R10. This is planning work for a future release where non-VSAM data sets will be supported in EAS.

In a future release, issuances of EXCP or XDAP will be failed for a non-VSAM data set under the following conditions:

- When the access method is BSAM, BPAM, or QSAM
- When the data set has a format 8 DSCB

Use the techniques described in the preceding sections to find and upgrade these programs and add the EADSCB=OK option to the DCBE macro. The upgraded program will continue to operate correctly on downlevel systems.

VSAM volume data set

The VSAM volume data set (VVDS) contains extent descriptors for VSAM data sets. For data sets that are eligible to be allocated or extended on an EAV volume, its extent descriptors may contain 28-bit cylinder addresses. Today these extents are stored in an internal VVDS structure called a VVR. Some programs may access the VVDS even though no interfaces are provided for this purpose. It is very important that these programs that refer to the extents within the VVR be reviewed, and possibly modified, to ensure they can handle the 28-bit cylinder addresses.

3.9 Migration considerations for using EAV volumes

For extended address volumes to use volume expansion,, an IBM System Storage DS8000 with the appropriate microcode level is required.

A software developer support table with EAV support information that also contains z/OS elements, features, and software products is available on the following Web site:

`http://www-03.ibm.com/systems/z/os/zos/software`

The following items affect the use of EAV support and may affect migration to this release when you are planning to implement extended address volumes.

VSAM CA system-computed CA sizes

In z/OS V1R10, the computed VSAM control area (CA) may be different than what was computed on prior releases. In z/OS V1R10, only one of the following CA sizes will be selected: 1, 3, 5, 7, 9, and 15 tracks. Adjustments to the primary and secondary space amount are made which may be different from prior releases.

DEVSERV QDASD and PATHS command changes

Fields are shifted to support larger number of cylinders and the number of cylinders (CYL) is added for PATHS.

IDCAMS LISTDATA PINNED

The track addresses in the output of IDCAMS LISTDATA PINNED are in a different format, irrespective of the volume size. Each track address will be printed in eight hex digits in the native format of the device, which is the format of CCCCcccH. In addition, instead of listing

one track per line, consecutive tracks for each data set will be gathered into one line and the range of track addresses will be shown.

IEHLIST LISTVTOC

Fields shifted to support larger numbers of cylinders. To handle larger numbers, many IEHLIST LISTVTOC fields are shifted to the right on the page even if the volume is small.

Programs that examine the capacity of a disk volume might find a discrepancy when adding the sizes of the data sets on the volume and comparing the sum to the volume size. That is because the number of cylinders on an EAV is in a different field as described, and the track addresses in the extent descriptions are different, as described previously.

IEHLIST output is not an intended programming interface but the output changes are mentioned here as a convenience.

OBTAIN SEARCH requests

OBTAIN SEARCH requests set additional bytes in the return area, and the caller must provide a 140-byte area per documentation. If your program passed the previous minimum of 101 bytes, unpredictable results would occur.

VTOC index data set

The VTOC index data set block size has been increased to 8192 bytes. Programs must be able to handle 2048 and 8192 byte blocks and be prepared to understand that the length and offsets of the bit maps in the index records will be different.

IPCS support of extended address volumes

IPCS has changed the use of OBTAIN to specify the EADSCB=OK option to exploit the new DSCB format. This allows IPCS to support larger volume sizes so customers can store more data on fewer devices.

SMF record type 19

SMF 19 record is now longer to provide additional fields for track-managed space and total volume and track-managed space size. The system writes an SMF type 19 records for each DASD volume that is online at IPL, when a **HALT** or **SWITCH** command is issued, and when a DASD is varied offline. It describes the space usage on the volume. The LSPACE macro creates and writes SMF type 19 records. It is mapped by the IGGSMF19 macro.

A new SMF19CYM bit means that the volume has cylinder-managed space in the flag SMF19FL1.

The following eight existing 2-byte counters describe free space for the entire volume. If the value exceeds the capacity of the counter, it will be set to the maximum of 65535. The correct values will always be in SMF19EVF and SMF19TMF.

Note that two fields have no label.

SMF19NDS	The number of data set control blocks (DSCB) calculated as the number of DSCBs per track times the number of tracks in the VTOC.
SMF19DSR	The number of data set control blocks (DSCB) - format 0 DSCBs; that is, the number of available DSCBs.
SMF19NAT	The number of unused alternate tracks.
SMF19SPC	The number of unallocated cylinders.
	The number of unallocated tracks.
SMF19LEX	The number of cylinders in the largest unallocated extent.
	The number of tracks in the largest unallocated extent.
SMF19NUE	The number of unallocated extents.

The following are new 4-byte unsigned counters in the record in the expanded SMF statistics:

SMF19SDS	Number of DSCBs.
SMF19SL0	Number of format 0 DSCBs.

The following fields contain free space statistics from the entire volume. For volumes with cylinder-managed space (SMF19CYM='1'), these statistics represent space from both the track- and cylinder-managed space on the volume. See SMF19TMF for statistics from the track-managed space on the volume. Fields are 4 bytes.

SMF19EVF	The total vol free space (20 bytes).
SMF19SUC	The number of free cylinders.
SMF19SUT	The number of additional free tracks.
SMF19SNC	The number of free cylinders in the largest free extent.
SMF19SNT	The number of additional free tracks in the largest free extent.
SMF19SNE	The number of free extents.

The following new counters apply only to the track-managed space on the volume. For volumes of 65520 cylinders or less (SMF19EAV is off), these counters are the same as the previous five counters for the whole volume. They are free space statistics from track-managed space on a volume for a volume with cylinder-managed space (SMF19CYM='1'). For volumes with no cylinder-managed space (SMF19CYM='0').

SMF19TMF	Track-managed free space (EBCDIC).
SMF19BUC	The number of free cylinders.
SMF19BUT	The number of additional free tracks.
SMF19BNC	The number of free cylinders in the largest free extent.
SMF19BNT	The number of additional free tracks in the largest free extent.
SMF19BNE	The number of free extents.

Two new fields are at the end the SMF type 19 record and are 4-byte fields.

SMF19TRK	The total number of tracks on the volume.
SMF19TRM	The total number of tracks in the track-managed space when SMF19CYM='1'. Set to the value of SMF19TRK otherwise. When SMF19CYM='1' then this value is also the first track address where cylinder-managed space begins.

3.9.1 Recommended migration actions

Following is a list of recommended actions for migrating to EAV-capable volumes:

- Look for OBTAIN, REALLOC, CVAFDIR, CVAFSEQ, CVAFDSM, and CVAFFILT macro calls.
- Look for programs that calculate volume or data set size by any means, including reading a VTOC or VTOC index directly with a BSAM or EXCP DCB. The system cannot distinguish between programs that read DSCBs for space information and those that read them for metadata. Both kinds of program require a new option.
- Look for EXCP, XDAP, and STARTIO macros for DASD channel programs and other programs that examine DASD channel programs or track addresses.
- Look for programs that examine any of the many operator messages that contain a DASD track or block address or data set or volume size. Such track or block addresses generally are represented in the documentation as cchh, cchhr, mbbcchhr, or track-address.
- Look for programs that access the VTOC index ("SYS1.VTOCIX.volser") and VVDS data sets ("SYS1.VVDS.Vvolser").

3.9.2 Coexistence maintenance for EAV support

The following APARs and PTFs must be installed on downlevel releases in support of the new functions available in z/OS V1R10 for extended address volumes.

DEVSERV QDASD command

You can use the **DEVSERV QDASD** command to display diagnostic information about the status of direct access storage devices and storage control units. You can also use it to validate

MVS storage resident control blocks for extended function status with the data acquired directly from the storage subsystem.

If you are using EAVs, install the appropriate coexistence PTF for APAR OA21487 on z/OS V1R9 or z/OS V1R8 if you have not already done so, so that the **DEVSERV QDASD** command will display the correct number of cylinders on z/OS V1R9 or z/OS V1R8. The **DEVSERV** command has the following changes:

- DEVSERV QDASD - fields are shifted to support larger number of cylinders
- DEVSERV PATHS - support for 3390 Model A (3390A)

Following is the **DEVSERV QDASD** command for an extended address volume:

```
DS QD,DC65,1
IEE459I 16.10.10 DEVSERV QDASD
 UNIT VOLSER SCUTYPE DEVTYPE       CYL  SSID SCU-SERIAL DEV-SERIAL EFC
 DC65 MLDC65 2107922 2107900     70119  89EC 0175-BALB1 0175-BALB1 *OK
****      1 DEVICE(S) MET THE SELECTION CRITERIA
****      0 DEVICE(S) FAILED EXTENDED FUNCTION CHECKING
```

ICKDSF R17

APAR PK56092 for ICKDSF R17 (which is the ICKDSF level in z/OS V1R10 down to z/OS V1R4), provides support for the extended address volume (EAV) function in z/OS V1R10. As part of this support, there is a change to the default size of a VTOC index when the INDEX parameter of the ICKDSF **INIT** command is defaulted. This default change occurs after you install the PTF for the APAR, regardless of whether you exploit EAV function.

Without the APAR, when the INDEX parameter is defaulted, the index is generated starting at the track following the end of the VTOC, with a size equal to the number of tracks per cylinder. However, if you specify VTOC(END), then the default index is located in the previous cylinder that precedes the VTOC and uses the whole cylinder.

With the APAR, when the INDEX parameter is defaulted, the index is generated starting at the track following the end of the VTOC. However, if you specify VTOC(END), the default index is located in the cylinder that precedes the VTOC. For both instances, the default size of the index is based on the required index records for a given volume size and the size of the VTOC.

LISTDATA PINNED command

Install APAR OA21487 for coexistence for the **LISTDATA PINNED** command with UNITNUMBER to be issued on pre-z/OS V1R10 systems to an EAV volume when the storage subsystems has pinned data. In this case, the **LISTDATA PINNED** command should be issued on a z/OS V1R10 system.

For IDCAMS **LISTDATA PINNED** command, this change returns the following message:

```
IDC31562 THE PINNED PARAMETER IS NOT AVAILABLE FOR THE SPECIFIED SUBSYSTEM OR
DEVICE for EAV volume.
```

PTFs are as follows:

```
Release 180   : UA40228
Release 190   : UA40229
```

LSPACE macro

APAR OA22449 adds support to DFSMS z/OS V1R7, V1R8, and V1R9 to allow these releases to map z/OS V1R10 LSPACE parameters XEXPMSG and EXPDATA to EXPMSG and DATA, respectively.

If this APAR is not applied, then when the V1R10 LSPACE new parameters XEXPMSG and EXPDATA are compiled, and the resulting load module is then executed on releases prior to DFSMS z/OS V1R10, a diagnostic code will be returned to the LSPACE issuer and the LSPACE call will be failed with a non-zero RC 12. It depends upon the issuing application as to the manifestations of the LSPACE failure.

The LSPACE macro changes for coexistence are in APAR OA22449 and PTFs as follows:

```
Release 1K0   : UA40219
Release 180   : UA40220
Release 190   : UA40221
```

DS8000 3390 Model A support

3390 Model A support was included in DS8000 R3 SPE APAR. This affected one operator command response, DEVSERV. It allowed `3390A` to be displayed instead of `3390` in the **`DESVERV PATHS`** command response. This was provided back to z/OS V1R7. The impact of not having this is that `3390` will be displayed. PTFs are available as follows:

```
UA38011 5695DF111- 1K0 (z/OS V1R7)
UA38012 5695DF111- 180 (z/OS V1R8)
UA38013 5695DF111- 190 (z/OS V1R9)
EREP V3R5 support for extended address volumes: IO03548
```

HSM Recall/Recover considerations

APAR OA22804 provides extended address volume (EAV) coexistence support for pre-V1R10 DFSMShsm versions. This APAR provides the following support:

- Restore EAS-eligible data sets from a backup or migrated copy on an online volume on z/OS V1R10.
- Enable pre-V1R10 systems to process the new VCC keywords that may be specified in the Management Class Backup Copy Technique field.
- Allow pre-V1R10 systems to recognize MCV records for EAVs.
- Restrict the selection of Extent Reduction request from the CRQ, if the request is directed to an EAV.
- To issue the new ARC0784I message and update FSR record if format 9 DSCB vendor attributes are lost during the RECALL/RECOVER/ARECOVER function.

Note: EAV volumes will not be accessible or able to be ADDVOLed on lower level (pre-V1R10) systems. The following errors can still occur in a mixed environment with the coexistence installed:

- Recall of a data set having a migration copy allocated on ML1/ML2 EAV will fail.
- Recall of a data set having a migration copy in an SDSP that resides on a ML1 EAV will fail.
- Recovery of a data set having a backup copy is allocated on a ML1 EAV will fail.
- Restore from dump of a data set will fail on the pre-V1R10 systems if the VTOC copy is allocated on ML1 EAV.
- If vendor attributes of the DSCB are lost during the RECALL/RECOVER/ARECOVER function, then the new ARC0784I message will be issued and the FSR record will be updated.

DSS restore considerations

APAR OA22900 provides toleration support when attempting to restore data from a DFSMSdss dump data set containing data from an extended address volume. Install this coexistence APAR on pre-z/OS V1R10 systems. Versions of DFSMSdss prior to V1R10 will not perform a full volume restore of a full volume dump from an EAV.

DFSMSdss Stand-Alone Restore will not perform a full volume or tracks restore of an EAV to a non-EAV volume. DFSMSdss full volume and tracks dumps of EAVs are not compatible with dumps of volumes that are 64 K cylinders or fewer due to changes required to format the extended-address space in the dump. Changes have been made in z/OS V1R10 DFSMSdss to identify dumps of EAVs.

When running more than one level of z/OS in an environment, or needing to provide data in the form of DFSMSdss dumps to out-centers that are not at the same z/OS level, DFSMSdss provides limited restore capability on supported lower levels of z/OS for data dumped from an EAV on z/OS V1R10 or higher levels.

In addition, tracks restore where track 0 is included will fail and tracks restore, not including track 0, will restore only the track-managed space from an EAV. Logical data set restore and physical data set restore are changed to convert format and format 9 DSCBs to format 1 DSCBs. When a data set is restored that had a format 8 and format 9 pair when it was dumped, if attributes are being lost due to the inability to restore the format 9, a new message, ADR556W, is issued. If no attribute values exist, no message will be issued.

DB2 V8 and V9 and extended address volumes

APAR PK58292 is needed for coexistence when you keep BSDS and active logs on non-EAV volumes and if you are not using Admin Enablement functions, you should be able to run DB2 V8 and V9 without any new DB2 EAV support code. Otherwise, the coexistence support to DB2 V8 and V9 will need to be applied.

Apply the appropriate coexistence PTFs before placing any BSDS or active logs on extended address volumes, or before using Admin Enablement functions on EAVs.

3.10 EAV migration assistance tracker

DFSMS is providing an EAV migration assistance tracker program. The tracking of EAV migration assistance instances uses the Console ID Tracking facility provided in z/OS V1R6. The EAV migration assistance tracker can help you find programs that you might need to

change if you want to support extended address volumes. The EAV migration assistance tracker is an extension of the console ID tracking facility. It helps you to do the following:

- Identify select systems services by job and program name, where the invoking programs might require analysis for changes to use new services. The program calls are identified as informational instances for possible migration actions. They are not considered errors, because the services return valid information.
- Identify possible instances of improper use of returned information in programs, like parsing 28-bit cylinder numbers in output as 16-bit cylinder numbers. These instances are identified as warnings.
- Identify instances of programs that will either fail or run with an informational message if they run on an EAV. These are identified as programs in error. The migration assistance tracker flags programs with the following functions, when the target volume of the operation is non-EAV, and the function invoked did not specify the EADSCB=OK keyword.

Note: Being listed in the tracker report does not imply that there is a problem. It is simply a way to help you determine what to examine to see whether it should be changed.

Error detection by the tracker

Each category of errors is explained here:

- Identify interfaces with access to the VTOC that should be upgraded to have EADSCB=OK specified for the following functions:
  - OBTAIN, CVAFDIR, CVAFDSM, CVAFVSM, CVAFSEQ, CVAFFILT, OPEN to VTOC, OPEN EXCP
- Identify programs that may want to use new services as informational messages.
- Identify the possible improper use of returned information, like parsing 28-bit cylinder numbers in output as 16-bit cylinder numbers as warning messages for the following commands and functions:
  - IEHLIST LISTVTOC, IDCAMS LISTCAT, IDCAMS LISTDATA PINNED
  - LSPACE, DEVTYPE, IDCAMS DCOLLECT

3.10.1 General information about the tracker

The tracker function allows component code to register tracking information as a text string of its choosing, of up to 28 characters in length. The tracker records this as a unique instance and appends additional information to it such as job name, program name, and count of occurrences, to avoid duplicates.

The tracker allows an exclusion list, via a SYS1.PARMLIB member, to specify to prevent an instance from being recorded. Instances already verified would then not show up in the tracker. The exclusion list filters will be the registered tracking information, job name, and program name. Wild-carding (? and *) will be allowed for these fields in the exclusion list.

In addition to these services the tracker will allow you, with an operator command, to activate the tracker, activate new exclusion lists, report on recorded instances, ABEND and dump on recorded instances, deactivate the tracker, and provide an IBM internet ID where a customer can provide instances of programs recorded in the tracker to IBM.

3.10.2 Migration assistance tracker commands

The tracking facility can be manipulated with the following commands:

- The **SETCON** command is used to activate and deactivate the Console ID Tracking facility.
- The **DISPLAY OPDATA,TRACKING** command is used to display the current status of the console ID tracking facility, along with any recorded instances of violations. Sample output of this command is shown in Figure 3-19 on page 90.

CNIDTRxx parmlib member

The CNIDTRxx parmlib member is used to list violations that have already been identified to prevent them from being recorded again.

An optional CNIDTRxx parmlib member can be defined to exclude instances from being recorded. The exclusion list is picked up when the tracker is started or via the **SET** command. The filters for the exclusion list are the three items that make an instance unique. You can exclude all SMS instances or just LSPACE instances, or exclude all instances created by job names that begin with BRS*. There are many more possibilities. The filters support wildcarding on these three fields.

The recommended exclusion list and the list of DFSMS instances are available at the following Web site:

`http://www-03.ibm.com/servers/eserver/zseries/zos/downloads/`

Recommended exclusion list on web site

Figure 3-17 on page 89 shows the Web site exclusion list that you can place into a CNTRIDxx parmlib member.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                             Jobname  Pgmname                           *
* Tracking Information Mask   Mask     Mask     Comments (ignored)       *
*---------------------------+--------+--------+----------------------+
|SMS-I:3 LSPACE*            |*MASTER*|IEE70110| Switch SMF           |
|SMS-I:3 LSPACE*            |ALLOCAS |IEFW21SD| VARY DEVICE OFFLINE  |
|SMS-I:3 LSPACE DATA=*      |*       |ADRDSSU | DSS DATA=            |
|SMS-I:3 LSPACE EXPMSG=*    |*       |ISRUDA  | ISPF EXPMSG=         |
|SMS-I:3 DEVTYPE*           |*       |IS*     | DEVTYPE Calls (ISPF) |
|SMS-I:3 DEVTYPE*           |*       |ADR*    | DEVTYPE Calls (DSS)  |
|SMS-I:3 DEVTYPE*           |*       |ASM*    | DEVTYPE Calls (ASM)  |
|SMS-I:3 DEVTYPE*           |*       |DISKMAP | DEVTYPE DISKMAP calls|
|SMS-I:3 DEVTYPE*           |*       |IEB*    | DEVTYPE Calls (Utili)|
|SMS-I:3 DEVTYPE*           |*       |CEE*    | DEVTYPE Calls (LDAP) |
|SMS-I:3 DEVTYPE*           |*       |EDG*    | DEVTYPE Calls (rmm)  |
|SMS-I:3 DEVTYPE*           |*       |ICH*    | DEVTYPE Calls (Racf) |
|SMS-I:3 DEVTYPE*           |*       |CEL*    | DEVTYPE Calls (LangE)|
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* To allow DFSMS instances to be recorded remove one or more of the   *
* entries by putting an asterisk in column 1 (or delete them)         *
* and continue tracking.                                              *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                             Jobname  Pgmname                           *
* Tracking Information Mask   Mask     Mask     Comments (ignored)       *
*---------------------------+--------+--------+----------------------+
|SMS*                       |*       |*       | SMS all instances    |
|SMS*E*                     |*       |*       | SMS Errors           |
|SMS*W*                     |*       |*       | SMS Warnings         |
|SMS*I*                     |*       |*       | SMS Informational    |
|SMS*:1*                    |*       |*       | EAV EADSCB=OK missing|
|SMS*:2*                    |*       |*       | EAV 28-bitcyls output|
|SMS*:3*                    |*       |*       | EAV new function     |
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
* End of SPECIAL PROCESSING FOR DFSMS                                  *
```

Figure 3-17 Web site exclusion list

Tracker exclusion list

The exclusion list prevents instances from being recorded. To identify or use an exclusion list, use the following operator command, as illustrated in Figure 3-18 on page 90.

```
set cnidtr=7t
IEE536I CNIDTR   VALUE 7T NOW IN EFFECT
```

```
SAMPLE EXCLUSION LIST 7T
*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*----------------------------+--------+--------+----------------------+
|SMS-I:3 LSPACE*             |*MASTER*|IEE70110| I SMF CALLS TO LSPACE|
|SMS-I:3 LSPACE*             |ALLOCAS |IEFW21SD| VARY DEVICE OFFLINE  |
|SMS-I:3 LSPACE*             |*       |VTDSOIS2| VTDSOIS2 PROG CALLS  |
|SMS-I:3 LSPACE*             |VTDSOIS1|*       | VTDSOIS1 JOB CALLS   |
```

Figure 3-18 Sample tracker exclusion list in CNIDTR7T parmlib member

Tracking command example

In Figure 3-19 on page 90, the tracking information column identifies the instance as being DFSMS-related with the SMS prefix. The additional I, E, or W appended to SMS identifies the instance as being an informational, error, or warning event.

The remaining text in the tracking information describes the event that was recorded or, for error events, the type of error that would have occurred if the function were executed on an EAV. The tracking value is a value unique to the error being recorded. JOBNAME, PROGRAM+OFF, and ASID identifies what was being run at the time of the instance.

Only unique instances are recorded. Duplicates are kept tracked by the NUM column being incremented. The tracking information, jobname, and program name fields make up a unique instance.

```
13.21.19 SYSTEM1           d opdata,tracking
13.21.19 SYSTEM1           CNZ1001I 13.21.19 TRACKING DISPLAY 831
STATUS=ON,ABEND NUM=15   MAX=1000 MEM=7T  EXCL=45     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFDIR STAT082        045201 CVAFJBN  CVAFPGM    756   28   4
SMS-E:1 CVAFDSM STAT082        045201 CVAFJBN  CVAFPGM    556   28   4
SMS-E:1 CVAFFILT STAT086     04560601 CVAFJBN  CVAFPGM    456   28   4
SMS-E:1 CVAFSEQ STAT082        045201 CVAFJBN  CVAFPGM    656   28   4
SMS-E:1 DADSM OBTAIN           C08001 OBTJBN   OBTPGM     856   28   4
SMS-E:1 DCB OPEN VSAM 113-44       01 OPENJBN  OPENPGM    256   28   4
SMS-E:1 DCB OPEN VTOC 113-48       01 OPENJBN  OPENPGM    356   28   4
SMS-I:3 DEVTYPE                    02 DEVTJOB  DEVTPROG  CE5C   11   1
SMS-I:3 IDCAMS DCOLLECT            02 DCOLLECT IDCAMS    1515   28   4
SMS-I:3 LSPACE  EXPMSG=          8802 VTDSOIS1 VTDSOIS2   118   28   2
SMS-I:3 LSPACE  MSG=             5002 ALLOCAS  IEFW21SD 4CE5C   11   2
SMS-I:3 LSPACE  MSG=             9002 *MASTER* IEE70110  52F6   01  43
SMS-W:2 IDCAMS LISTDATA PINN       03 LISTDATX IDCAMS    E48E   28   2
SMS-W:2 IDCAMS LISTCAT             03 LISTCAT  IDCAMS     956   28   4
SMS-W:2 IEHLIST LISTVTOC           03 LISTVTOC IEHLIST   1056   28   4
------------------------------------------------------------------------
TO REPORT THESE INSTANCES, SEND THIS MESSAGE VIA E-MAIL TO
CONSOLES@US.IBM.COM. FOR ADDITIONAL INFORMATION OR TO OBTAIN A CURRENT
EXCLUSION LIST, SEE APAR II13752.
```

Figure 3-19 Tracker instance report output

As events are recorded by the Tracking facility, report the instance to the product owner. After the event is reported, update the parmlib member so that the instance is no longer recorded by the facility. In this way, the facility will only report new events.

3.10.3 Special processing for the tracker

When a tracking instance occurs for a non-expanded address volume type, it will be recorded in the tracker as an error message. The possible instances are described in the following sections.

SMS-E:1 DADSM OBTAIN

DFSMS Tracking category 1: EAV Migration. Error Message. DADSM OBTAIN was issued with the search or seek option to a non-EAV volume. The caller did not specify with EADSCB=OK that it supports the extended attribute DSCBs and the target data set is 'EAS eligible'.

SMS-E:1 CVAFDIR STAT082

DFSMS Tracking category 1: EAV Migration. Error Message. CVAFDIR was issued with the search or seek option to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK that it supports the extended attribute DSCBs and the target data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a device that supports extended attribute DSCBs.

SMS-E:1 CVAFSEQ STAT082

DFSMS Tracking category 1: EAV Migration. Error Message. CVAFSEQ was issued for physical sequential or index order to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK that it supports the extended attribute DSCBs and the target data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a volume that supports extended attribute DSCBs.

SMS-E:1 CVAFDSM STAT082

DFSMS Tracking category 1: EAV Migration. Error Message. CVAFDSM was issued to retrieve unallocated space on a volume (CVAFDSM ACCESS=MAPDATA, MAP=VOLUME, RTA4BYTE=YES) that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK that it supports the extended attribute DSCBs. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a volume that supports extended attribute DSCBs.

SMS-E:1 CVAFFILT STAT086

DFSMS Tracking category 1: EAV Migration. Error Message. CVAFFILT was issued to obtain DSCB information for fully or partially qualified data set names on a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK that it supports the extended attribute DSCBs and the qualified data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of X'56' along with data set name status in the FCL (FCLDSNST of X'06' would have been set if the request was issued to a volume that supports extended attribute DSCBs.

SMS-E:1 CVAFVSM STAT082

DFSMS Tracking category 1: EAV Migration. Error message. CVAFVSM was issued to allocate space for a volume that was not an EAV. The caller did not specify with EADSCB=OK that it supports an EAV. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to an EAV. The CVAFVSM interface is an internal system function that is not documented externally for general use.

SMS-E:1 DCB OPEN VTOC 113-48

DFSMS Tracking category 1: EAV Migration. Error Message. A DCB Open of a VTOC was issued to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK on the DCBE macro that it supports the extended attribute DSCBs in the VTOC. Open would have issued an ABEND, MSGIEC142I 113-48 if an attempt was made to open the VTOC of a volume that supported extended attribute DSCBs.

SMS-E:1 DCB OPEN VSAM 113-44

DFSMS Tracking category 1: EAV Migration. Error Message. A DCB Open (MACRF = E for EXCP) of an EAS-eligible data set (VSAM) was issued to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK on the DCBE macro that it supports the extended attribute DSCBs for an EAS-eligible data set. Open would have issued an ABEND, MSGIEC142I 113-44 if an attempt was made to open the EAS eligible data set on a volume that supported extended attribute DSCBs.

SMS-I:3 LSPACE EXPMSG=, DATA=, MSG=

DFSMS Tracking category 3: EAV Migration. An LSPACE request with the DATA=, MSG=, or EXPMSG= keywords was issued. Additional data from track-managed space is available with the EXPDATA= and XEXPMSG= keywords. When this instance occurs for any volume type, it will be recorded in the tracker as an informational message. New function is available.

SMS-I:3 DEVTYPE

DFSMS Tracking category 3: EAV Migration. Informational Message. New function is available. Additional data from DEVTYPE INFO=DASD invocation is available. See mapping macro IHADVA.

SMS-I:3 IDCAMS DCOLLECT

DFSMS Tracking category 3: EAV Migration. Informational Message. An IDCAMS DCOLLECT request for 'V' (Volume Record Field) and 'VL' (SMS Volume Definition Field) records was processed. Additional data for track-managed space was recorded.

SMS-W:2 IDCAMS LISTDATA PINN

DFSMS Tracking category 2: EAV Migration. Warning Message. An IDCAMS LISTDATA PINNED request was processed. The track addresses for the PINNED tracks may contain 28-bit cylinder numbers.

SMS-W:2 IEHLIST LISTVTOC

DFSMS Tracking category 2: EAV Migration. Warning Message. An IEHLIST LISTVTOC request was processed. Extent descriptors may contain cylinder addresses 65 520 or larger. Free space descriptors may contain track addresses 982800 or larger and/or full cylinders 65520 or larger. The generated report will display the information in different columns as compared to reports generated on releases prior to z/OS V1R10.

SMS-W:2 IDCAMS LISTCAT

DFSMS Tracking category 2: EAV Migration. Warning Message. An IDCAMS LISTCAT request was processed that printed extent descriptors for one or more EAS-eligible data sets (VSAM in z/OS V1R10). The returned extent descriptors may contain 28-bit cylinder numbers. This instance is recorded for both EAS- and non-EAS-capable volumes. Note that AMS LISTCAT output format may change as a result of service and new function support. IBM recommends applications processing LISTCAT output be updated to obtain results directly from the Catalog Search Interface (CSI). For more information on CSI, see *z/OS*

DFSMS Managing Catalogs, SC26-7409 and *High Level Assembler for z/OS & z/VM & z/VSEProgrammer's Guide*, SC26-4941.

3.11 Data migration to EAV volumes

The prerequisites for the implementation of extended address volumes in production is the z/OS V1R10, DFSMS V1R10, and a new level of support for the DS8000 on the LCU level for systems sharing the device. The EAV feature is only available on the DS8000.

The DS8000 requires the Version 4.0 Licensed Internal Microcode level. The HMC installation of this code implements the necessary upgrade to the DS8000 Storage Manager to work with the full features of dynamic volume expansion (DVE). Also required is an upgrade of the DS CLI to level 5.4.0.262 to be able to use the full functions as an alternative to the DS8000 Storage Manager.

Defining new EAVs

When defining new EAV volumes, consider the following criteria needed:

- The new 3390 Model A is now a selectable volume type and is used in place of the previous model types such as 3390-3, 3390-9, and 3380.
- You can use the following methods to define the EAV volume:
  - DS CLI to define the new 3390 Model A, as explained in "Command-line interface (DSCLI)" on page 94
  - DS8000 Storage Manager, which is a Web browser GUI interface as explained in "Using the IBM System Storage DS8000 Storage Manager" on page 95

Note: The Model A type only applies to the DS8000. If you expand a 3390 volume to an extended addressing volume (EAV), the data type is changed to 3390-A. Attempting to reduce the size returns an error and causes the transaction to fail.

3.11.1 Dynamic volume expansion

The IBM System Storage DS8000 series supports dynamic volume expansion. Dynamic volume expansion increases the capacity of existing zSeries volumes, while the volume remains connected to a host system.

This capability simplifies data growth by allowing volume expansion without taking volumes offline. Using DVE significantly reduces the complexity of migrating to larger volumes.

Note: For the dynamic volume expansion function, volumes cannot be in Copy Services relationships (point-in-time copy, FlashCopy SE, Metro Mirror, Global Mirror, Metro/Global Mirror, and z/OS Global Mirror) during expansion.

It is possible to grow an existing volume with the DS8000 with dynamic volume expansion (DVE). Previously, it was necessary to use migration utilities that require an additional volume for each volume that is being expanded and require the data to be moved.

A logical volume can be increased in size while the volume remains online to host systems for the following types of volumes:

- 3390 model 3 to 3390 model 9.
- 3390 model 9 to EAV volume sizes using z/OS V1R10. Dynamic volume expansion can be used to expand volumes beyond 65520 cylinders without moving data or causing an application outage.

There are two methods to dynamically grow a volume:

- Use the command-line interface (DSCLI); refer to “Command-line interface (DSCLI)” on page 94 for more information.
- Use a Web browser GUI; refer to “Web browser GUI” on page 95 for more information.

Note: All systems must be at the z/OS V1R10 level for the DVE feature to be used when the systems are sharing the Release 4.0 Licensed Internal Microcode updated DS8000 at a LCU level.

3.11.2 Command-line interface (DSCLI)

You can use a command-line interface (DSCLI), an SMI-S industry-standard-compatible API.

The **chckdvol** command changes the name of a count key data (CKD) base volume. It can be used to expand the number of cylinders on an existing volume.

New options

A new parameter on the **chckdvol** command allows a volume to be increased in size.

- **-cap new_capacity** (Optional and DS8000 only)

  This specifies the quantity of CKD cylinders that you want allocated to the specified volume. 3380 volumes cannot be expanded. For 3390 Model A volumes (DS8000 only), the **-cap** parameter value can be in the range of 1 to 65,520 (increments of 1) or 65,667 to 262,668 (increments of 1113). For 3390 volumes, the **-cap** parameter value can be in the range of 1 to 65,520 (849 KB to 55.68 GB).

This example shows how to use DSCLI to change the size of a 3390 Mod 3 device, ID 0860 (actual device address D860) to a 3390 Mod 9:

```
chckdvol –cap 10017 0860
```

Figure 3-20 on page 94 shows the change of the device to a 3390 Mod 9, which is displayed using the Web browser GUI interface.

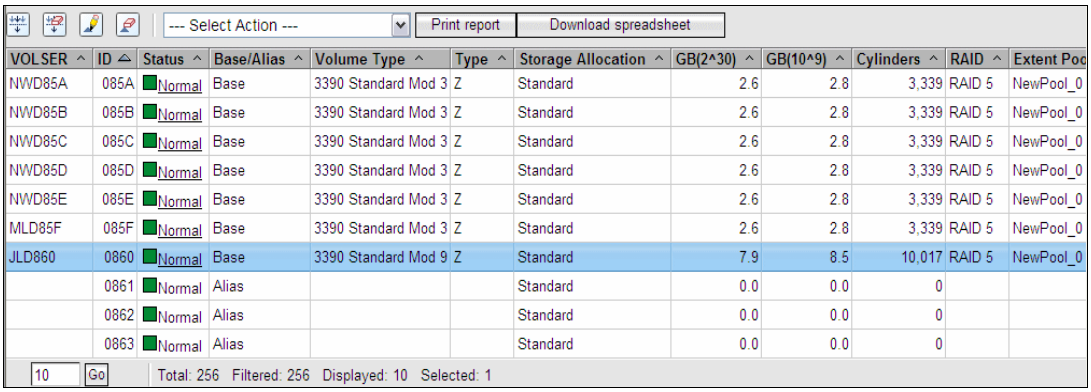


Figure 3-20 Device change from a 3390 Mod 3 to a 3390 Mod 9

3.11.3 Using the IBM System Storage DS8000 Storage Manager

The 3390 Model A can be defined as new or can be defined when a new EAV volume beyond 65520 cylinders is specified. The number of cylinders when defining an EAV volume can be defined as low as 1 cylinder and up to 262668 cylinders.

In addition, expanding a 3390 type of volume that is below 65520 cylinders to be larger than 65520 cylinders converts the volume from a 3390 standard volume type to a 3390 Model A and thus an expanded address volume.

Web browser GUI

You can use an intuitive GUI to manage the system from any Web browser capable of accessing the system management server. The GUI can be used in offline mode, for performing simulated configuration modeling. To increase the number of cylinders on an existing volume, you can use the Web browser GUI by selecting the volume (shown as volser NWDF64 in Figure 3-21 on page 95) and then using the Select Action pulldown to select **Increase Capacity**.

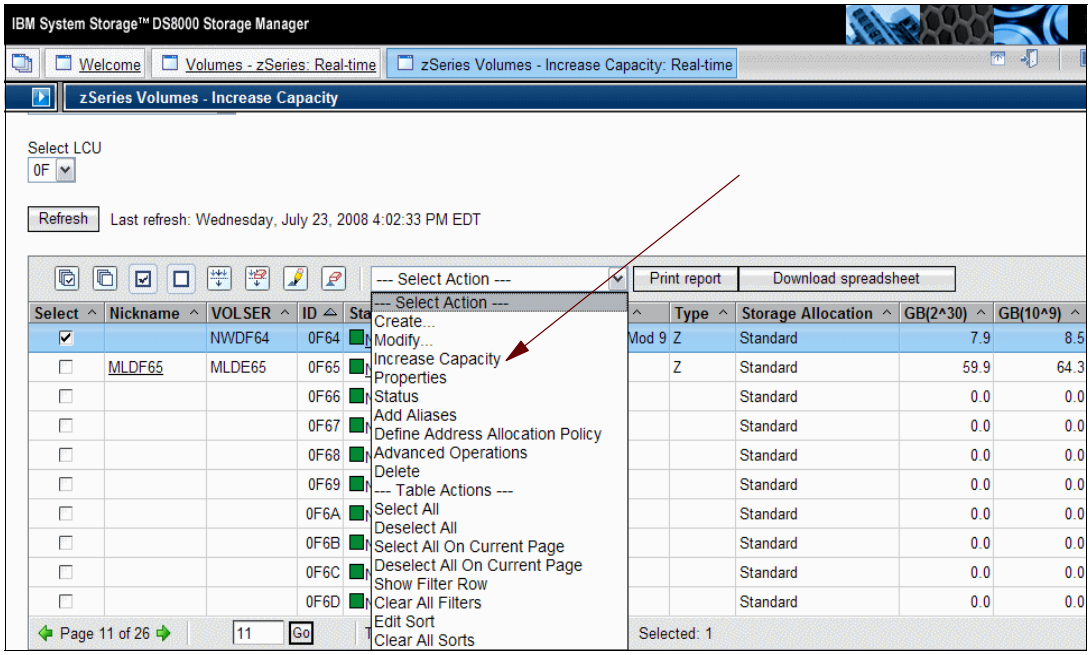


Figure 3-21 Screen to increase the size of a volume

Figure 3-22 on page 96 is displayed after you select Increase Capacity. A warning message is issued regarding a possible volume type change to 3390 custom. You can then select **Close Message**. Notice that the panel displays the maximum number of cylinders that can be specified.

Note: Only volumes of type 3390 model 3, 3390 model 9, and 3390 custom, can be expanded. The total volume cannot exceed the available capacity of the storage image. Capacity cannot be increased for volumes that are associated with Copy Services functions.

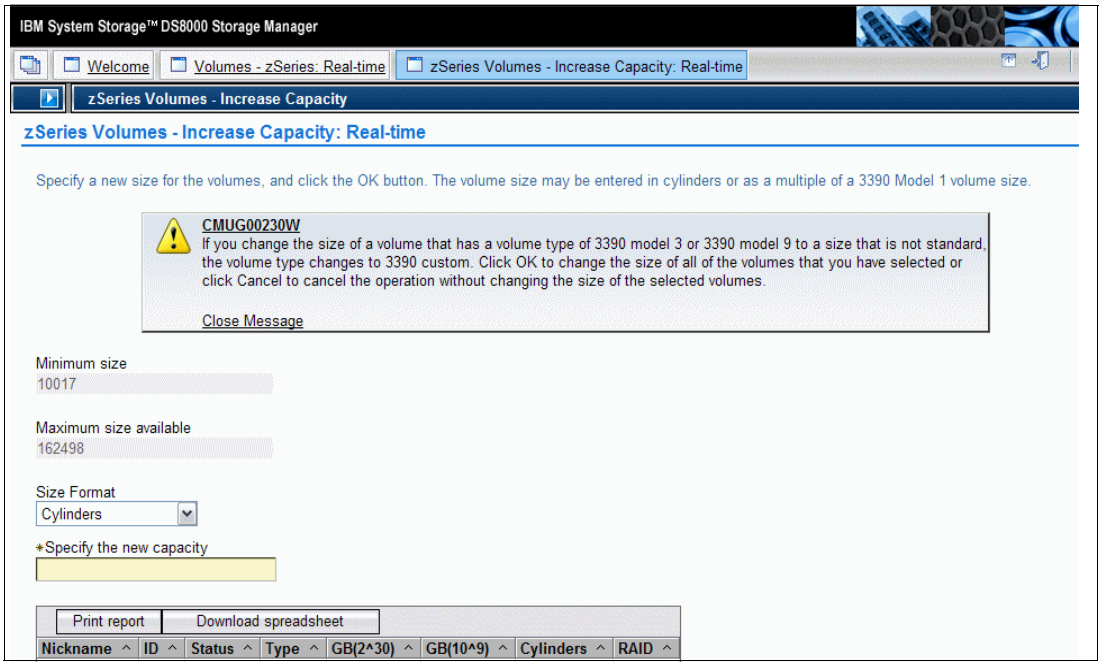


Figure 3-22 Warning message issued regarding the volume type change

After you close the message, the panel in Figure 3-23 on page 96 is displayed where you can specify the number of cylinders to increase the volume size. When you specify a new capacity to be applied to the selected volumes, specify a value that is between the minimum and maximum size values that are displayed. Maximum values cannot exceed the amount of total storage that is available.

.

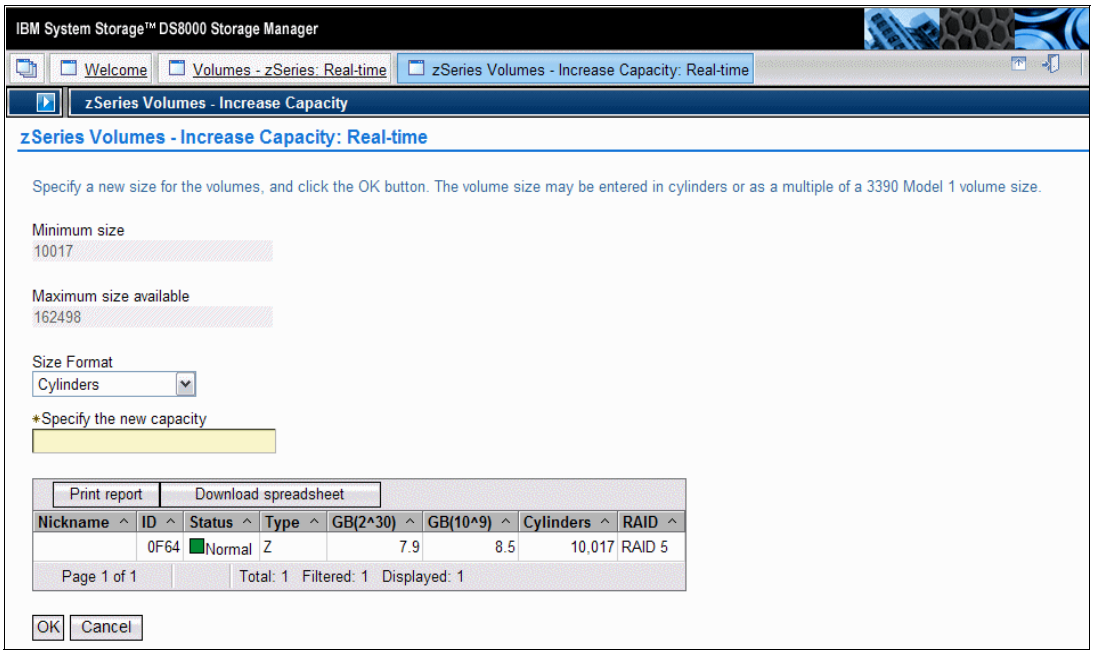


Figure 3-23 Panel to specify the new volume capacity

Thus, if 75000 cylinders is specified on the panel shown in Figure 3-23 on page 96, the message displayed in Figure 3-24 on page 97 is issued when you click **OK**.

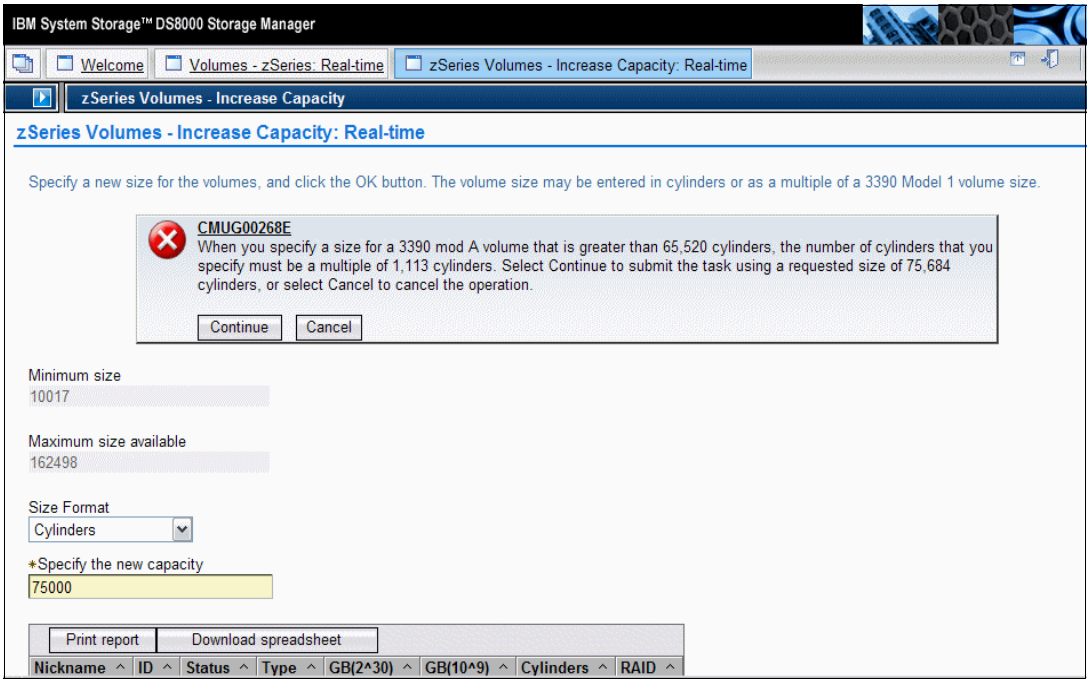


Figure 3-24 Message showing the actual number of cylinders required

Specify **Continue**, and a requested size of 75,684 is processed for the expansion of the volume.

Note: Remember that the number of cylinders must be as stated earlier. The reason an MCU value is 21 cylinders is because it is derived from being the smallest unit that can map out the largest possible EAV and stay within the index architecture (with a block size of 8192 bytes). It is also a value that divides evenly into the 1 GB storage segments of a DS8000. These 1 GB segments are the allocation unit in the DS8000 and are equivalent to 1113 cylinders. Data sets allocated in cylinder-managed space may have their requested space quantity rounded up to the next MCU.

3.11.4 Update VTOC after volume expansion

The **REFORMAT** command must be used to rebuild the VTOC and index structures after a volume has been expanded. The command updates portions of a previously initialized volume. You can also enter ICKDSF commands with Interactive Storage Management Facility (ISMF), which is a component of DFSMS.

If an index exists when you expand the VTOC, it must be deleted and rebuilt to reflect the VTOC changes. These parameters are used to specify the total track size of the index to be rebuilt.

- EXTINDEX(n)
- XINDEX(n)

For 'n', substitute the decimal or hexadecimal digits (for example, X'1E') to specify the total number of tracks for the new index after expansion. If the value for 'n' is less than the current value, the current value is used.

3.11.5 Adding EAVs using SMS

SMS storage groups, storage pools/esoteric and updated ACS routines can be used to create extended address volumes.

Extended address volumes (EAVs) and non-EAV volumes may reside in the same volume pools. SMS prefers EAV volumes for EAS-eligible data sets that are equal to or larger than the BPV. For non- EAS-eligible requests that are smaller than the BPV, SMS will not have a preference for EAV volumes.

3.12 Using EAV volumes

The DS8000 series offers enhancements to Parallel Access Volumes (PAV) with support for HyperPAV, which is designed to enable applications to achieve equal or better performance than PAV alone, while also using the same or fewer operating system resources.

HyperPAV is an optional licensed function that you can use in conjunction with the parallel access volumes (PAV) function. IBM HyperPAV associates the volumes with either an alias address or a specified base logical volume number. When a host system requests IBM HyperPAV processing and the processing is enabled, aliases on the logical subsystem are placed in an IBM HyperPAV alias access state on all logical paths with a given path group ID. IBM HyperPAV is only supported on FICON channel paths.

3.12.1 EAV installation parmlib changes

Following are the IGDSMSxx parmlib member changes for EAV volumes.

USEEAV(YES|NO)

USEEAV(YES|NO) specifies, at the system level, whether SMS can select an extended address volume during volume selection processing. This check applies to new allocations and when extending data sets to a new volume.

YES This means that extended address volumes can be used to allocate new data sets or to extend existing data sets to new volumes.

NO (Default) This means that SMS does not select any extended address volumes during volume selection. Note that data sets might still exist on extended address volumes in either the track-managed or cylinder-managed space of the volume.

Note: You can use the **SETSMS** command to change the setting of USEEAV without having to re-IPL. This modified setting is in effect until the next IPL, when it reverts to the value specified in the IGDSMSxx member of PARMLIB.

To make the setting change permanent, you must alter the value in SYS1.PARMLIB. The syntax of the operator command is:

```
SETSMS USEEAV(YES|NO)
```

BreakPointValue (0- 65520)

BreakPointValue (0- 65520) is used by SMS in making volume selection decisions and subsequently by DADSM. If the allocation request is less than the BreakPointValue, the system prefers to satisfy the request from free space available from the track-managed space.

If the allocation request is equal to or higher than the BreakPointValue, the system prefers to satisfy the request from free space available from the cylinder-managed space. If the preferred area cannot satisfy the request, both areas become eligible to satisfy the requested space amount.

Note: The BreakPointValue is only used to direct placement on an expanded address volume.

Generally, for VSAM data set allocation requests that are equal to or larger than the BreakPointValue, SMS prefers extended address volumes for:

- Non-VSAM allocation requests
- VSAM allocation requests that are smaller than the BreakPointValue, SMS does not have a preference.

The default is 10 cylinders.

Note: You can use the **SETSMS** command to change the setting of BreakPointValue without having to re-IPL. This modified setting is in effect until the next IPL when it reverts to the value specified in the IGDSMSxx member of PARMLIB. To make the setting change permanent, you must alter the value in SYS1.PARMLIB. The syntax of the operator command is:

```
SETSMS BreakPointValue(0-65520)
```

3.12.2 SMS volume selection

SMS uses storage groups to contain the definitions of volumes that are managed similarly. Each storage group has the following defined thresholds:

- A high allocation threshold

  SMS uses the high allocation threshold to determine candidate volumes for new data set allocations.

- A low migration threshold

  Volumes with occupancy lower than the high allocation threshold are selected in favor over those volumes that contain more data than the high allocation threshold specifies. DFSMShsm uses the low migration threshold during primary space management. It uses the interval threshold during interval migration to determine when to stop processing data.

Extended address volume selection considerations

The two new SMS IGDSMSxx parmlib member parameters need to be set before you can use extended address volumes with SMS for volume selection, as follows:

- USEEAV
- BreakPointValue

For an extended address volume, the system and storage group break point value (BPV) helps direct disk space requests to cylinder-managed or track-managed space. The breakpoint value is expressed in cylinders and is used as follows:

- When the size of a disk space request is the breakpoint value or more, the system prefers to use the cylinder-managed space for that extent. This rule applies to each request for primary or secondary space for data sets that are eligible for the cylinder-managed space.

- If cylinder-managed space is insufficient, the system uses the track-managed space or uses both types of spaces.
- When the size of a disk space request is less than the breakpoint value, the system prefers to use the track-managed space.
- If space is insufficient, the system uses the cylinder-managed space or uses both types of spaces.

SMS allocation thresholds

For EAS-eligible data sets on volumes that support cylinder-managed space, the allocation threshold is divided into categories. All categories are assessed to determine the volumes capability of meeting the threshold requirements. These thresholds include the volumes capability in meeting the track-managed space, as follows:

- Cylinder-managed space thresholds
- Total volume space threshold

Note that the Allocation Threshold percentage applies to both cylinder-managed space and total volume space. These thresholds are further classified as primary or secondary thresholds, as follows:

- For space requests that are less than the break point value, the primary threshold is the track-managed space and the secondary threshold is the total volume space.
- For space requests that are equal to or greater than the break point value, the primary threshold is the cylinder-managed space and the secondary threshold is the total volume space.

3.13 Summary

A foreseeable constraint of z/OS running out of z/OS addressable disk storage has been addressed by defining an extended address volume (EAV), which extends the number of cylinders per device to be >65520 by utilizing a new track addressing structure.

Using EAV, z/OS customers benefit from of a huge (more than 100 TB) increase in the amount of storage that z/OS can now address. EAS-eligible and supported data sets in z/OS V1R10 include:

- SMS-managed VSAM (all types)
- Non-SMS VSAM (all types)
- zFS data sets (which are VSAM LS)
  - zFS aggregates are supported in an EAV environment.
  - zFS aggregates/file systems can reside on an extended address volume (EAV).
  - zFS aggregates/file systems can reside in track-managed space or cylinder-managed space (subject to any limitations that DFSMS may have).
  - zFS still has an architected limit of 4 TB for the maximum size of a zFS aggregate.
- Database (DB2, IMS) use of VSAM.

EAV documentation

Refer to the following DFSMS publications for more information about extended address volumes.

- *z/OS DFSMS Advanced Copy Services*, SC35-0428

- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407
- *z/OS DFSMS Installation Exits*, SC26-7396
- *z/OS DFSMS Introduction*, SC26-7397
- *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408
- *z/OS DFSMS Managing Catalogs*, SC26-7409
- *z/OS DFSMS Storage Administration Reference (for DFSMSdfp, DFSMSdss, DFSMShsm)*, SC26-7402
- *z/OS DFSMS Using the New Functions*, SC26-7473
- *z/OS DFSMS Using Data Sets*, SC26-7410
- *z/OS DFSMS Using ISMF*, SC26-7411
- *z/OS DFSMSdfp Advanced Services*, SC26-7400
- *z/OS DFSMSdfp Diagnosis*, GY27-7618
- *z/OS DFSMSdfp Utilities*, SC26-7414
- *z/OS DFSMSdss Storage Administration Guide*, SC35-0423
- *z/OS DFSMShsm Diagnosis*, GC52-1083
- *z/OS DFSMShsm Implementation and Customization Guide*, SC35-0418
- *z/OS DFSMShsm Managing Your Own Data*, SC35-0420
- *z/OS DFSMShsm Storage Administration Guide*, SC35-0421

4

64-bit common virtual storage

z/OS V1R10 provides support for common storage above the 2 GB bar. A new virtual storage area, the high common storage area (HCSA), is defined. Storage management services and RMF support for HCSA are also planned. This new support provides the infrastructure required for many users of CSA and ECSA storage to move data above the bar. This is expected to lead to virtual storage constraint relief (VSCR) over time.

Many system components currently get CSA or SQA storage using the GETMAIN or STORAGE macro. As we strive for VSCR, many of these components will be asked to move their blocks above the bar. In addition to all the effort to change to amode 64 to access storage above the bar, many of these components would need to create a 64-bit storage manager. This support allows components to make a simple substitution of macro IARST64 for either GETMAIN or STORAGE.

This chapter describes the following functional changes:

- 64-bit common virtual storage

4.1 64-bit common virtual storage

What is common area virtual storage constraint relief (VSCR) and why is it important? Common area virtual storage usage has been a critical constraint issue over the years and will continue to be a critical limiting factor in application growth. This chapter's main objective is to describe how the common area virtual storage constraint has been relieved.

To achieve this aim, a 64-bit common virtual storage support is introduced in z/OS V1R10, as well as large page support in the hardware and architecture DAT of 1 MB.

4.2 Common area VSCR overview

Figure 4-1 shows the 31-bit address storage areas where the storage areas are continuing to grow. A balance is needed between the minimum required private area sizes for specific applications against a larger common area needed for the overall system health. The storage areas shown have the following concerns:

- 24-bit (below 16 M) common storage use continues to grow.
- 31-bit (16 M to 2 G) common storage use continues to grow at about 5%- 20% each release because the following applications and storage areas have problems:
  - Middleware and applications require larger private areas for optimal operation (such as the DB2 DBM1 private area).
  - LPA areas are growing as well in the 50 M-100 M range.
  - The common area is squeezing the private area and this continues to be a well-known issue.

Both the private and common areas need to add up to 2 G. If the common area is expanded, addressing is taken away from the private local area, and vice versa.

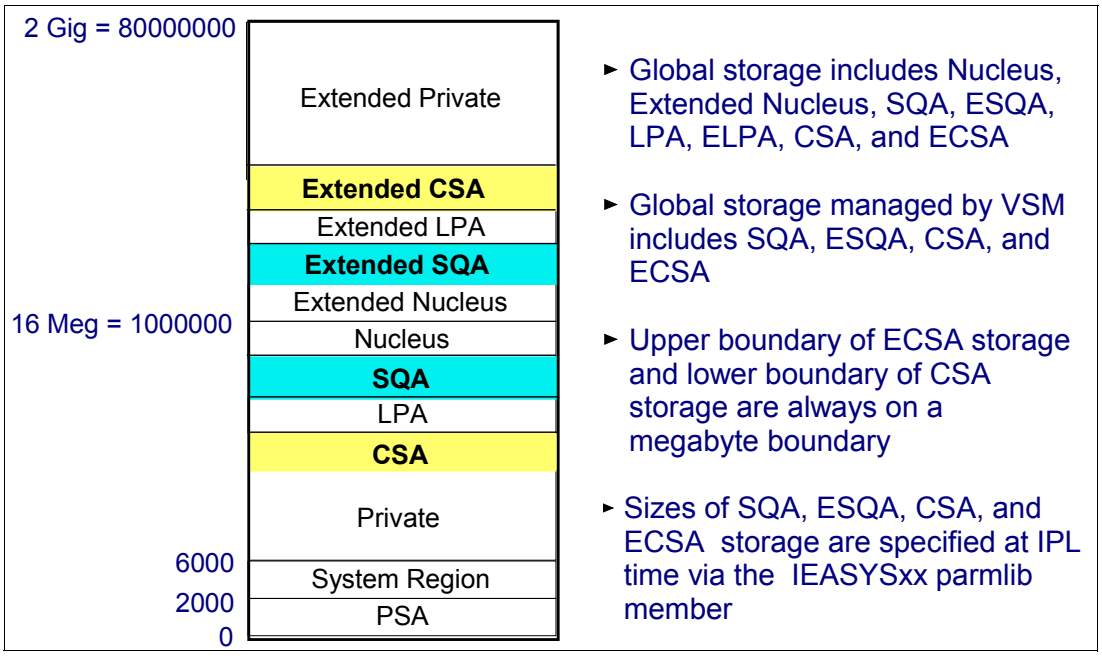


Figure 4-1 31-bit address space memory map

With the 64-bit address space, shown in Figure 4-2, additional relief was added but not for the 31-bit storage areas. The common area below 2 GB cannot accommodate current or future workload needs for common storage. Without additional VSCR relief, z/OS images would be capped, limiting customer growth and resulting in outages.

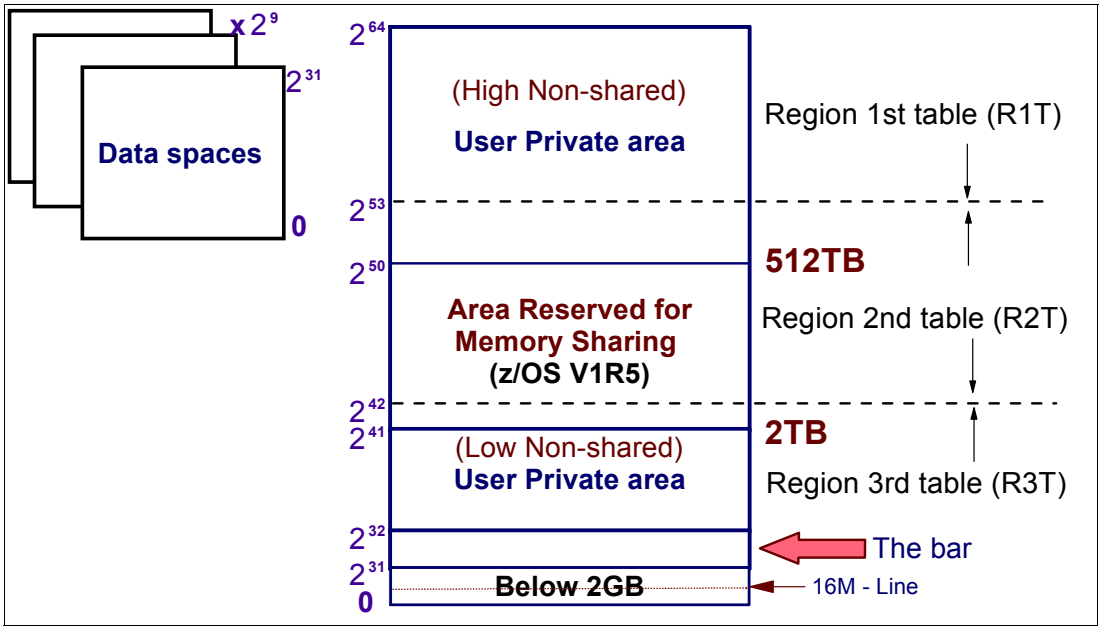


Figure 4-2 z/OS V1R9 addressability

4.2.1 Alleviating common area VSCR

To address this problem, several actions should be taken:

- Move data out of common, whenever possible.
- Where to move data depends on attributes of the required storage. 64-bit common is the direction IBM has decided to focus on exploiting.
- Data traditionally in the common area needs to move elsewhere:
  - Private
  - Common addressed dataspaces (CADs), still a strategic direction
  - 64-bit shared
  - 64-bit common with z/OS V1R10
- IBM has established specific common storage savings goals for VSCR relief, for every z/OS release starting with z/OS V1R9.

Identify candidates to move

All authorized code (from IBM, vendors, and customer-owned) has a role to play in meeting VSCR goals. z/OS, middleware, and applications that are authorized to obtain common storage, need to consider to move data out of the common area.

To help provide relief for 24-bit and 31-bit common storage shortages, a z/OS brand-wide effort has been launched to reduce common storage usage. Because all products use these areas to some degree, all products will need to contribute to the reduction.

Moving data currently located in common

Where to move depends on attributes of the required storage. To determine that, we must firstly analyze its current use of common storage and answer the following questions:

- Why is the block in common storage and how is the block accessed—by one address space, by a set of selected address spaces, or by every and any address space in the system.
- Does the block need fixed storage and is the block used for I/O.
- Does the block need disabled reference storage and do you need to split the block. Does the block contain data that is input to services that only accept 31-bit or non-ALET qualified parameters. Does the block contain data that must remain in 31-bit common storage (lock words for example).

4.2.2 Evaluate storage move alternatives

There are potentially four different alternatives for choosing where to move data from under the 2 GB common storage area, as follows:

- Move to private

  Does the data really need to be in common storage? If not, move it to the private area. Considerations are to move to 31-bit private if there is a small amount of data, or move to 64-bit private if there is a large amount of data or data growth is based on workload. Then, other address spaces can access the data if a proper ALET is used. Storage can be DREF or fixed and is either task or address space owned. Isolation is at its maximum with this first alternative.

  Note: All storage supported by the system is either fixed, pageable, or disabled reference (DREF). Because the system resolves page faults occurring on DREF storage synchronously, your program can reference DREF storage while it runs disabled for I/O and external interrupts. Your program, therefore, can use DREF storage in place of fixed storage if it needs to reference storage while it is disabled. An advantage of DREF storage is that it need not be backed by central storage frames until it is referenced. However, if the system cannot obtain a frame of central storage to back the DREF storage when it is referenced, the program referencing the storage is ended abnormally.

- Move to SCOPE=COMMON dataspaces (CADS)

  A SCOPE=COMMON dataspace can be used by all programs in the system. It provides a commonly addressable area similar to the common storage area (CSA). SCOPE=COMMON dataspaces have been used extensively by quite a number of different MVS components and can also be used by subsystems and applications that need common storage.

  Using a dataspace ALET on PASNAL, any address space can access the CADs using AR mode. If more than 2 GB of virtual is needed, more dataspaces have to be created because although the z/OS architecture allows 64-bit dataspaces, z/OS V1R10 does not yet provide this capability. The MAXCAD parameter specified in the IEASYSxx parmlib member can have a maximum value of 250. MAXCAD specifies the maximum number of SCOPE=COMMON dataspaces to be allowed during an IPL. It reserves the number of entries available for SCOPE=COMMON dataspaces on all primary address space access lists (PASN-ALs) in the system, hence its common scope. The maximum total number of ALETs on a PASNAL is 510.

A larger MAXCAD means less entries for other spaces. Each SCOPE=COMMON data space uses one entry on all PASN-ALs in the system. Because the maximum number of entries in a PASN-AL is currently 510, each SCOPE=COMMON data space your program creates and adds to the PASN-AL, reduces the number of SCOPE=SINGLE and SCOPE=ALL data spaces that a program can address through its PASN-AL. When you select a value for MAXCAD, you must take into account the number of SCOPE=COMMON dataspaces that subsystems, applications, and MVS use.

Note: A re-IPL is required if adding another CAD and exceeding the MAXCAD parameter. With z/OS V1R10, MAXCAD cannot be modified in an IPL. Should this prevent z/OS continuity, proper enhancements in the z/OS supervisor should allow this VSCR constraint to be relieved in a future release.

Storage is managed by a component or application-written storage manager. If requirements become strong enough, a set of functions (equivalent to the 64-bit addressing scheme) for managing memory objects in SCOPE=COMMON dataspaces might be introduced in a future release.

Since z/OS V1R9, dataspaces can be accessed from C programs, thanks to the metal option of the C compiler, the so-called “Metal C”.

- Move to 64-bit shared memory

  This allows sharing above the bar across multiple address spaces. Allocating storage via the IARV64 GETSHARED macro does not give other address spaces access to the storage. Storage must be explicitly freed and each address space needs to explicitly obtain access via a IARV64 SHAREMEMOBJ request. A 64-bit shared memory object can not be fixed or use DREF. Any isolation is based on the need for an address space to explicitly obtain access to the shared memory object, in order to access it. Though with a strong RAS value, such an approach is not as efficient as the hardware-defined capability-based addressing available with SCOPE=COMMON dataspaces.

- Move to 64-bit common, new in z/OS V1R10

  This alternative allows sharing above the bar across every address space. Any address space can access storage without explicitly having to request access. This allows authorized code to obtain storage and storage can only be assigned system keys 0-7 (to limit overlays by unauthorized code). This storage can be DREF and fixed. The storage needs to be explicitly freed. Any address space (even the one staying in 31-bit mode) has a Region 3 table in real memory.

  The LFAREA parameter in the IEASYSxx parmlib member has a fixed and a maximum value. The maximum amount of real storage that can be used to back large pages is 80% of the online storage available at IPL minus 2 GB. A re-IPL is required if the number of large page frames in real storage has to be changed. With z/OS V1R10, the LFAREA cannot be modified in an IPL.

  In terms of RAS, given that any address space can access storage in the 64-bit common area, without explicitly having to request access, this is the lowest of the 4 envisioned alternatives. In fact, we are back where MVS was some 25 years ago when XA was available and the advanced address space facility of ESA was not yet announced. It indeed gave then the opportunity to benefit from a larger addressing scheme and to have common areas (such as ECSA, ESQA), much the same way 64-bit common is designed.

Important: The order of the considered alternatives implies a decreasing RAS level, and for alternative 4, 64-bit common storage areas provides a far lower RAS level for the overall z/OS structure than alternatives 1 and 2.

Summary of alternatives

Table 4-1 summarizes where to move the data. It all depends on the storage attributes of the required storage as well as the implied RAS level that is to be achieved.

Table 4-1 Summary of the alternatives

Storage attributes	Private (MVS)	CADS (AASF)	64-bit Shared (z/OS V1R5)	64-bit Common (z/OS V1R10)
Accessed by one space	Natural	Ideal for data isolation	Not best solution	Not best solution
Accessed by a set of spaces	Poorly efficient	Ideal for both RAS and efficiency	Natural if scalability not a problem	Possible but potential of overlays
Accessed by every space	Inappropriate	Ideal for 10's GB with RAS and efficiency	Possible but cumbersome when large scale	Easy for 100's GB but potential RAS exposures.
DREF storage	Yes for 31-bit No for 64-bit	Yes	No	Yes
Fixed storage	Yes	Yes for internal callers	No	Yes
Storage ownership	Task or address space	Task	System - storage must be explicitly freed	System - storage must be explicitly freed

Known candidates for the common storage area

Current z/OS common storage VSCR focus areas are:

- I/O Supervisor of z/OS
- DB2
- Communication storage manager used by IP Services
- IMS
- Potential future areas of interest:
  - MQ, CICS, and WebSphere Application Server
- ISV cooperative initiative
  - IBM will assist ISVs in identifying common storage relief opportunities in ISV code
- Have profiled customers with VSCR issues to help us identify key VSCR areas

4.3 64-bit common storage overview

The 64-bit common area size can be specified via the HVCOMMON keyword in the IEASYSxx parmlib member:

```
HVCOMMON ={xxG} {xxT}
```

The HVCOMMON parameter specifies the size of the 64-bit common area. The IARV64 macro allows the allocation of common virtual storage above 2 GB. When defining a 64-bit common storage area, it starts down from 2**41. When reserving, the actual amount of common storage consumed is the amount specified with the HVCOMMON parameter plus 2 GB for operating system purposes.

The 64-bit common area is placed below the 2 TB line (in the R3T area as shown in Figure 4-3). The HVCOMMON value can also be specified by responding to the IEA101A message.

Value Range: 2 gigabytes to 1 terabyte (which is 1024 gigabytes) and is rounded up by 2 GB.

Default Value: 64 gigabytes .

From z/OS V1R10 on, the 64-bit address space memory map can be depicted as in Figure 4-3.

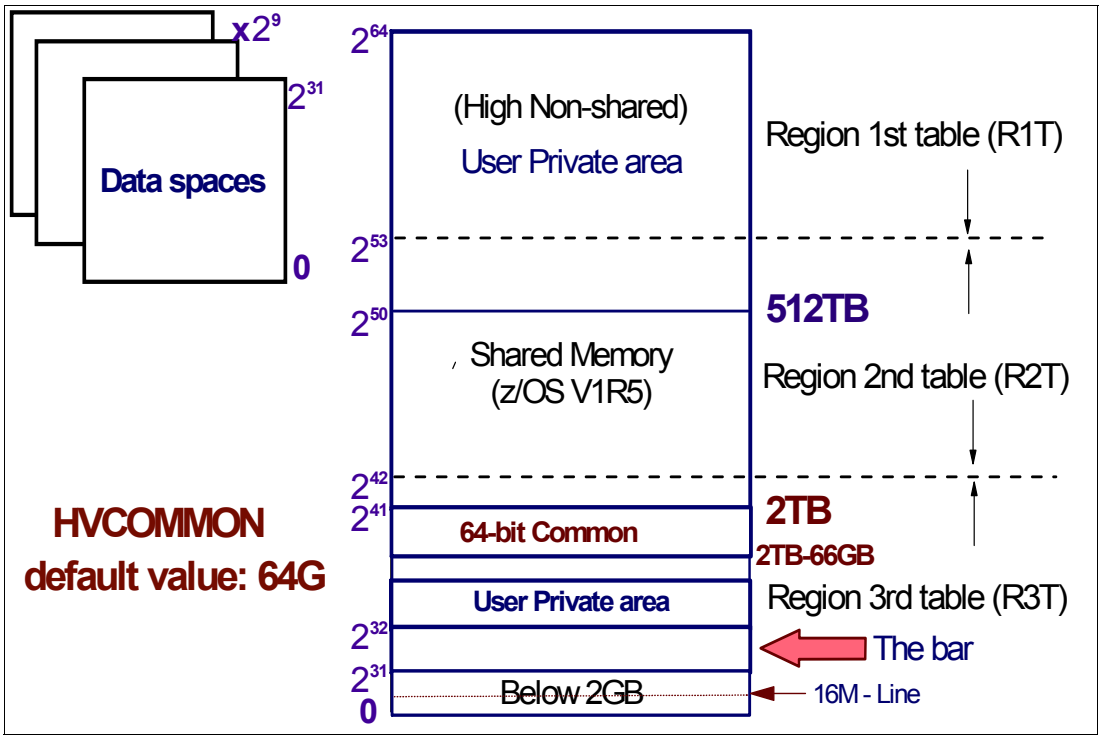


Figure 4-3 z/OS V1R10 addressability with default value

Operator commands

Use the DISPLAY VIRTSTOR command to identify the virtual storage configuration. The following information is displayed in message IAR019I. When the DISPLAY VIRTSTOR,HVCOMMON command is specified, the following information is provided:

- Source of HVCOMMON parameter can be a parmlib member, operator supplied, or the default.
- The size of the 64-bit common area in gigabytes, in decimal.
- The range of the 64-bit common area in gigabytes, in decimal.
- The amount of 64-bit common area that is allocated in megabytes, in decimal.

Figure 4-4 displays the 64-bit common range, and how much of the 64-bit common area has been allocated in the system, which has an additional 2 GB for the operating system purposes.

```
D VIRTSTOR,HVCOMMON
IAR019I  16.47.33 DISPLAY VIRTSTOR 470
  SOURCE = DEFAULT
  TOTAL 64-BIT COMMON = 66G
  64-BIT COMMON RANGE = 1982G-2048G
  64-BIT COMMON ALLOCATED = 4M
```

Figure 4-4 DIsplaying the HVCOMMON specification

SRM monitoring

64-bit common storage is monitored by SRM. SRM monitors how much of the specified 64-bit common storage (HVCOMMON=) is currently in use and alerts the operator when the usage exceeds certain thresholds. Messages are issued by SRM when 80% or 95% of the 64-bit common storage is in use. One of the following messages may be issued:

```
IRA130E HIGH COMMON SHORTAGE
IRA131E CRITICAL HIGH COMMON SHORTAGE
IRA132I HIGH COMMON SHORTAGE RELIEVED
```

4.4 Using 64-bit common storage

z/OS V1R10 provides two new ways to obtain and manage storage in the 64-bit common area, as follows:

- Using the IARV64 macro by chunks of 1 MB for storage managed as memory objects
- Using the IARST64 and IARCP64 macros for smaller pieces of storage, which are managed in the same way as their equivalent was managed in the old 2 GB address spaces:
  - IARST64 as a previous GETMAIN or STORAGE OBTAIN would do
  - IARCP64 to manage pools of storage cells

4.4.1 64-bit common memory objects

64-bit common storage (Figure 4-5 on page 111) is managed with memory objects that can be created via a new option on the IARV64 macro, as follows:

```
IARV64 REQUEST=GETCOMMON
```

The 64-bit common memory objects are allocated and used as follows:

- In 1 MB multiples on a 1 MB boundary, just like GETSTOR and GETSHARED.
- They are visible at the same address in every address space and every address space has access to the memory object once it is created.
- They ave a single protection key (only keys 0-7 are allowed) and fetch protection attribute.
- They must be explicitly freed.
- For diagnostic purposes they can be associated with an owner of HOME | PRIMARY | SYSTEM | BYASID.

- They can group a set of related 64-bit common memory objects by specifying a memory object token.

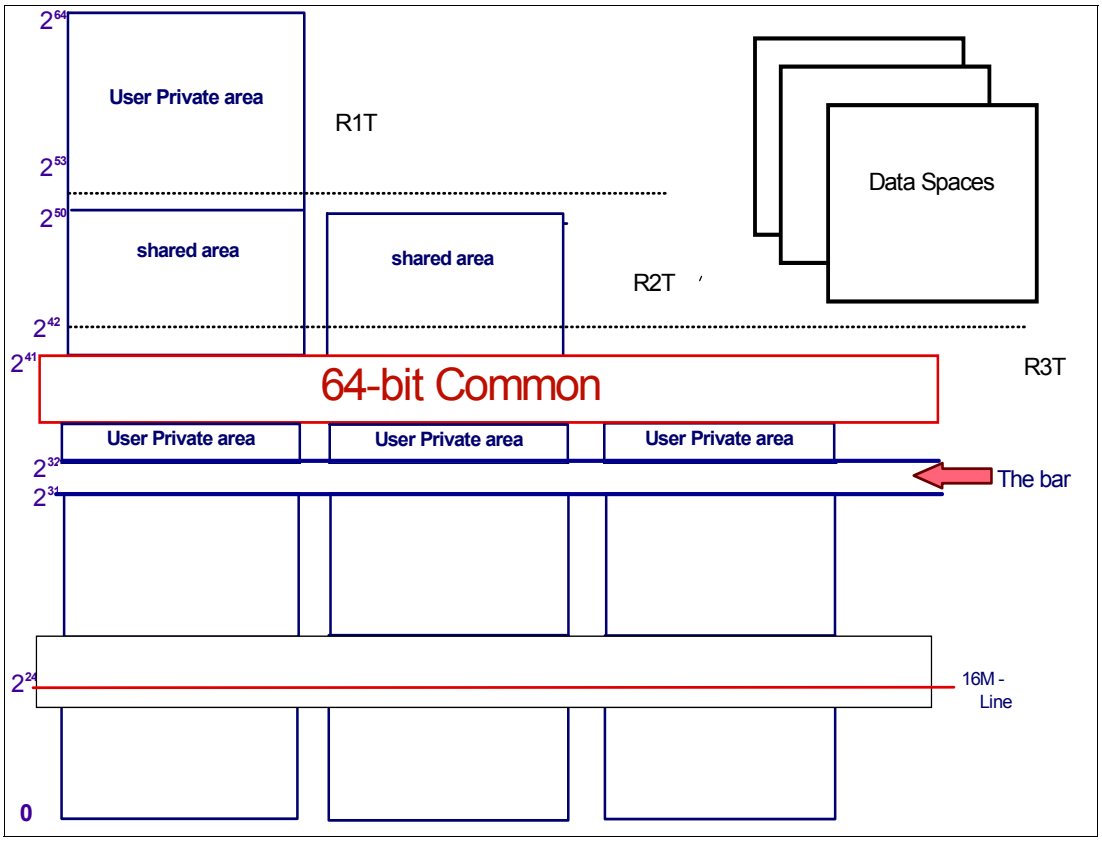


Figure 4-5 64-bit common storage area with z/OS V1R10

4.5 IARV64 REQUEST= type calls

With z/OS V1R10, the new and changed REQUEST= type calls are as follows:

GETCOMMON, PAGEFIX, PROTECT, DETACH, and GETSTOR

The storage returned by the GETSTOR, GETSHARED, or GETCOMMON services is called a memory object and is called as follows:

- The storage returned by GETSTOR is called a private memory object.
- The storage returned by GETCOMMON is called a common memory object.

4.5.1 IARV64 REQUEST=GETCOMMON required syntax

The new REQUEST=GETCOMMON requests that a 64-bit common memory object be created. The memory object is visible at the same virtual address in every address space in the system and is accessible by every address space in the system. Successful completion of this service creates the memory object. The memory object must be explicitly freed (via DETACH AFFINITY=SYSTEM).

The 64-bit common areas are requested via IARV64 with a new set of required and optional parameters or subparameters, as in Figure 4-6.

```
IARV64 REQUEST=GETCOMMON
   SEGMENTS=xsegments
   ORIGIN=xorigin
   KEY=xkey|CALLERKEY
   MOTKNSOURCE=USER
   MOTKN=xmotkn|NO_MOTKN
   MOTKNSOURCE=SYSTEM
   OUTMOTKN=xoutmotkn
   PAGEFRAMESIZE=4K|1MEG|MAX
   TYPE=PAGEABLE|DREF
```

Figure 4-6 IARV64 REQUEST=GETCOMMON request

The IARV64 REQUEST=GETCOMMON required parameters are as follows:

MOTKNSOURCE= **USER | SYSTEM** - Is an optional keyword input that indicates the source of the memory object token to be associated with this memory object. DEFAULT: USER

USER - The user is providing the memory object token.

SYSTEM - The system provides the memory object token.

MOTKN= **xmotkn | 0** - Is the name (RS-type), or address in register (2)-(12), of an optional doubleword input that identifies the token to be associated with the memory object. This must be a token that was returned by the system on a previous GETCOMMON request via the OUTMOTKN keyword. If you specify no user token, the default is that no user token is supplied to associate this memory object with others.

DEFAULT: NO_MOTKN

OUTMOTKN= **xoutmotkn** - Is the name (RS-type), or address in register (2)-(12), of a required doubleword output in which the system returns the token associated with this memory object. This token can be used on subsequent GETCOMMON requests as a user-supplied token in order to associate other memory objects with this token. This token can be supplied on a later DETACH request in order to free all the memory objects that have been associated with the token.

TYPE= **PAGEABLE | DREF** - Is an optional keyword input that specifies the type of the requested storage. The TYPE keyword is honored when PAGEFRAMESIZE=4K or when PAGEFRAMESIZE=MAX is specified and the memory object is backed with 4K page frames. Note the TYPE keyword is ignored when PAGEFRAMESIZE=1MEG is specified or when PAGEFRAMESIZE=MAX is specified and the memory object is backed with 1 M page frames. DEFAULT: PAGEABLE

PAGEABLE - Pages backing this memory object are pageable. Pages will be backed at first reference and can be paged out to AUX. Virtual address ranges in the memory object can be explicitly fixed after allocation by using the IARV64 REQUEST=PAGEFIX request.

DREF - The memory object is referenced while running disabled. Note that the DREF attribute applies to the entire memory object. Pages are backed in real at first reference. Pages belonging to memory objects

with the TYPE=DREF attribute will remain in real and will never be paged out to AUX.

4.5.2 REQUEST=GETCOMMON optional syntax

Optional keywords have also been introduced in the IARV64 macro syntax, as follows:

OWNERCOM= **HOME | SYSTEM | PRIMARY | BYASID** - Is an optional keyword input that specifies the entity to which the system will assign ownership of the 64-bit common memory object. The system uses this ownership information to track the use of 64-bit common storage for diagnostic purposes. DEFAULT: HOME

HOME - The home address space is assigned as the owner of the 64-bit common memory object

PRIMARY - The primary address space is assigned as the owner of the 64-bit common memory object.

SYSTEM - The 64-bit common memory object is not associated with an address space. The system is assigned as the owner of the 64-bit common memory object.

BYASID - The address space specified by OWNERASID is assigned as the owner of the 64-bit common memory object.

OWNERASID= **xownerasid** - Is the name (RS-type), or address in register (2)-(12), of an optional halfword input that specifies the ASID of the address space that will own the 64-bit common memory object for tracking purposes.

DEFAULT: HOME

DUMP= **LIKECSA | LIKESQA | NO | BYOPTIONVALUE** - Is an optional keyword input that specifies whether the 64-bit common memory object is included in an SVC dump when CSA or SQA is specified on SDATA. When TYPE=PAGEABLE is specified on IARV64 REQUEST=GETCOMMON the default is DUMP=LIKECSA. When TYPE=DREF is specified on IARV64 REQUEST=GETCOMMON the default is DUMP=LIKESQA.

LIKECSA - The 64-bit common memory object will be included in an SVC dump when CSA is specified on SDATA.

LIKESQA - The 64-bit common memory object will be included in a SVC dump when SQA is specified on SDATA.

NO - The 64-bit common memory object will not be included in a SVC dump when either CSA or SQA is specified on SDATA.

BYOPTIONVALUE - The 64-bit common memory object will be dumped according to the option specified by the OPTIONVALUE keyword.

OPTIONVALUE= **xoptionvalue** - Is the name (RS-type), or address in register (2)-(12), of a required byte input that contains one of the dump option values as specified by the bit constants.

DETACHFIXED= **NO | YES** - Is an optional keyword input that specifies whether the memory object can be detached when it contains fixed pages at the time of the DETACH request.

DEFAULT: NO

NO - The memory object will not be detached if it has any fixed pages when it is being detached.

YES - The memory object will be detached even if some or all the pages of that memory object are fixed.

Storage tracking

An owner can be associated with the 64-bit common memory object when the memory object is created for storage tracking allocation purposes using:

```
OWNERCOM=HOME | PRIMARY | SYSTEM | BYASID
```

Storage tracking keeps track of the following information:

- Date and time when the memory object was created
- Primary and home ASIDs at the time the allocation request was made
- Address of the program that made the request
- Owner, ASID, jobname, and jobid
- Date and time when the owner terminated the request

4.5.3 REQUEST=PAGEFIX

Authorized programs can fix 4K pages in memory objects in real storage for I/O or other purposes by issuing the request shown in Figure 4-7.

```
IARV64 REQUEST=PAGEFIX
   RANGLIST=xranglist
   NUMRANGE=xnumrage|1
   LONG=YES|NO
```

Figure 4-7 IARV64 REQUEST=PAGEFIX request

RANGLIST= **xranglist** - Is the name (RS-type), or address in register (2)-(12), of a required 8-byte input that contains the address. The range list consists of a number of entries (as specified by NUMRANGE) where each entry is 16 bytes long (Figure 4-8 on page 115). A description of the fields in each entry is as follows:

VSA - Is the starting address of the data to be acted on. The address specified must be within a memory object returned by GETSTOR CONTROL=AUTH (not GETSHARED), or a memory object returned by GETCOMMON. The value must always be on a physical page boundary. The length of this field is 8 bytes.

NUMPAGES - Contains the number of physical pages to be acted on. The number of pages specified starting with the specified VSA must lie within a single memory object. The length of this field is 8 bytes.

NUMRANGE= **xnumrange | 1** - Is the name (RS-type), or address in register (2)-(12), of an optional fullword input that specifies the number of entries in the supplied range list. The value specified must be no greater than 16. DEFAULT: 1.

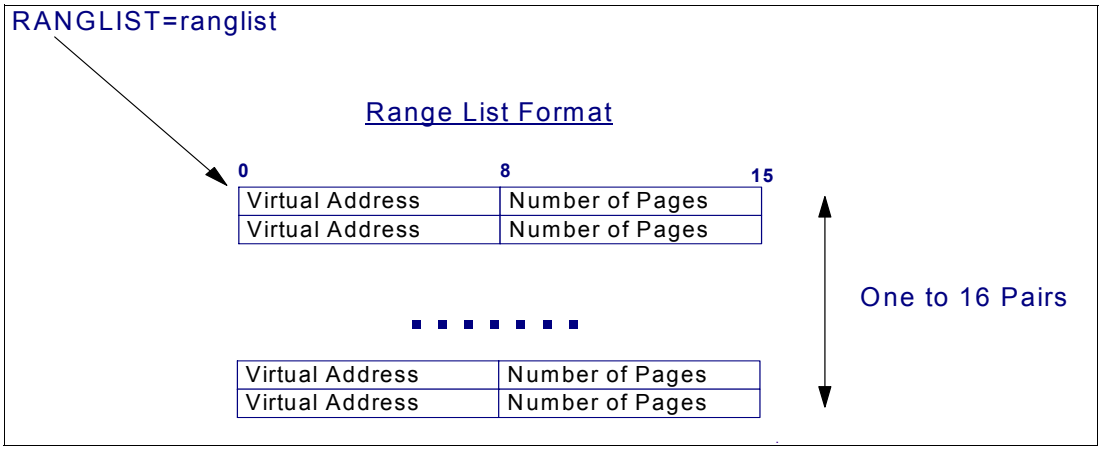


Figure 4-8 IARV64 REQUEST=PAGEFIX RANGLIST

4.5.4 REQUEST=PROTECT

REQUEST=PROTECT notifies the system that data within the specified ranges should be made read-only. Areas of the memory object that are in guard areas or hidden are affected. The caller must have either PSW key 0 or a PSW key of the storage being protected; see Figure 4-9.

```
IARV64 REQUEST=PROTECT
   RANGLIST=xranglist
   NUMRANGE=xnumrage|1
```

Figure 4-9 IARV64 REQUEST=PROTECT requests

The RANGLIST is specified as in Figure 4-8 on page 115 and the keywords have the same meaning as specified in “REQUEST=PAGEFIX” on page 114.

4.5.5 REQUEST=DETACH

REQUEST=DETACH frees one or more memory objects, as shown in Figure 4-10.

```
IARV64 REQUEST=DETACH
   MATCH=SINGLE|MOTOKEN|USERTOKEN
   MOTKN=xmotkn
   MOTKNCREATOR=USER|SYSTEM
   AFFINITY=SYSTEM
   V64COMMON=YES
   COND=YES|NO
```

Figure 4-10 IARV64 REQUEST=DETACH requests

MATCH= **SINGLE | MOTOKEN | USERTOKEN** is an optional keyword input that indicates which memory objects are to be freed. DEFAULT: SINGLE

SINGLE specifies that the input contains MEMOBJSTART, for a single memory object. **MEMOBJSTART=xmemobjstart** is the name (RS-type), or address in register (2)-(12), of a required 8-byte input that contains the address of the first byte in the memory object.

MOTOKEN specifies that the input contains a memory object token that was passed to GETSTOR, GETSHARED or SHAREMEMOBJ. Note that memory objects not associated with a memory object token are not affected. (Such objects would have to have been created using GETSTOR without MOTKN/USERTKN).

Note: If MATCH=MOTOKEN or MATCH=USERTOKEN with COND=YES, and no matching memory object token exists, the system returns a return code instead of abending the caller. USERTOKEN is a synonym for MOTOKEN. The following is a set of mutually exclusive keys. This set is optional; only one key may be specified.

MOTKN= **xmotkn** belongs to a set of mutually exclusive keys. It is the name (RS-type), or address in register (2)-(12), of an optional doubleword input that identifies the memory object token to uniquely identify the memory object, as previously passed to GETSTOR, GETSHARED or SHAREMEMOBJ, or the token that was generated by the system on a GETCOMMON, GETSTOR, or GETSHARED requests. When the memory object is not associated with the input token value, it is not to be processed.

MOTKNCREATOR= **USER | SYSTEM** is an optional keyword input that indicates who created the memory object token. DEFAULT: USER

USER is the memory object token that is user-created.

SYSTEM is the memory object token that is system-created.

USERTKN= **xusertkn | NO_USERTKN** belongs to a set of mutually exclusive keys. It is the name (RS-type), or address in register (2)-(12), of an optional doubleword input that is synonym for MOTKN. You can use either USERTKN or MOTKN interchangeably, but a usertkn is always presumed to be user-created.

DEFAULT: NO_USERTKN

This is the end of a set of mutually exclusive optional keys. The default is MOTKN.

AFFINITY= **LOCAL | SYSTEM** is an optional keyword input that specifies whether local or system affinity for the memory object is to be affected.

DEFAULT: LOCAL

LOCAL affinity to the memory object is to be affected, i.e. the interest in the memory object defined by the input ALETVALUE and memory object token. Private memory objects are affected only by AFFINITY LOCAL. Shared memory objects for which an appropriate SHAREMEMOBJ has been done for the address space defined by the input ALETVALUE will also be affected by AFFINITY LOCAL. Common memory objects are not affected by AFFINITY=LOCAL.

SYSTEM affinity to the memory object is to be affected. This is only relevant to shared memory objects and common memory objects. SYSTEM can be used only by callers running in supervisor state or with PSW key 0-7.

V64COMMON= **NO | YES** is an optional keyword input that specifies whether this DETACH request is for common memory objects.DEFAULT: NO

NO indicates the detach request is for common memory objects.

YES indicates the detach request is not for common memory objects.

4.5.6 REQUEST=GETSTOR

REQUEST=GETSTOR creates a memory object; see Figure 4-11. The initial value of storage within a memory object is binary zero. At completion, the memory object is created in the address space you indicate on the ALETVALUE parameter. If you do not specify ALETVALUE, the system creates the memory object in the primary address space. Note that problem state routines running in PSW key 8-15 can use GETSTOR only when primary = home. When the memory object owner terminates, the memory object is freed.

```
IARV64 REQUEST=GETSTOR
   SEGMENTS=xsegments
   ORIGIN=xorigin
   TYPE=PAGEABLE|DREF
   MEMLIMIT=YES|NO
   SVCDUMPRGN=YES|NO
   DUMP=LIKERGN|LIKELSQA|NO|BYOPTIONVALUE
      ,DUMPPRIORITY={xdumppriority|99}
   OPTIONVALUE=xoptionvalue
   KEY=xkey|CALLERKEY
   FPROT=YES|NO
   DETACHFIXED=YES|NO
```

Figure 4-11 IARV64 REQUEST=GETSTOR requests

The new keywords available with z.OS V1R10 are as follows:

TYPE= **PAGEABLE | DREF** is an optional keyword input that specifies the type of the requested storage. The TYPE keyword will be honored when PAGEFRAMESIZE=4K or when PAGEFRAMESIZE=MAX is specified and the memory object is backed with 4K page frames. Note the TYPE keyword is ignored when PAGEFRAMESIZE=1MEG is specified or when PAGEFRAMESIZE=MAX is specified and the memory object is backed with 1M page frames. DEFAULT: PAGEABLE

PAGEABLE - Pages backing this memory object will be pageable. Pages will be backed at first reference and can be paged out to AUX. Virtual address ranges within the memory object can be explicitly fixed after allocation by using the IARV64 REQUEST=PAGEFIX request.

DREF - The memory object will be referenced while running disabled. Note that the DREF attribute applies to the entire memory object. Pages will be backed in real at first reference. Pages belonging to memory objects with the TYPE=DREF attribute will remain in real and will never be paged out to AUX.

MEMLIMIT= **YES | NO** is an optional keyword input that specifies whether the allocation of the 64-bit private memory object is to count towards the address space MEMLIMIT. DEFAULT: YES

YES indicates that the 64-bit private memory object will contribute towards the address space MEMLIMIT.

NO indicates that the 64-bit private memory object will not be counted against the address space MEMLIMIT. MEMLIMIT=NO can only be specified by authorized callers in supervisor state or key 0-7.

SVCDUMPRGN= **YES | NO** - SVCDUMPRGN and DUMP are mutually exclusive keys. This set is optional; only one key may be specified. An optional input parameter that specifies whether the memory object should be included

in an SVC dump when a region is requested. The default is YES for TYPE=PAGEABLE. If neither the SVCDUMPRGN keyword or the DUMP keyword are specified, the defaults that apply are as described under the defaults for the DUMP keyword.

YES - The memory object should be included in an SVC dump when RGN is specified on SDATA. This is equivalent to DUMP=LIKERGN.

NO - The memory object should not be included in an SVC dump when RGN is specified on SDATA.

DUMP= **LIKERGN | LIKELSQA | NO | BYOPTIONVALUE** is an optional keyword input that specifies whether the 64-bit private memory object will be included in an SVC dump when RGN or LSQA is specified on SDATA. When TYPE=PAGEABLE is specified on IARV64 REQUEST=GETSTOR the default is DUMP=LIKERGN. When TYPE=DREF is specified on IARV64 REQUEST=GETSTOR, the default is DUMP=LIKELSQA.

LIKERGN - The 64-bit common memory object will be included in an SVC dump when RGN is specified on SDATA.

,DUMPPRIORITY={xdumppriority|99} is the name (RS-type), or address in register (2)-(12), of an optional byte input that specifies the dump priority of the memory object. This must be a non-zero value in the range of 1 to 99, with 1 being the highest priority and 99 being the lowest. DEFAULT: 99. See “Dump priority of LE memory objects” on page 122.

LIKELSQA - The 64-bit private memory object will be included in an SVC dump when LSQA is specified on SDATA.

NO - The 64-bit private memory object will not be included in an SVC dump when either RGN or LSQA is specified on SDATA.

BYOPTIONVALUE - The 64-bit common memory object will be dumped according to the option specified by the OPTIONVALUE keyword.

OPTIONVALUE= **xoptionvalue** is the name (RS-type), or address in register (2)-(12), of a required byte input that contains one of the dump option values.

FPROT= **YES | NO** - An optional input parameter that specifies whether the memory object should be fetch protected. The default is FPROT=YES.

YES - The entire memory object will be fetch protected. A program must have a PSW key that matches the storage key of the memory object (or have PSW key 0) to reference data in the memory object.

NO - The memory object will not be fetch protected.

DETACHFIXED= **NO | YES** - Is an optional keyword input that specifies whether the memory object can be detached when it contains fixed pages at the time of the DETACH request. DEFAULT: NO

NO - The memory object will not be detached if it has any fixed pages when it is being detached.

YES - The memory object will be detached even if some or all the pages of that memory object are fixed. DETACHFIXED=YES can only be specified by authorized callers in supervisor state or key 0-7.

4.5.7 Memory objects with IARV64

To use the storage in a memory object, the program must be in AMODE 64. To get there, a program in AMODE 24 or AMODE 31 uses the assembler instruction SAM64. While in AMODE 64, a program can issue only the IARV64 macro. The parameter lists the program passes to IARV64 can reside above or below the bar.

The IARV64 macro allows a program to use the full range of virtual storage in an address space that is supported by 64-bit addresses. The macro creates and frees storage areas above the 2-gigabyte address and manages the physical frames behind the storage. Each storage area is a multiple of one megabyte in size and begins on a megabyte boundary. You can think of the IARV64 macro as the GETMAIN/FREEMAIN, PGSER or STORAGE macro for virtual storage above the 2-gigabyte address.

The 2-gigabyte address in the address space is marked by a virtual line called the bar. The bar separates storage below the 2-gigabyte address, called below the bar, from storage above the 2-gigabyte address, called above the bar. The area above the bar is intended to be used for data only, not for executing programs. Programs use the IARV64 macro to obtain storage above the bar in “chunks” of virtual storage called memory objects. Your installation can set a limit on the use of the address space above the bar for a single address space. The limit is called the MEMLIMIT.

Managing memory objects

When you create a nonshared memory object you can specify a guard area (not accessible) and a usable area. Later, you can create alternate guard areas or change all or some of a guard area into an accessible area or vice versa. Following are the IARV64 macro keywords for memory objects in 64-bit common:

GETSTOR	Creates a private memory object (only for private memory objects)
CHANGEGUARD	Increases or decrease the amount of usable memory in a memory object (only for private memory objects)
GETSHARED	Creates a shared memory object (only for shared memory objects)
SHAREMEMOBJ	Allows an address space to access shared memory objects (only for shared memory objects)
CHANGEACCESS	Manages the type of access an address space has to the shared virtual storage (only for shared memory objects)
GETCOMMON	Creates a common memory object (only for common memory objects)
DETACH	Deletes memory objects (applies to all memory objects: private, shared, and common)

Changing status of pages within a memory object

The status of memory objects can be changed with the following IARV64 macro, as follows:

PAGEFIX	Fix virtual pages in central storage (only for private and common memory objects)
PAGEUNFIX	Undo a pagefix (only for private and common memory objects)
PROTECT	Make pages within one or more memory objects read-only (only for private and common memory objects)
UNPROTECT	Make pages within one or more memory objects modifiable (only for private and common memory objects)
DISCARDDATA	Discard the data and free the frames (all memory objects)

PAGEOUT	Notify the system that pages will probably not be referenced again soon (all memory objects)
PAGEIN	Notify the system that pages will be referenced soon (all memory objects)
LIST	Request information about memory objects (all memory objects)

4.6 64-bit storage services with z/OS V1R10

As most z/OS systems strive for virtual storage constraint relief (VSCR), many components (such as LE, USS, or XCF) need to move their blocks above the bar. In consequence, every user of 64-bit storage will have to write their own storage manager to hand out smaller pieces of the storage to their application.

4.6.1 IARST64 and IARCP64 services

To help alleviate this transition task, besides IARV64, z/OS V1R10 also provides a set of services that allow components to obtain storage as needed without having to write their own storage manager:

IARST64 This service enables the caller to request private or common storage in sizes from 1 byte to 64 K. IARST64 is equivalent to GETMAIN and FREEMAIN, or STORAGE OBTAIN or RELEASE for 64-bit storage, or GET and FREE 1 to 64K bytes of private or common.

IARCP64 This service enables the caller to create a private or common storage cell pool with cells in sizes from 1 byte to almost half a megabyte. IARCP64 is equivalent to CPOOL for 64 bit storage. Also, for BUILD, GET, FREE, and DELETE of a pool supporting cells from 1 byte to .5 MB of private or common storage.

IARST64 and IARCP64 service characteristics

Both IARST64 and IARCP64 services have the following common characteristics:

- Pool extents are all 1 MB in size.
- GET and FREE requests are branch entered. They run with just registers and only program call when a new pool or new extent is needed.
- They use a register interface and have no parameter list.

New pools and extents are implemented in a way that no actual lock or latch is taken when the function is running. At the last moment, when everything is settled and it is just required to insert the newly defined pool or extent into the existing schemes, a Compare-and-Swap is issued. If it fails, the loser has to clean up and reiterate its process, making another try to acquire the required pool or extent. It is considered that such a “losing” situation should rarely be happening.

IARST64 pools cannot be deleted. EOT will clean up a private pool. Common pools are kept forever. Other characteristics that are shared in common by IARCP64 and IARST64 are:

- No contraction of pools is currently supported in z/OS V1R10.
- Boundaries are forced to quadword, cache line, or page, depending on cell size.
- Trailers are used when they fit, to detect overruns.
- Double free detected and rejected with abend.

- Abend reason codes are documented in the macro prolog and constants are provided in the IAXSERVC macro.
- Common can be key 0-7, Private can be key 0-15.
- Supports task or SRB modes, runs enabled or disabled, and uses any cross-memory mode.

Note: See *z/OS MVS Programming Assembler Services Reference*, SA22-7606 for a complete description of these two new macro services.

4.7 SDUMP memory object dump prioritization

To request an SVC dump, the recovery routine can issue the SDUMPX (or SDUMP) macro. These macros produce an unformatted SVC dump that is written to a SYS1.DUMPnn data set. IBM recommends using SDUMPX over SDUMP, although MVS accepts SDUMP macro invocations.

With z/OS V1R10, SDUMP has been changed in order for RSM™ to support dump prioritization for 64-bit memory objects. Language Environment exploits these memory object dump priorities. Previously, AMODE 64 Language Environment dumps may not include all the information to diagnose a problem. The contributing factors for this situation were caused by the amount of storage that needed to be dumped.

Now, the order and grouping of Language Environment storage within and among multiple memory objects in which RSM presents the memory objects to be dumped is determined.

To address this current limitation, z/OS V1R10 changes the way that AMODE 64 Language Environment allocates memory objects, taking advantage of RSM and dump services mechanisms to prioritize the order of dumping. These services allow an application to:

- Obtain and free memory objects with a user-defined dump priority
- Specify a dump priority for a shared memory segment

This improves the chance that the right information is captured in an AMODE 64 Language Environment dump.

64-bit common storage

64-bit common storage can be dumped by specifying a list of 64-bit address ranges to be included in the dump via the LIST64 keyword. Any 64-bit common memory object can be dumped this way. This is the only way to dump 64-bit common memory objects that specified DUMP=NO on the IARV64 GETCOMMON request.

SDATA=(CSA,SQA)

64-bit common memory objects allocated with a dump attribute of DUMP=SQA are included in an SVC dump when SQA is specified as a SDATA option. 64-bit common memory objects allocated with a dump attribute of DUMP=CSA will be included in a SVC dump when CSA is specified as a SDATA option. 64-bit common storage is dumped after common storage below 2 GB and before private storage.

Memory object services

In z/OS V1R10, the __moservices() function is added to AMODE 64 Language Environment to allow an application to do the following:

- Create a memory object that has a user-specified dump priority and is associated with the current Language Environment enclave
- Free a memory object that had been created using __moservices()
- Specify a shared memory dump priority to be used when allocating shared memory

Applications can hence create memory objects with user-specified characteristics that are treated as part of the enclave:

- Dumped along with those of Language Environment, in priority order
- Propagated on a fork()
- Cleaned up during environment termination

4.7.1 Dump priority of LE memory objects

Language Environment uses the DUMPPRIORITY keyword on IARV64 GETSTOR calls. This allows a priority range from 1 to 99 where 1 is the highest priority level. The information sent to the dump data set consists of the following:

- Stacks and control blocks identified as critical information for diagnosing problems
- Memory objects containing stacks and control blocks assigned a priority of 5
- Memory objects containing heaps assigned a priority of 15

More critical information is written to the dump data set first, helping to ensure that it is available for debugging.

IARV64 REQUEST=GETSTOR syntax

Specify the DUMP=LIKERGN parameter as follows indicating that the 64-bit common memory object will be included in a SVC dump when RGN is specified on SDATA:

```
DUMP=(LIKERGN,DUMPPRIORITY=99)
```

IARV64 REQUEST=SHAREMEMOBJ syntax

SVCDUMPRGN=YES parameter in as follows indicating that an SVC dump will include in its virtual storage capture for the owning address space, the usable area of the memory object whenever SDATA=RGN is specified:

```
SVCDUMPRGN=(YES,DUMPPRIORITY=99)
```

IARV64 REQUEST=LIST syntax

Authorized programs can use the IARV64 REQUEST=LIST service to obtain information about all 64-bit common memory objects allocated in the system, or on specific subsets of these objects with the use of filtering options, such as:

- A specific memory object token
- Using the owner's specific ASID or jobname:

  ```
  OWNERASID=xownerasid
  ```

- Using the ownercom= attribute:

  ```
  OWNERCOM=ALL|HOME|SYSTEM|PRIMARY|BYASID
  ```

- Using a particular dump attribute, such as LIKECSA or LIKESQA:

  ```
  DUMP=ALL|LIKECSA|LIKESQA|LIKERGN|LIKELSQA|NO
  ```

The DUMP=LIKERGN parameter indicates that the 64-bit private or 64-bit shared memory objects that have one of the following options specified will be included in the set of memory objects returned, as follows:

- DUMP=LIKERGN or SVCDUMPRGN=YES attribute specified using ORDER=(ASCENDING | DUMPPRIORITY)
- Or defaulted to on the IARV64 REQUEST=GESTOR
- Or IARV64 REQUEST=SHAREMEMOBJ request

Where:

ORDER= **ASCENDING | DUMPPRIORITY** is an optional keyword input that specifies the order in which the memory objects matching the selection criteria on LIST request will be returned. DEFAULT: ASCENDING

ASCENDING - The memory objects that match the selection criteria will be returned in ascending start address order.

D**UMPPRIORITY** - The memory objects that match the selection criteria will be returned in dump priority order where memory objects with higher priority are listed before memory objects with lower priority. Within a dump priority level, memory objects will be listed based on ascending start address.

Note:
ORDER=DUMPPRIORITY cannot be specified with V64SHARED=YES.
ORDER=DUMPPRIORITY cannot be specified with V64COMMON=YES.
ORDER=DUMPPRIORITY also cannot be specified when SVCDUMPRGN=NO is specified.

4.7.2 C language programs and dump prioritization

Invoking dump prioritization functions from a C language program can be done in the following way, with the service call shown in Figure 4-12 and a coding example shown in Figure 4-13 on page 124. Exploiters of SDUMP memory object prioritization are shown in the examples. All AMODE 64 Language Environment applications and middleware will benefit from this support; and more generally, any application that exploits the support.

```
#include <stdlib.h>
int __moservices(int reqtype, size_t mopllen,
         struct __mopl_s * mopl, void ** moorigin);
```

Figure 4-12 C language program service call example

Where:

reqtype __MO_GETSTOR: Use IARV64 REQUEST(GETSTOR) to create a memory object.

__MO_DETACH: Use IARV64 REQUEST(DETACH) to free a memory object.

__MO_SHMDUMPPRIORITY: Set a shared memory dump priority.

mopllen The length of the passed mopl structure.

mopl Points to a structure describing additional characteristics for the request.

moorigin Contains the origin address of the memory object that was obtained or to be attached.

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>

int main(void)  {
    __mopl_t mymopl;
    int  mos_rv;
    void * mymoptr;

    memset(&mymopl, 0, sizeof(__mopl_t));

    /* Set a shared memory dump priority for subsequent shmget() */
    mymopl.__mopldumppriority = 8;

    mos_rv = __moservices(__MO_SHMDUMPPRIORITY, sizeof(mymopl),
                          &mymopl, &mymoptr);
if (mos_rv != 0) {
      perror("moservices(SHMDUMPPRIORITY) call failed");
    }
```

Figure 4-13 C language program invoking dump prioritization example

C language program examples

The input mopl must be cleared to binary zeroes when initializing.

For a __moservices() __MO_SHMDUMPPRIORITY call, the __mopldumppriority field needs to be set in the input mopl.

For a __moservices() __MOGETSTOR call, a dump priority between 1-99 must be specified, as well as the size of the memory object, in MB, to be obtained, as shown in Figure 4-14.

```
/* Obtain a 100MB memory object whose dump priority falls  */
/* between the dump priorities of the Language Environment */
/* stacks and heaps.                                       */
    mymopl.__mopldumppriority = __MO_DUMP_PRIORITY_STACK + 5;
    mymopl.__moplrequestsize = 100;

    mos_rv = __moservices(__MO_GETSTOR, sizeof(mymopl),
                          &mymopl, &mymoptr);

    if (mos_rv == 0) {
      printf("moservices(GETSTOR) successful, MO addr: %p\n",
             mymoptr);
```

Figure 4-14 Example to obtain a 100 MB memory object

Figure 4-15 shows an example to free the memory object. For a __moservices() __MO_DETACH call, the origin address of the memory object to be detached must be provided.

```
/* Free the 100MB memory object.                        */
      mos_rv = __moservices(__MO_DETACH, 0, NULL, &mymoptr);

      if (mos_rv != 0) {
        perror("moservices(DETACH) call failed");
      }
    }
    else {
      perror("moservices(GETSTOR) call failed");
    }

    return 0;
}
```

Figure 4-15 Example to free the memory object

4.8 64-bit common services aids

Besides the new SDUMP memory object prioritization described previously, other aids for the 64-bit common environment have been added in z/OS V1R10.

4.8.1 Dumping 64-bit common areas

64-bit common storage can be dumped in the following ways:

- Specifying a list of 64-bit address ranges to be included in the dump via the LIST64 keyword.
  - Any 64-bit common memory object can be dumped this way.
  - This is the only way to dump 64-bit common memory objects that specified DUMP=NO on the IARV64 GETCOMMON request.
- SDATA=(CSA,SQA)
  - 64-bit common memory objects allocated with a dump attribute of DUMP=SQA will be included in an SVC dump when SQA is specified on SDATA.
  - 64-bit common memory objects allocated with a dump attribute of DUMP=CSA will be included in an SVC dump when CSA is specified on SDATA.
- 64-bit common storage will be dumped after common storage below 2 G and before private storage.

4.8.2 Obtaining information about the use of 64-bit common storage

Authorized programs can use the IARV64 REQUEST=LIST, described in “IARV64 REQUEST=LIST syntax” on page 122, to obtain information about all 64-bit common memory objects allocated in the system, or on specific subsets of these objects, by filtering on the keywords.

In addition to the list described in “IARV64 REQUEST=LIST syntax” on page 122, use the following IARV64 keywords:

V64COMMON= **NO | YES** is an optional keyword input that specifies whether the list of memory objects returned is for the current primary address space, or a list of all 64-bit common memory objects allocated in the system via IARV64 REQUEST=GETCOMMON. DEFAULT: NO

NO - When V64SHARED=NO is also specified the list of memory objects returned for the current primary address space includes private memory objects (which are defined for the private area via an IARV64 REQUEST=GETSTOR), and shared memory objects (connected to the current primary space via an IARV64 REQUEST=SHAREMEMOBJ).

YES - The list of memory objects returned contains all 64-bit common memory objects defined in the system via IARV64 REQUEST=GETCOMMON.

V64SELECT= **NO | YES** is an optional keyword input that specifies whether the list request is for all allocated memory objects or for a subset of the allocated memory objects.DEFAULT: NO

4.9 Large page support

Memory architecture is extended to support large (1 MB) pages. When large pages are used in addition to the existing 4 KB page size, they are expected to reduce memory management overhead for exploiting applications.

With the z10 EC models, page frames can be allocated with a 4 KB size. The z10 EC additionally supports a larger page frame size of 1 MB. It is expected that long running, memory access intensive applications will benefit from large page frames. It should be noted that large pages are treated as fixed pages and are never paged out.

As hardware usage has shown in recent years, the Translation Lookaside Buffer (TLB) coverage has been shrinking as % of memory size, as follows:

- Over the past few years application memory sizes have dramatically increased due to support for 64-bit addressing in both physical and virtual memory.
- TLB sizes have remained relatively small due to low access time requirements and hardware space limitations.
- TLB coverage today represents a much smaller fraction of an application’s working set size, leading to a larger number of TLB misses.
- Applications can suffer a significant performance penalty resulting from an increased number of TLB misses as well as the increased cost of each TLB miss.

The solution introduced with z10 EC is to increase TLB coverage without proportionally enlarging the TLB size by using large pages, as follows:

- Large pages allow for a single TLB entry to fulfill many more address translations.
- Large pages will provide exploiters with better TLB coverage.

The overall benefit is that the z10 EC is designed for better performance by decreasing the number of TLB misses that an application incurs.

Parameters or messages, issued from the Real Storage Manager (RSM), use the term large frame (LF), which speaks to the hardware reality of the memory.

Large page performance considerations

More thorough examination is required to adequately exploit large pages, as follows:

- Large page is a special purpose performance improvement feature. It is not recommended for general use. Large page usage provides performance value to a select set of applications:

  These are primarily long running memory access intensive applications, for which locality of reference does not hold in the long term.

- Not all applications benefit from using large pages. Some applications can be severely degraded by the use of large pages:

  Short lived processes with small working sets are usually not good candidates for large pages

- Factors to consider when trying to either estimate the potential benefit or understand measured performance differences of using larger pages instead of 4 K pages include:
  - Memory usage, and more precisely its pattern of references
  - A workload's page translation overhead

Large page potential exploiters

A future release of DB2 will support large pages for buffer pools, and Java 6.0 SR1 for z/OS is planned to support large pages. Large pages can be used to back the object heap.

4.9.1 Possible large page usage in z/OS V1R10

Because of the previous considerations, large page usage is restricted in z/OS V1R10 to 64-bit common. See "Memory objects with IARV64" on page 119 for detailed information.

With z/OS V1R10, only the following applies to application use of large pages:

- Must run on a z10 EC.
- Must issue the IARV64 macro.
- Must not be dependent upon LE environment.
- Must explicitly run in 64-bit mode.
- Must have no restriction on giving total visibility to their data from any other address space that can benefit from this large page support.

4.9.2 64-bit common exploiters dependencies

64-bit common is introduced in z/OS V1R10 in order to relieve a prevalent virtual storage constraint for common areas; refer to Figure 4-16 on page 128. Exploiters that start to exploit this new functionality are:

- I/O supervisor, which moves some of its UCBX data into 64-bit common.
- Communication storage manager (CSM) on behalf of IP Services.
  - Starting with z/OS V1R10, the valid range for the IVTPRMxx parmlib member can be from 1024K to 30720M.

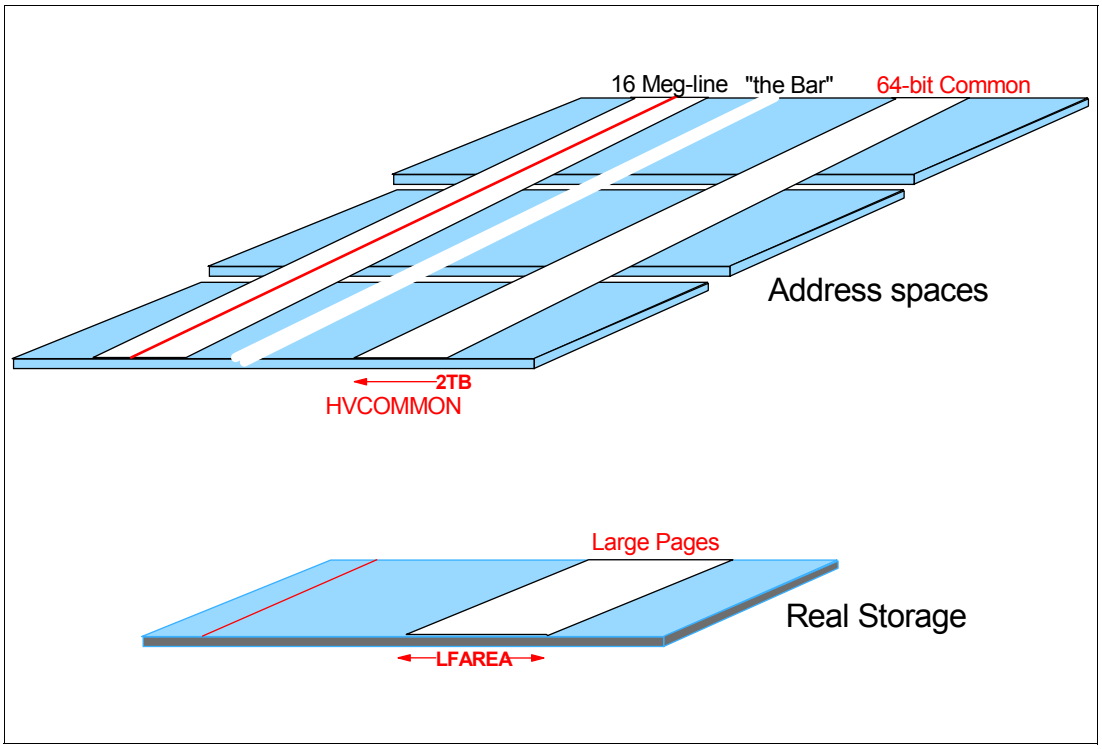


Figure 4-16 64-bit common customization

Note: Large page (1 MB) support is not supported by z/OS when z/OS runs as a guest on z/VM on a System z10 server.

Large page support

The large page support function is not enabled without the required software support. Without the large page support, page frames are allocated at the current 4 K size.

Memory reserved for large page support can be defined with a new parameter in the IEASYSxx parmlib member:

```
LFAREA=xx% | xxxxxxM | xxxxxxG
```

This parameter specifies the amount of real storage to be made available for 1 MB pages, either in absolute value or as a percentage of real storage, as follows:

xx% Indicates that the amount of real storage to be used for large pages is specified as a percentage of all online real storage available at IPL time. The maximum amount of real storage that can be used to back large pages is 80% of the online storage available at IPL minus 2 GB.

If the amount specified exceeds this value, the specification is rejected and the system issues a prompt for a new value to be specified on the LFAREA parameter.

The maximum amount of real storage that can be used to back large pages is 80% of the online storage available at IPL minus 2 GB. The formula to calculate the maximum amount of real storage that can be used to back large pages is below:

xxxxxxG Indicates the amount of online real storage available at IPL time in gigabytes that is to be used for large pages. The maximum amount of real storage that

can be used to back large pages is 80% of the online storage available at IPL minus 2 GB. If the amount specified exceeds this value, the specification is rejected and the system issues a prompt for a new value to be specified on the LFAREA parameter.

For calculations in gigabytes, xxxxxx = online real storage at IPL in GB. MAX_LFAREA = (x - 2G) * .8

xxxxxxM Indicates the amount of online real storage at IPL time in megabytes that is to be used to back large pages. The maximum amount of real storage that can be used to back large pages is 80% of the online storage available at IPL minus 2 GB. If the amount specified exceeds this value, the specification is rejected and the system issues a prompt for a new value to be specified on the LFAREA parameter.

For calculations in megabytes, x = online real storage at IPL in MB. MAX_LFAREA = (y - 2048M) * .8

Important: Default Value is none (No LFAREA is defined). This parameter cannot be changed dynamically during an IPL.

4.10 RMF support for 64-bit common

The new Monitor III Storage Memory Objects report (STORM) contains measurements of 64-bit virtual common storage consumption per system and per job. Measurements of the use of memory objects previously were contained in the following reports:

- Monitor III Storage Resource Delays (STORR).
- Storage Delay Summary (STORS).
- Storage Frames (STORF) reports are moved to this new report.

Also, the following postprocessor reports are enhanced to provide 64-bit virtual common storage usage information:

- In the Frame and Slot Counts section of the Paging Activity report, a new Memory Objects and Frames category is added.
- In the Private Area Detail section of the Virtual Storage Activity report, the section containing memory allocation values above 2 GB is enhanced with new information about memory objects and 1 MB frames.

In addition, RMF provides new overview conditions for the postprocessor based on SMF record 71. RMF provides support for 64-bit common memory in the RMF postprocessor and Monitor III on a per address space and system basis:

- By collecting additional data in SMF record 71
- By collecting additional data in SMF record 78 subtype 2
- By extending the RMF Postprocessor Paging Activity Report
- By presenting system-wide and address space-related information in the new RMF Monitor III Storage Memory Objects report

4.10.1 RMF Postprocessor report

The Frame and Slot Counts Section of the Paging Activity report provides information about total and fixed frames in central storage, local page data set slot counts, and shared frames and slots. All items are shown as minimum, average and maximum values.

The number of 64-bit common memory auxiliary storage slots is not shown in the report, but provided in the SMF71 record.

4.10.2 New Monitor III Storage Memory Objects report (STORM)

All memory objects data is now concentrated in the new job oriented Monitor III Storage Memory Objects report (STORM). The large page support fields formerly reported in the STORR and STORS report headers and in the STORF reports are moved into the new report.

The number of 64-bit common memory auxiliary storage slots is not shown in the reports, but is accessible via the RMF utility for Monitor III reports. If enhanced DAT architecture is not supported, the large page support items are not shown.

New system-wide metrics in RMF

64-bit common support introduces new system-wide metrics in the RMF Postprocessor Paging Activity report and the new RMF Monitor III Storage Memory Objects report system summary section:

- Number of 64-bit common memory objects allocated in the system
- Number of 64-bit common memory frames backed in real storage
- Number of 64-bit common memory frames fixed in real storage
- Number of 64-bit common memory auxiliary storage slots
- Percentage of 64-bit common frames backed in real storage of high common area size (MIII)

Additionally, the following metrics are presented:

- Number of shared memory objects allocated in the system
- Number of high virtual shared memory frames backed in real storage
- Percentage of shared frames backed in real storage of shared area size (MIII)
- Percentage of 1-MB frames backed in real storage of large frames area size (MIII)

The following 64-bit Common Support address space-related metrics are supported by the Postprocessor Virtual Storage Report and the new RMF MIII Storage Memory Objects report:

- 64-bit common memory objects allocated with this address space as the owner
- Amount of 64-bit common storage allocated
- High-water mark for the amount of 64-bit common storage allocated

Additionally, the following metrics are presented:

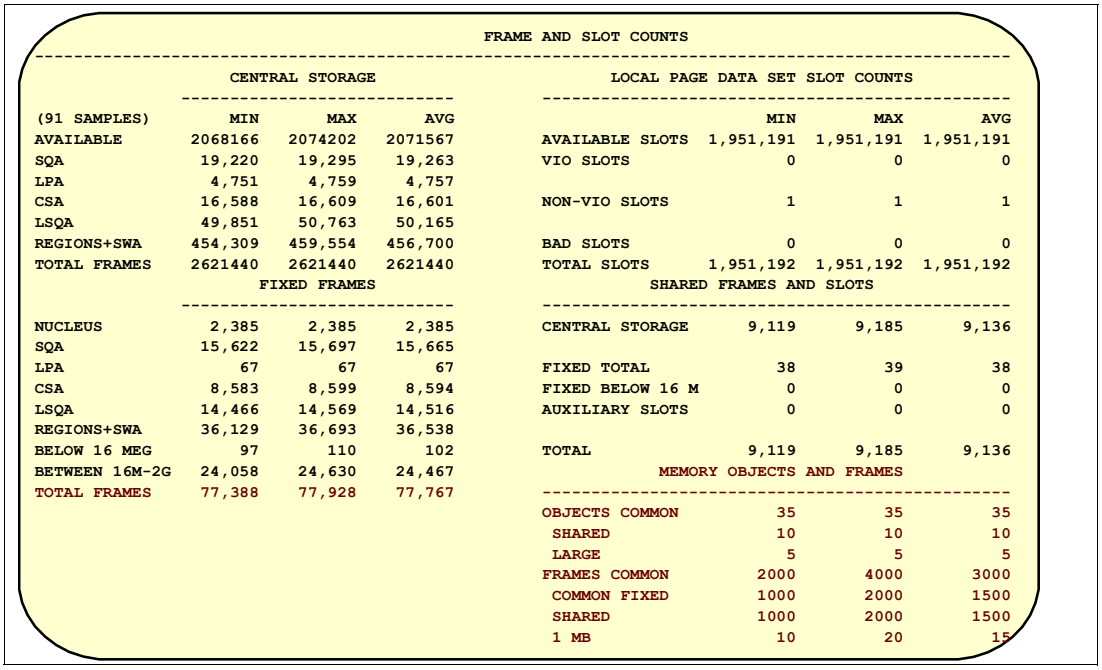
- Number of memory objects allocated with this address space as the owner
- Number of shared memory objects allocated
- Amount of storage allocated from large virtual memory in memory objects
- Amount of shared storage from large virtual memory in memory objects

Frames and Slots Counts section

The Frames and Slots Counts section of the RMF Postprocessor Paging Activity report is enhanced, as shown in Figure 4-17.

The memory objects category is broken down into the following fields:

- OBJECTS that include the number of 64-bit COMMON memory objects, SHARED and LARGE memory objects (large page support).
- FRAMES, which include the number of 64-bit COMMON memory frames backed in real storage, 64-bit COMMON memory frames FIXED in real storage, high virtual SHARED memory frames backed in real storage, and 1 MB frames backed in real storage (large page support).



FRAME AND SLOT COUNTS

CENTRAL STORAGE			
(91 SAMPLES)	MIN	MAX	AVG
AVAILABLE	2068166	2074202	2071567
SQA	19,220	19,295	19,263
LPA	4,751	4,759	4,757
CSA	16,588	16,609	16,601
LSQA	49,851	50,763	50,165
REGIONS+SWA	454,309	459,554	456,700
TOTAL FRAMES	2621440	2621440	2621440

FIXED FRAMES			
NUCLEUS	2,385	2,385	2,385
SQA	15,622	15,697	15,665
LPA	67	67	67
CSA	8,583	8,599	8,594
LSQA	14,466	14,569	14,516
REGIONS+SWA	36,129	36,693	36,538
BELOW 16 MEG	97	110	102
BETWEEN 16M-2G	24,058	24,630	24,467
TOTAL FRAMES	77,388	77,928	77,767

LOCAL PAGE DATA SET SLOT COUNTS			
	MIN	MAX	AVG
AVAILABLE SLOTS	1,951,191	1,951,191	1,951,191
VIO SLOTS	0	0	0
NON-VIO SLOTS	1	1	1
BAD SLOTS	0	0	0
TOTAL SLOTS	1,951,192	1,951,192	1,951,192

SHARED FRAMES AND SLOTS			
CENTRAL STORAGE	9,119	9,185	9,136
FIXED TOTAL	38	39	38
FIXED BELOW 16 M	0	0	0
AUXILIARY SLOTS	0	0	0
TOTAL	9,119	9,185	9,136

MEMORY OBJECTS AND FRAMES			
OBJECTS COMMON	35	35	35
SHARED	10	10	10
LARGE	5	5	5
FRAMES COMMON	2000	4000	3000
COMMON FIXED	1000	2000	1500
SHARED	1000	2000	1500
1 MB	10	20	15

Figure 4-17 Overview of Postprocessor Paging Activity Report

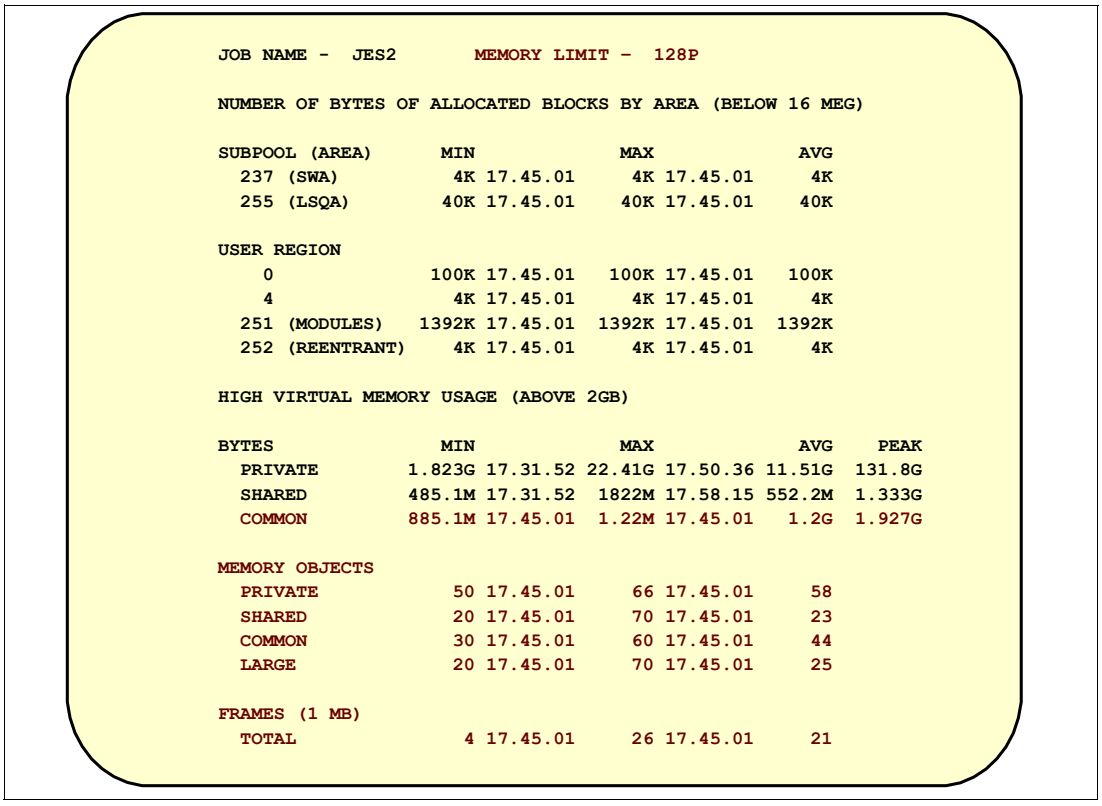
POSTPROCESSOR VSTOR report

The HIGH VIRTUAL MEMORY USAGE (ABOVE 2 GB) part of the RMF Postprocessor VSTOR report. The Private Area detail section is enhanced, as follows:

- 64-bit common byte counts are added to the existing BYTE block.
- A new MEMORY OBJECTS block is added.
- Number of 64-bit private memory objects, 64-bit common memory objects, SHARED and LARGE memory objects (large page support)
- A new FRAMES block is added
- 1-MB frames backed in real storage (large page support)

For all fields, the minimum, maximum, and average numbers are shown.

The large page support fields (large memory objects and frame counts) are shown only if Enhanced DAT Architecture is supported, otherwise the lines are not displayed.



```
JOB NAME -  JES2       MEMORY LIMIT -  128P

NUMBER OF BYTES OF ALLOCATED BLOCKS BY AREA (BELOW 16 MEG)

SUBPOOL (AREA)      MIN             MAX             AVG
  237 (SWA)          4K 17.45.01     4K 17.45.01     4K
  255 (LSQA)        40K 17.45.01    40K 17.45.01    40K

USER REGION
    0              100K 17.45.01   100K 17.45.01   100K
    4                4K 17.45.01     4K 17.45.01     4K
  251 (MODULES)   1392K 17.45.01  1392K 17.45.01  1392K
  252 (REENTRANT)    4K 17.45.01     4K 17.45.01     4K

HIGH VIRTUAL MEMORY USAGE (ABOVE 2GB)

BYTES               MIN             MAX             AVG    PEAK
  PRIVATE        1.823G 17.31.52 22.41G 17.50.36 11.51G  131.8G
  SHARED         485.1M 17.31.52  1822M 17.58.15 552.2M  1.333G
  COMMON         885.1M 17.45.01  1.22M 17.45.01   1.2G  1.927G

MEMORY OBJECTS
  PRIVATE             50 17.45.01     66 17.45.01     58
  SHARED              20 17.45.01     70 17.45.01     23
  COMMON              30 17.45.01     60 17.45.01     44
  LARGE               20 17.45.01     70 17.45.01     25

FRAMES (1 MB)
  TOTAL                4 17.45.01     26 17.45.01     21
```

Figure 4-18 Overview of Postprocessor VSTOR Report

4.11 SMF support for 64-bit common storage

Figure 4-19, Figure 4-20 on page 133, and Figure 4-21 on page 133 display the fields that are added to SMF record type 71.

Offsets	Name	Len	Format	Description
1164 x48C	SMF71COM	8	float	Min.number of 64-bit common memory objects allocated in the system
1172 x494	SMF71COX	8	float	Max. number of 64-bit common memory objects allocated in the system
1180 x49C	SMF71COA	8	float	Average number 64-bit common memory objects allocated in the system
1188 x4A4	SMF71CRM	8	float	Minimum number 64-bit common memory frames backed in real storage
1196 x4AC	SMF71CRX	8	float	Maximum number 64-bit common memory frames backed in real storage
1204 x4B4	SMF71CRA	8	float	Average number 64-bit common memory frames backed in real storage
1212 x4BC	SMF71CFM	8	float	Minimum number of 64-bit common memory frames fixed in real storage
1220 x4C4	SMF71CFX	8	float	Maximum number of 64-bit common memory frames fixed in real storage
1228 x4CC	SMF71CFA	8	float	Average number of 64-bit common memory frames fixed in real storage
1236 x4D4	SMF71CSM	8	float	Minimum number of 64-bit common memory auxiliary storage slots
1244 x4DC	SMF71CSM	8	float	Maximum number of 64-bit common memory auxiliary storage slots
1252 x4E4	SMF71CSA	8	float	Average number of 64-bit common memory auxiliary storage slots

Figure 4-19 Paging Activity Paging Data Section of SMF Record 71

Offsets	Name	Len	Format	Description
1260 x4EC	SMF71COM	8	float	Min number of shared memory objects allocated in the system
1268 x504	SMF71COX	8	float	Max number of shared memory objects allocated in the system
1276 x50C	SMF71COA	8	float	Average number shared memory objects allocated in the system
1284 x514	SMF71CRM	8	float	Minimum number of high virtual shared memory frames backed in real storage
1292 x51C	SMF71CRX	8	float	Maximum number of high virtual shared memory frames backed in real storage
1300 x524	SMF71CRA	8	float	Average number of high virtual shared memory frames backed in real storage

Figure 4-20 Paging Activity Paging Data Section of SMF Record 71

Condition	Name	Source
Average number of 64-bit common memory objects allocated in the system	CMOA	SMF71COA
Average number of 64-bit common memory frames backed in real storage	CFRA	SMF71CRA
Average number of 64-bit common memory frames fixed in real storage	CFFRA	SMF71CFA
Average number of 64-bit common memory auxiliary storage slots	CAUXSA	SMF71CAA
Average number of shared memory objects allocated in the system	SMOA	SMF71SOA
Average number of high virtual shared memory frames backed in real storage	SFRA	SMF71SRA

Figure 4-21 New overview/ and xception conditions are available in SMF type 71

SMF Record 78 Subtype 2

SMF record type 82 has fields related to the use of memory objects that are new with z/OS V1R10.

Offset	Name	Length	Format	Description
440 1B8	R782SHBY	48	mixed	Number of shared bytes allocated in storage above the 2-GB-line.
488 1E8	**R782COBY**	**48**	**mixed**	**Number of 64-bit common bytes allocated in storage above the 2-GB-line.**
536 218	**R782TOMO**	**40**	**mixed2**	**Total number of 64-bit private memory objects allocated.**
576 240	**R782SHMO**	**40**	**mixed2**	**Number of shared memory objects.**
616 268	**R782COMO**	**40**	**mixed2**	**Number of 64-bit common memory objects.**
656 290	**R782LGMO**	**40**	**mixed2**	**Number of large memory objects**
696 2B8	**R782TOFR**	**40**	**mixed2**	**Number of 1 MB frames.**
736 2E0	**R782MEML**	**8**	**binary**	**Address space memory limit in MB**

Figure 4-22 SMF Record 78 Subtype 2

Tables 1 and 2 are shown in Figure 4-23 and Figure 4-23, respectively.

- Table 1shows a layout for the following SMF78-2 record fields (format mixed): R782TOHBY, R782SHBY, and R782COBY.

Offset		Name	Length	Format	Description
0	0	VSDGMIN	8	floating	Minimum value
8	8	VSDGNTME	4	binary	Time stamp for minimum value.
12	C		4		reserved
16	10	VSDGMAX	8	floating	Maximum value
24	18	VSDGXTME	4	binary	Time stamp for maximum value.
28	1C		4		reserved
32	20	VSDGTOTL	8	floating	Total all samples (used to calculate the average).
40	28	VSDGHWM	8	floating	High watermark

Figure 4-23 SMF record type 78 Subtype 2 Table 1

- Table 2 shows a layout for the following SMF78-2 record fields (format mixed2): R782TOMO, R782SHMO, R782COMO, R782COMO, and R782TOFR.

Offset		Name	Length	Format	Description
0	0	VSDCMIN	8	floating	Minimum number of memory objects / frames
8	8	VSDCNTME	4	binary	Time stamp for minimum value.
•	C		4		reserved
16	10	VSDCMAX	8	floating	Maximum number of memory objects / frames
24	18	VSDCXTME	4	binary	Time stamp for maximum value.
28	1C		4		reserved
32	20	VSDCTOTL	8	floating	Total all samples (used to calculate the average).

Figure 4-24 SMF record type 78 Subtype 2 Table 2

4.11.1 Dumping for 64-bit common storage

With the 64-bit addressing scheme, four different service requests have been introduced:

- IARV64 Request(GETSTOR) DUMP(LIKELSQA) specifies that the 64-bit memory object will be included in the SDUMP when LSQA is specified on SDATA.
- IARV64 Request(GETSTOR) DUMP(LIKERGN) specifies that the 64-bit memory object will be included in the SDUMP when RGN is specified on SDATA.
- IARV64 Request(GETCOMMON) DUMP(LIKESQA) specifies that the 64-bit common memory object will be included in a SDUMP when SQA is specified on SDATA.
- IARV64 Request(GETCOMMON) DUMP(LIKECSA) specifies that the 64-bit common memory object will be included in a SDUMP when CSA is specified on SDATA.

These service requests indicate that the following information is placed into the dump:

- SDUMPX SDATA(SQA) will cause storage obtained with DUMP(LIKESQA) to be dumped as well.
- SDUMPX SDATA(CSA) will cause storage obtained with DUMP(LIKECSA) to be dumped as well.

- SDUMPX SDATA(LSQA) will cause storage obtained with DUMP(LIKELSQA) to be dumped as well.
- SDUMPX SDATA(RGN) will cause storage obtained with DUMP(LIKERGN) to be dumped as well.

4.11.2 Order of storage dumping

The order of the storage being dumped is as follows:

1. Global storage
   - Global storage below 2 G
   - Global storage above 2 G
     - If SDATA=SQA is specified, storage is obtained with DUMP(LIKESQA).
     - If SDATA=CSA is specified, storage obtained with DUMP(LIKECSA).
2. Private storage (for each of the address spaces specified)
   - Private storage below 2 G
   - Private storage above 2 G
     - If SDATA=LSQA is specified, storage obtained with DUMP(LIKELSQA).
     - Storage specified in LIST64 (this may include shared or common storage which was not already dumped together with the global storage).
     - If SDATA=RGN is specified, the rest of storage obtained with DUMP(LIKERGN).

5

InfiniBand support

InfiniBand® is a powerful interconnect technology designed to deliver I/O connectivity for large server and network infrastructures. It is supported by all major server vendors as a means to deliver the next generation I/O interconnect standard for servers.

InfiniBand is an industry-standard specification that defines an input/output architecture used to interconnect servers, communications infrastructure equipment, storage, and embedded systems. InfiniBand is a true fabric architecture that leverages switched, point-to-point channels with high bandwidth data transfers, both in chassis backplane applications as well as through external copper and optical fiber connections. InfiniBand is a pervasive, low-latency, high-bandwidth interconnect which requires low processing overhead and is ideal for carrying multiple traffic types (clustering, communications, storage, management) over a single connection.

This chapter describes the following features of InfiniBand:

- InfiniBand background and capabilities
- Benefits of using InfiniBand
- Different InfiniBand usage in z10 EC implementation
- I/O connectivity options related to InfiniBand
- Defining InfiniBand coupling links in the HCD/IOCP
- PSIFB link support
- Sample configuration with PSIFB links

5.1 InfiniBand background and capabilities

System z servers are able to do more and more work. Thus, an interconnect architecture is needed that is able to scale with the increasing performance of the platform to satisfy the I/O interconnect requirements that go along with it. InfiniBand is such a powerful interconnect architecture, and is designed to scale with increasing processor speeds.

In 1999, two competing input/output (I/O) standards called Future I/O (developed by Compaq, IBM and Hewlett-Packard) and Next Generation I/O (developed by Intel®, Microsoft® and Sun) merged into a unified I/O standard called InfiniBand. The InfiniBand Trade Association® (IBTA) organization maintains the InfiniBand specification. The IBTA is led by a steering committee comprised of members of these corporations. For more information about this topic, refer to:

http://www.infinibandta.org/home

The goal of System z was to introduce an industry-standard high speed host bus physical interface to replace the Self-Timed Interconnect (STI) proprietary host bus interface. When used in a data center, InfiniBand coupling links are designed to complement but may not replace Integrated Cluster Bus-4 (ICB-4) and InterSystem Channel-3 (ISC-3). For example, InfiniBand coupling may be used for consolidating multiple ISC-3 links within a data center at distances up to 150 meters (compared to 10 meters).

The InfiniBand host bus physical interface supports 12x Double Data Rate (12x IB-DDR) with a link speed of 6 gigabytes per second (GBps) when attached to a z10 EC. It supports 12x Single Data Rate (12x IB-SDR) with a link speed of 3 Gbps when a z10 EC is attached to a z9 EC or z9 BC.

Server Time Protocol (STP) will use InfiniBand Coupling links to exchange timekeeping messages required for time synchronization.

5.2 Benefits of using InfiniBand

InfiniBand offers the following benefits:

- Superior performance

  InfiniBand provides superior latency performance.

- Reduced complexity

  InfiniBand allows for the consolidation of multiple I/Os on a single cable or backplane interconnect, which is critical for blade servers, data center computers and storage clusters, and embedded systems.

- Highest interconnect efficiency

  InfiniBand was developed to provide efficient scalability of multiple systems. InfiniBand provides communication processing functions in hardware, thus relieving the CPU of this task, and it enables the full resource utilization of each node added to the cluster.

- Reliable and stable connections

  InfiniBand provides reliable end-to-end data connections and defines this capability to be implemented in hardware.

Figure 5-1 on page 139 shows the InfiniBand communications stack. Several types of architected transactions can be used to execute a transaction with another user. Work is posted on the appropriate queue and the channel adapter executes the operation.

In case of a send operation, the channel adapter interprets the type of work, creates a message, segments it (if needed) into multiple packets, adds the routing information, and sends the packets to a port. Port logic is responsible for sending the packets across the link through the fabric to its destination. When the packets arrive, the port logic validates them and the channel adapter puts them on the queue and executes. If requested, the channel adapter creates an acknowledgement and sends it back to its origin.

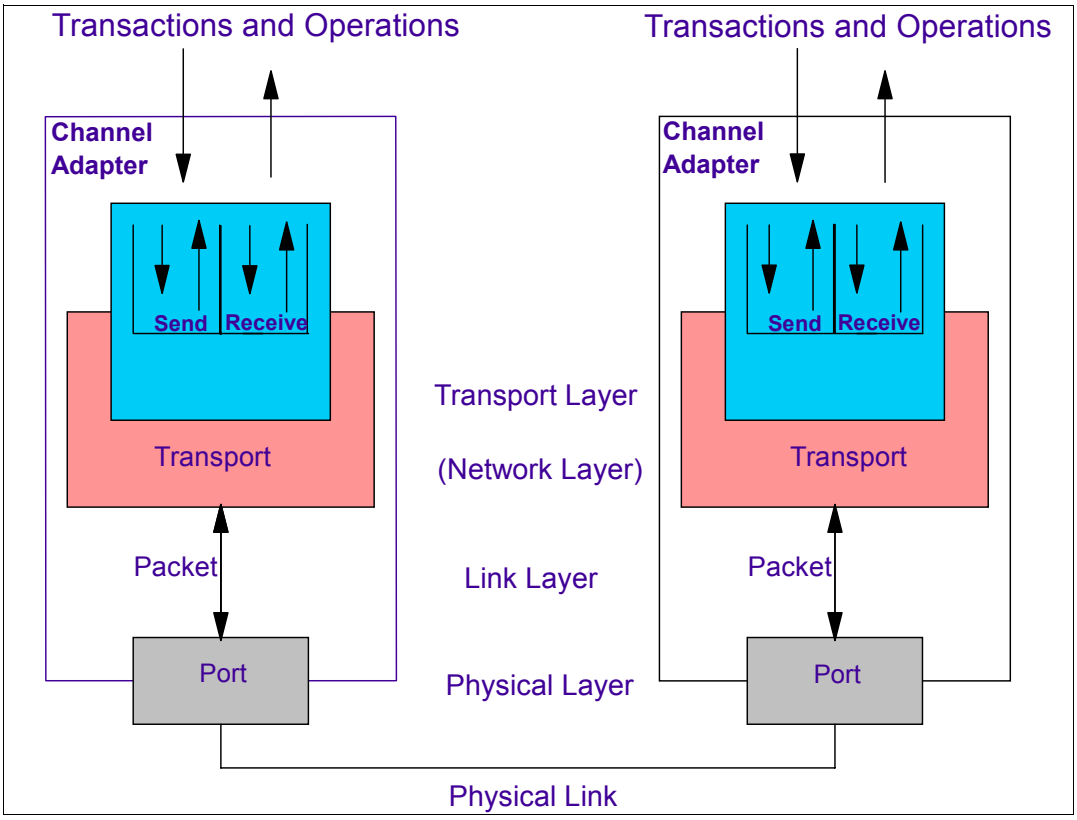


Figure 5-1 Communication stack

5.3 Different InfiniBand usage in z10 EC implementation

As shown in Figure 5-2 on page 140, InfiniBand technology is exploited in different areas in the System z9 and z10 EC, for the following reasons:

- I/O cage intra-connection (in lieu of the Self-Timed Interconnect Multiplexer (STI-MP) of the z9 system)
- Coupling link (in lieu of the CF links)

Only the coupling link usage is visible to z/OS, so it must be customized.

Type	System z9	z10 EC
MBA fanout	I/O cage connection to processor nest or Coupling (ICB-4)	ICB-4
HCA1-O fanout	Optical - Coupling 12x IB-SDR	NA
HCA2-C fanout	NA	Copper - I/O Cage - 12x IB-DDR
HCA2-O fanout	NA	Optical - Coupling 12x IB-DDR
STI-MP	I/O cage intra-connection	NA
IFB-MP	NA	I/O cage intra-connection

Acronym	Full name	Comments
MBA	Memory Bus Adapter	Path for communication
HCA	Host Channel Adapter	Path for communication
STI-MP	Self-Timed Interconnect Multiplexer	I/O cage intra-connection
IFB-MP	InfiniBand Multiplexer	I/O cage intra-connection
CIB	Coupling using InfiniBand	CHPID type z10 EC, System z9
PSIFB	Parallel Sysplex using InfiniBand	Coupling Link using InfiniBand

Figure 5-2 z10 EC I/O connectivity

5.3.1 z I/O connectivity evolution

The introduction of the InfiniBand technology in the z systems can be illustrated as in Figure 5-3 on page 140.

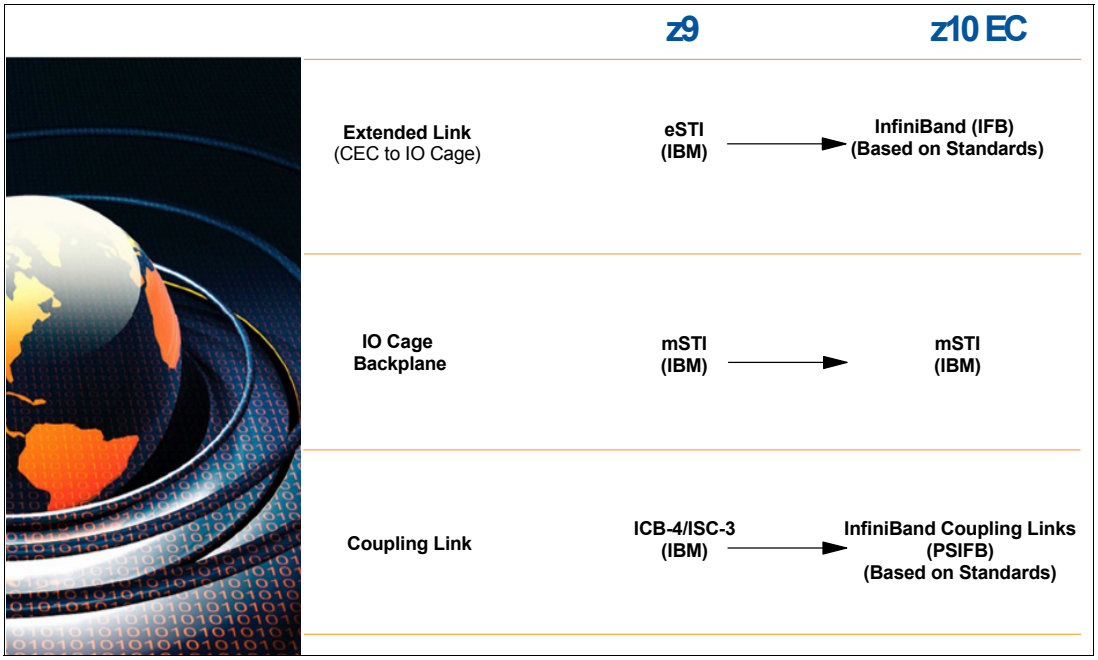


Figure 5-3 z I/O interface evolution

5.3.2 Connectivity for coupling

z10 EC allows a combination of up to eight InfiniBand Host Channel Adapter (HCA2-Optical or HCA2-Copper) and Memory Bus Adapter (MBA) fanout cards. Each one has two ports

supporting up to 16 connections. HCA2-Copper connections are for links to I/O cages in this server, and HCA2-Optical and MBA are to external servers (coupling links). MBA cards are used for ICB-4 links only.

As I/O cards continue to support higher data transfer rates to the devices, the connection between the I/O cards and the CEC cage needs to provide a higher data rate as well. The connectivity to the I/O cages (I/O domains) in the System z10 is implemented by InfiniBand technology.

In the z10 EC I/O cage, the InfiniBand Multiplexer (IFB-MP) card replaces the Self-Timed Interconnect Multiplexer card present in z9. The z10 EC supports a combination of old technology Memory Bus Adapter (MBA) and Host Channel Adapter (HCA) InfiniBand fanout cards on the CEC cage. The MBA fanout cards are used exclusively for ICB-4 coupling facility links.

With the System z10 there are two types of HCAs in one book:

- The HCA2-C fanout connects via an IFB copper cable to an IFB-MP (InfiniBand - Multiplex) card installed in the I/O cages.
- The HCA2-O fanout connects via a CF link optical cable (external) with another z10 EC server.

Figure 5-4 on page 142 shows the following connections:

- HCA2-C adapters connected through copper cables with IFB-MP located in one I/O slot in the I/O cage. This IFB-MP handles and manages the flow of four I/O cards constituting a domain.
- HCA2-O adapters connected through optic fiber cables named Parallel Sysplex using InfiniBand (PSIFB) with HCA2-O located in another z10 EC server.
- MBA adapters connected through copper cables with an ICB-4.

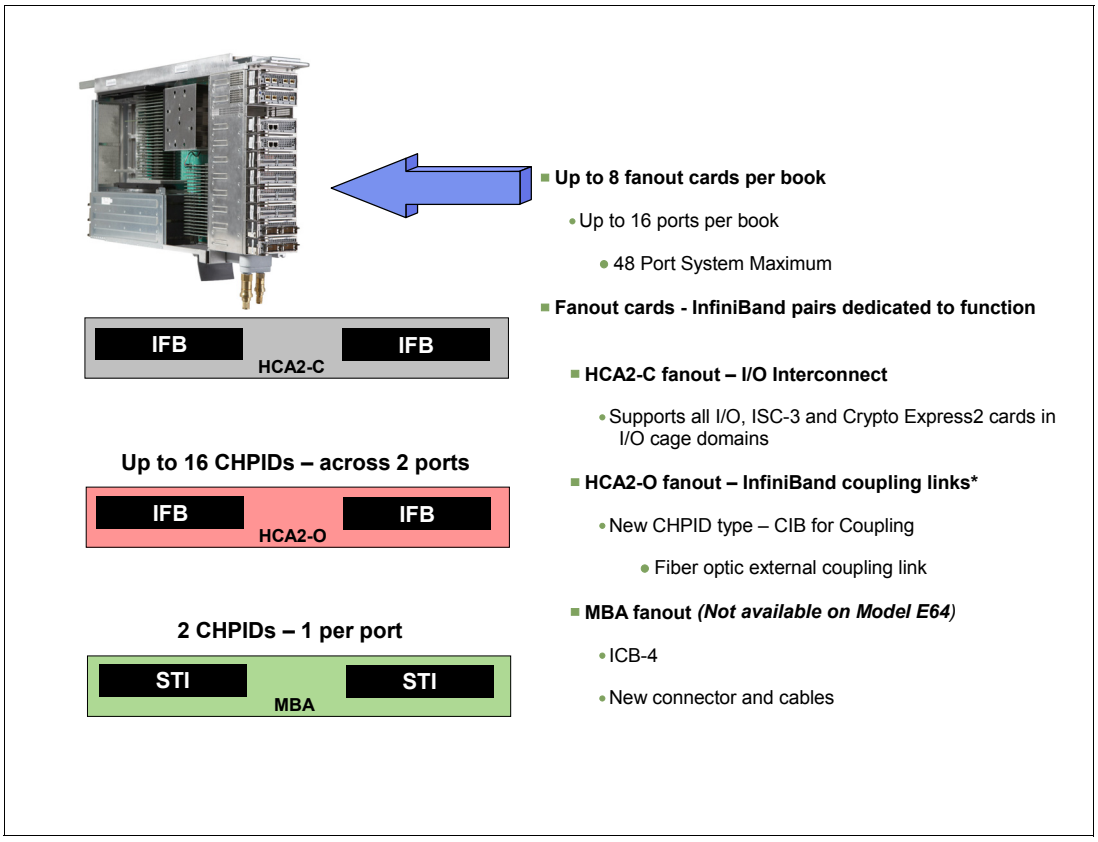


Figure 5-4 Connectivity for coupling and I/O

5.3.3 z10 EC coupling link options

z10 EC offers a choice of four possible kinds of coupling links:

PSIFB A 12x IB-DDR for high speed communication at medium distance. There is a new Coupling using InfiniBand (CHPID CIB). There is also a new 50-micron OM3 (2000 MHz-km) multimode fiber with MPO connectors and up to 150m at up to 6 GBps.

ICB-4 For short distances over copper cabling. New ICB-4 cables are required. Note that z10 EC-to-z10 EC and z10 EC-to-System z9/z990/z890 systems need a 10 meter distance.

ISC-3 For extended distance over fiber optic cabling. There is no change to current cabling.

IC Internal Coupling channels (IC) can be used.

The newly available coupling link option PSIFB (available for both z9 and z10 systems) is based on the InfiniBand technology, as shown in Figure on page 143.

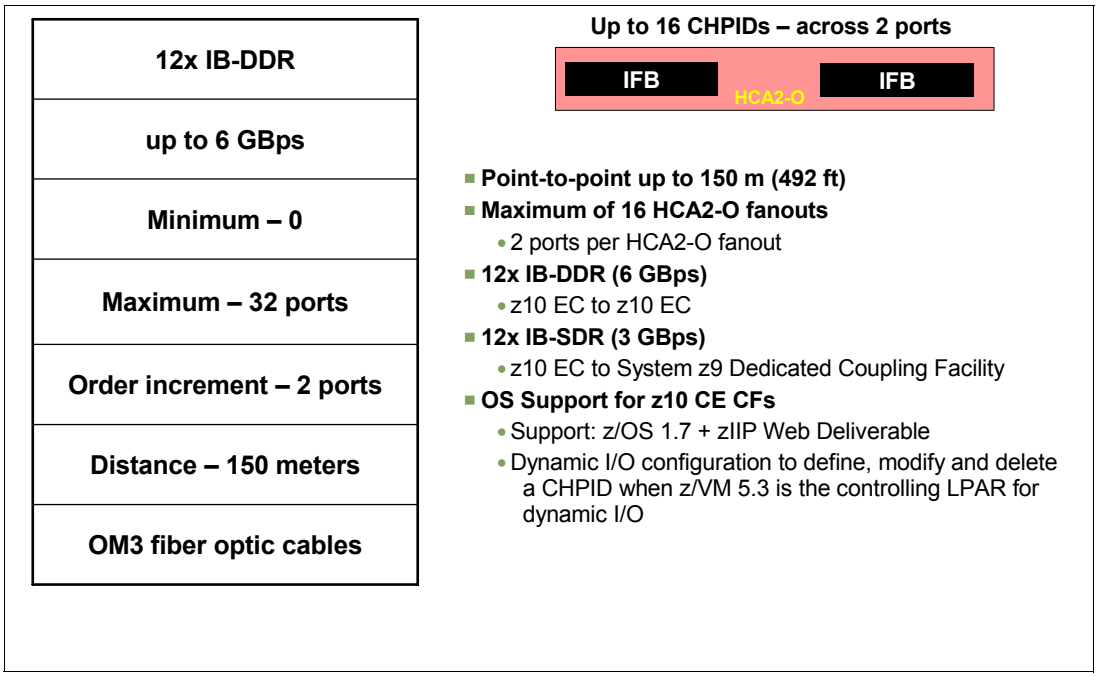


Figure 5-5 z10 EC InfiniBand coupling link connectivity

It is slightly different from the equivalent connectivity card made available for System z9, as shown in Figure 5-6 on page 143.

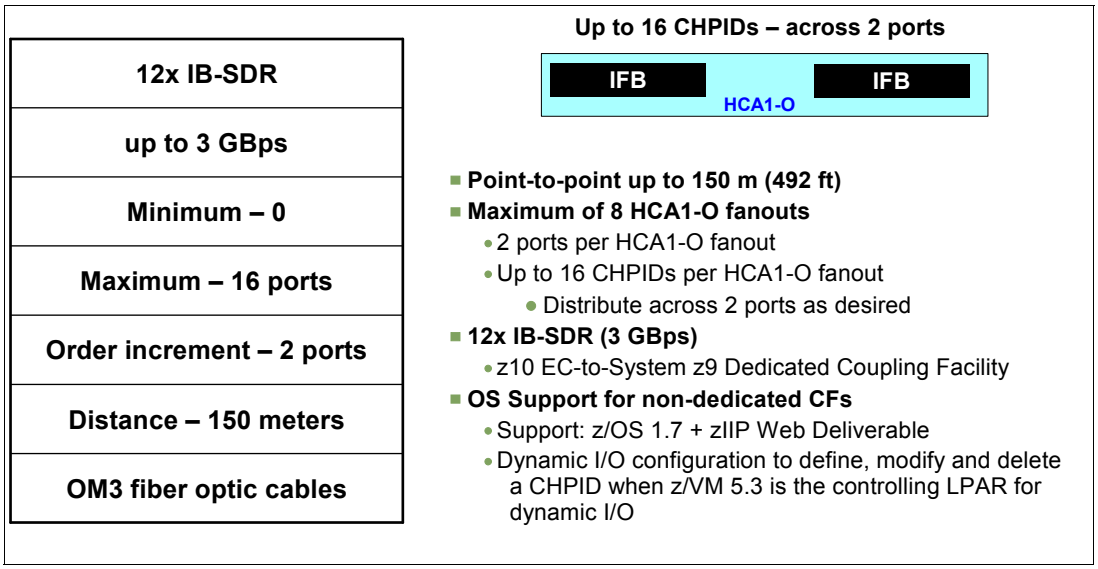


Figure 5-6 System z9 InfiniBand coupling link connectivity

z10 EC coupling link example

As shown in Figure 5-7 on page 144, there are two ways to use InfiniBand to couple a z10 EC:

- To another z10 EC up to 150 meters away at up to 6 GBps
- Or to a z9 EC or z9 BC as dedicated CF only, up to 150 meters away at up to 3 GBps

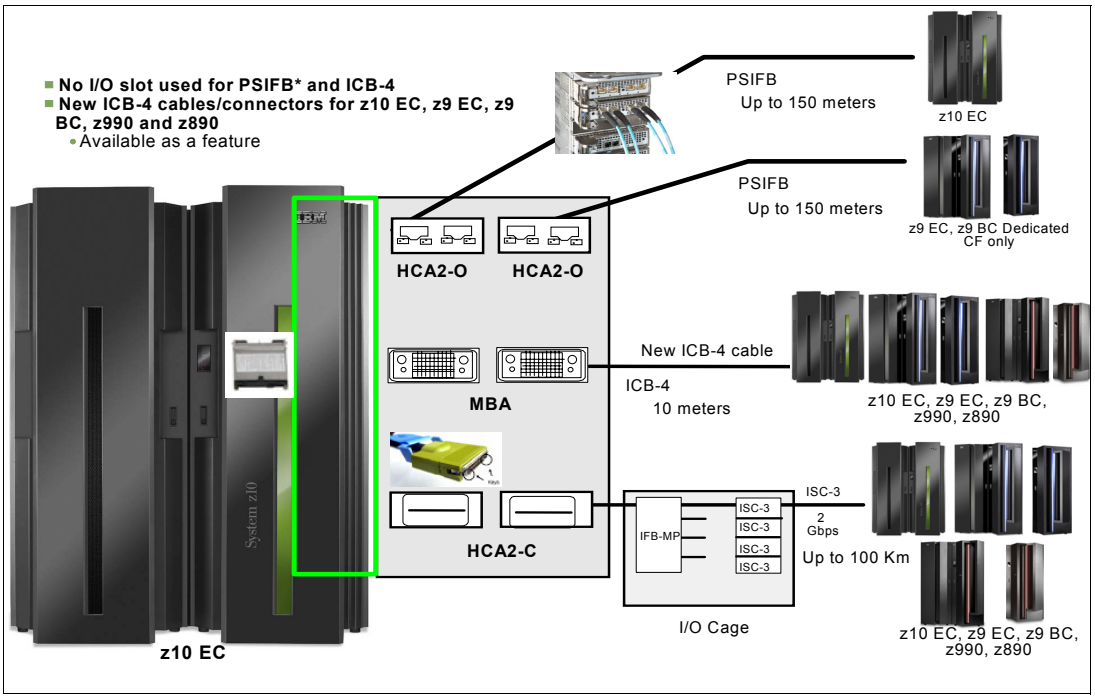


Figure 5-7 System z10 EC coupling links

5.4 I/O connectivity options related to InfiniBand

InfiniBand coupling link multiple channel paths are shown in Figure 5-8 on page 145. This provides up to 16 CHIPIDs using the same physical links and allows for the following:

- More subchannels per physical link
- Not more subchannels per CHPID
- MIF uses the same address, with 7 subchannels per CHPID

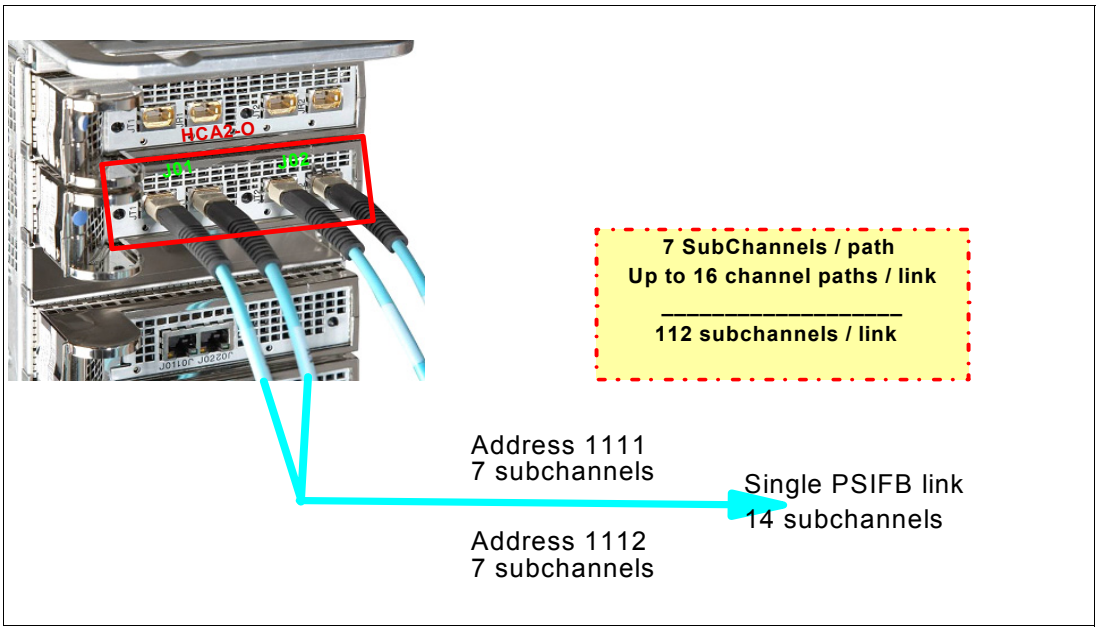


Figure 5-8 InfiniBand coupling link multiple channel paths

System z CF link connectivity–peer mode only

Possible connectivity options in peer mode for CF links are listed in Figure 5-9 on page 145. The figure illustrates that an N-2 server generation connection is allowed. The theoretical maximum rates are shown.

Connectivity Options	z10 EC ISC-3	z10 EC ICB-4	z10 EC PSIFB*
z10 EC/z9/z990/z890 ISC-3	2 Gbps	N/A	N/A
z10 EC/z9/z990/z890 ICB-4	N/A	2 GBps	N/A
z9 Dedicated CF with ISC-3	2 Gbps	N/A	N/A
z9 Dedicated CF with ICB-4	N/A	2 GBps	N/A
z9 Dedicated CF with PSIFB*	N/A	N/A	up to 3 GBps
z10 EC PSIFB*	N/A	N/A	up to 6 GBps

Figure 5-9 System z CF link connectivity – peer mode only

System z configuring CF links

As depicted in Figure 5-10 on page 146, the maximum number of coupling links per system depends on the type that is used with the following:

- Maximum of 32 PSIFB + ICB4 links on System z10 EC
- ICB-4 is not supported on a Model E64

System	PSIFB** (2Q2008)	ICB-4	ICB-3	ISC-3	IC	Max # Links
z10 EC	32*	16* Except E64	N/A	48	32	64
z9 Dedicated CF	16	16	16	48	32	64
Any z9	SOD**	16	16	48	32	64
z990	N/A	16	16	48	32	64
z890	N/A	8	16	48	32	64

Figure 5-10 System z configuring CF links

5.5 Defining InfiniBand coupling links in the HCD/IOCP

Only the InfiniBand for coupling links (either for z10 EC or System z9) will have to be defined in the IOCDS, either through HCD or directly in IOCP.

Adapter ID

New to z10 EC and System z9 for the support of InfiniBand coupling links is an Adapter ID (AID). An AID is used in defining CIB CHPIDs in HCD/IOCP. It has a number range of 00-1F. When installed, each HCA is permanently assigned an Adapter ID (AID), based on the HCA serial number, for as long as it is installed in the same CEC.

The algorithm for assigning Adapter IDs is based on physical location to enable the Order Process to accurately predict the AID for a new HCA, with the following notes:

- The Adapter IDs are shown on the PCHID Report from eConfig when HCA is ordered.
- The assigning of Adapter IDs to an HCA serial number will not be implemented on z9. If an HCA is moved on a z9, its AID will change.
- The Adapter IDs will change on a family upgrade (for example, when upgrading from z9 to z10 EC).

Using and invoking PSIFB

The same validation rules apply to CIB as for ISC and ICB channel paths and STP links, as listed here:

- A CIB channel path can only be connected to another CIB channel path.
- A processor that has a connected CIB channel path must have defined a local system name.
- When a production IODF is built, all CIB channel paths have to be connected.
- A spanned CIB channel path must have defined the same Adapter ID and port for all channel subsystems where it is defined.
- The HCA adapter ID is within the range defined by the processor type.
- The HCA port is within the range defined by the processor type.
- The maximum number of CHPIDs on an HCA is not exceeded.

Support for coupling using InfiniBand can be concurrently added for IOP support on a System z9. However, CIB CHPIDs may not be added (for example, using dynamic) and HCAs may not be installed until after a POR/IML with the new IOP enablement code. Adding the first HCA to a System z9 will be disruptive.

Defining the InfiniBand coupling link

As depicted in Figure 5-11 on page 148, the support InfiniBand coupling link is invoked by:

- Defining the new channel path type CIB (Coupling using IB)
- Specifying the Host Channel Adapter (HCA) ID and port number

Channel path type CIB has the following characteristics:

- CIB channel path can be DED, REC, SHR, or SPAN
- Up to 16 CHPIDs per HCA2-O
- Maximum of 16 CHPIDs per Adapter ID
- 16 CHPIDs can be shared across the two ports of the HCA2-O
- No PCHID value
- Point-to-point connections only
- The target server identified by CSYSTEM on the CHIPID statement
- The local server identified by LSYSTEM on the ID statement

The CIB CHPID type is being defined in the IOCP as:

- Dedicated (DED), Reconfigurable (REC), Shared (SHR), or Spanned (SPAN)
  - AID (Adapter ID)
    - z10 EC 00-1F
    - z9 EC 00-1F
    - z9 BC 08-0F
  - Maximum of 16 CIB CHPIDs per AID
- PORT: port on the HCA where the CHPID is defined
  - Two ports per adapter (1 and 2)
- CSYSTEM: Target system
- CPATH: CSS and CHPID on the target system

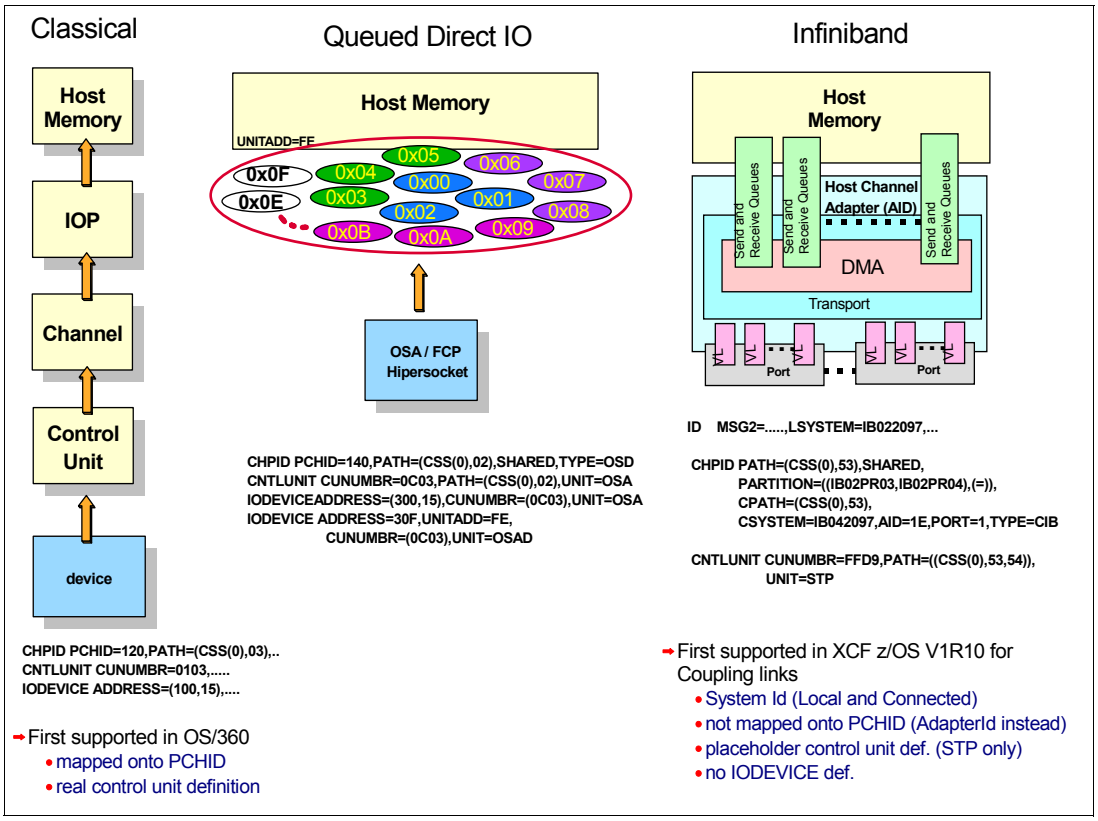


Figure 5-11 InfiniBand I/O definition as compared with previous technology

5.5.1 General I/O configuration definition process

Figure 5-12 illustrates a basic roadmap that you can use to establish a typical PSIFB link configuration.

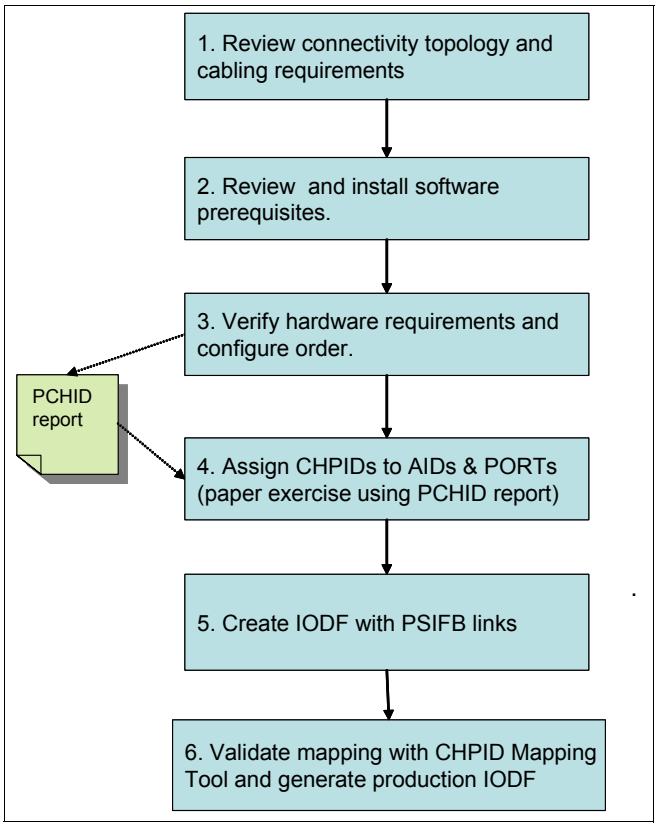


Figure 5-12 PSIFB configuration roadmap

We recommend that you document your planned PSIFB environment with the proper connectivity details. This will ensure a smooth setup of your configuration and will also be useful if needed for problem determination. Subsequent sections illustrate the use of such documentation.

5.6 PSIFB link support

The following connectivity options are available in support of PSIFB links:

- A PSIFB link
- A self-coupling PSIFB link

PSIFB link

A PSIFB link is a coupling link that connects a z/OS logical partition to a CF logical partition via a port on a host channel adapter (HCA) using a physical cable to the equivalent on a different server, as shown in Figure 5-13 on page 150.

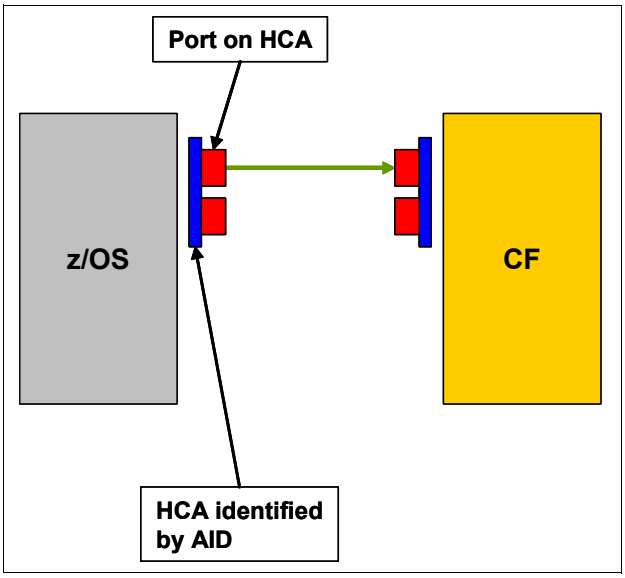


Figure 5-13 PSIFB link between two System z servers

Self-coupling PSIFB link

A self-coupling PSIFB link is a coupling link that connects a z/OS logical partition to a CF logical partition via two ports on two different host channel adapters in the same server, using a physical cable, as shown in Figure 5-14. It can be used in place of internal coupling (IC or ICP) links.

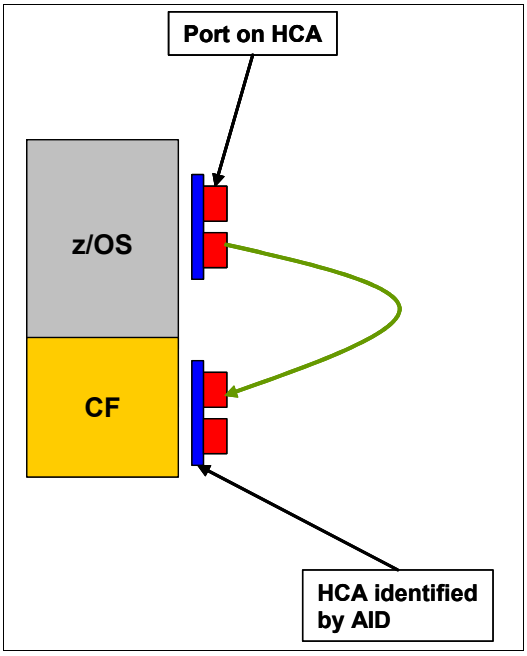


Figure 5-14 Self-coupling PSIFB link

5.6.1 PSIFB IOCP support

IOCP statements are typically built using the Hardware Configuration Dialog (HCD) or the Hardware Configuration Manager (HCM). These statements have been modified to support the configuration of PSIFB links, as listed here.

CHPID type CIB

There is a new CHPID of type CIB (Coupling over InfiniBand) to identify a Coupling over IB channel path. This can be DED, REC, SHR, or SPAN. High level configuration guidelines for CIB CHPIDs are listed here:

- No PCHID can be specified for a CIB CHPID. This is replaced with the Adapter ID (AID) and PORT.
- A spanned CIB CHPID requires the same AID and PORT to be specified for all channel subsystems where it is defined.
- Up to 16 CIB CHPIDs may be defined to the same AID. This is verified at the HCA level, meaning that all 16 could be defined on one port.
- A combination of CIB, CBP, CFP, and ICP CHPIDs are allowed to the same control unit (CF image) up to the maximum of eight paths per CU.
- All CIB CHPIDs on a single control unit must have the same connecting system (CSYSTEM). This is handled by HCD.
- A CIB CHPID may only be connected to another CIB CHPID.
- All CIB CHPIDs must be connected within HCD/HCM before a production IODF can be built.
- A processor that has a connected CIB channel path must have an established local system name (LSYSTEM). This is handled by HCD.
- Only CF-capable partitions may be specified in the access or candidate list of a CF peer channel path.

5.6.2 Adapter identifier (AID)

The Adapter Identifier (AID) specifies the Host Channel Adapter (HCA). This value is in the range 0-1F (or 8-F in the case of a z9 BC server). Adapter IDs are assigned when the server is configured. They are listed in the PCHID report provided by your IBM technical representative when the HCA is ordered and after the install via HMC/SE panels.

PORT

PORT specifies the port (1 or 2) on the Host Channel Adapter.

LSYSTEM

LSYSTEM specifies the system name of the local server. It is an alphanumeric value of 1-8 characters. It may begin with either an alphabetic character or a numeric character. All alphabetic characters are upper case. HCD will default to the CPC name specified for the Processor ID.

CSYSTEM

CSYSTEM is the system name of the target server connected to by the LSYSTEM. It is an alphanumeric value of 1-8 characters. It may begin with either an alphabetic character or a numeric character. All alphabetic characters are upper case.

CPATH

CPATH specifies the CSSID and CHPID on the connecting system (or the CSYSTEM). This is handled by HCD.

5.7 Sample configuration with PSIFB links

This section defines a sample Parallel Sysplex configuration exploiting PSIFB links and details the process required to configure it. Figure 5-15 illustrates the PSIFB links in this configuration. The sections that follow document the connectivity detail and the steps required to define the different types of PSIFB links.

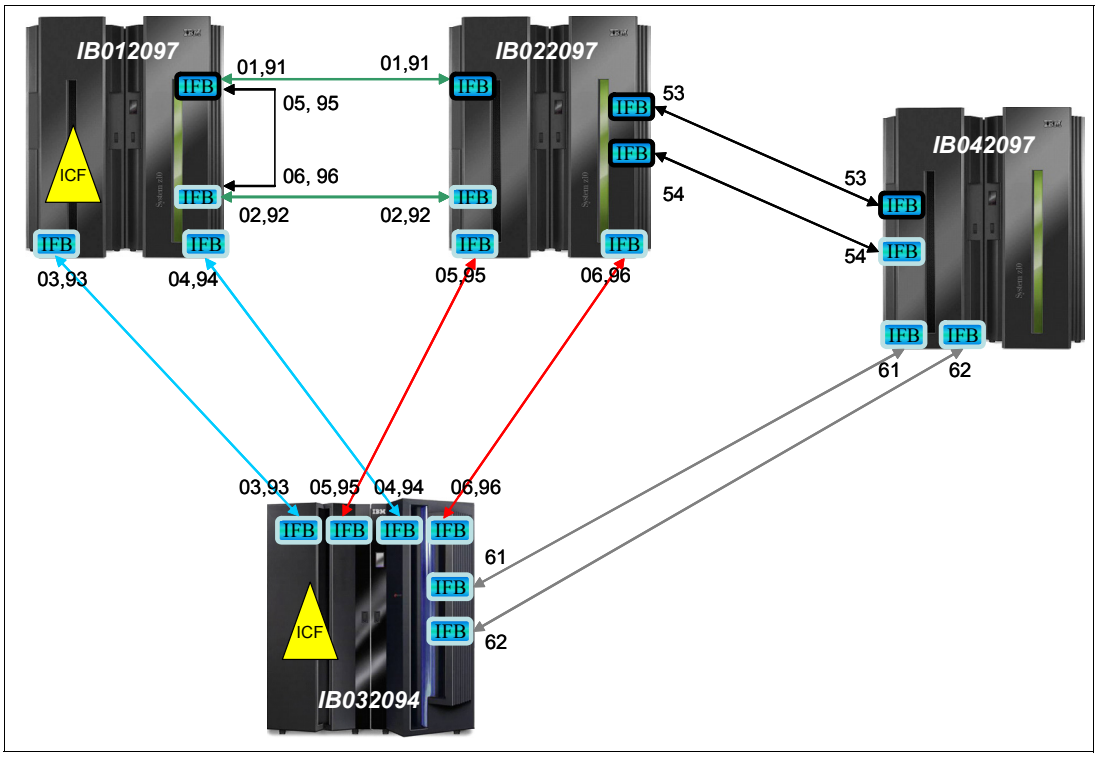


Figure 5-15 Sample configuration with PSIFB links

Connectivity detail pertaining to the configuration in Figure 5-15 is as follows:

- Each PSIFB link reflects a PORT on an Adapter ID (AID).
- Each PSIFB link has CHPID numbers listed (0x, 9x, 5x, 6x).
- CHPIDs prefixed 0 are PSIFB links from the PROD sysplex.
- CHPIDs prefixed 9 are PSIFB links from the DEV sysplex.
- CHPIDs prefixed 5 are PSIFB timing only links between the IB022097 and IB042097 servers.
- CHPIDs prefixed 6 are PSIFB links between the IB032094 and IB042097 servers.

Logical partition configuration

To understand and qualify the connectivity, the logical partition (LPAR) configuration must be available for reference. This is reflected in Table 5-1 on page 153.

Table 5-1 Logical partition configuration details

System name	Model	LPARs in CSS0	TYPE	Sysplex	LPARs in CSS1	TYPE	Sysplex
IB012097	2097-E56	IB01PR01	OS	PROD	IB01DV01	OS	DEV
		IB01PR02	OS	PROD	IB01DV02	OS	DEV
		IB01PRC1	CF	PROD	IB01DVC1	CF	DEV
IB022097	2097-E56	IB02PR03	OS	PROD	IB02DV03	OS	DEV
		IB02PR04	OS	PROD	IB02DV04	OS	DEV
IB032094	2094-S08	IB03PRC2	CF	PROD			
		IB03DVC2	CF	DEV			
IB042097[a]	2097-E12	IB04SDM	OS	N/A			

a. For the purpose of this exercise, IB042097 is a System DataMover (SDM) LPAR; not part of any sysplex.

Adapter ID (AID) and port assignments

Adapter ID information is found in the PCHID report for the machine. Details for the servers in this configuration are provided in Figure 5-16 on page 154.

```
PCHID report for System IB012097
CHPIDSTART
  31160460                      PCHID REPORT                      Nov 07,2007
  Machine: 2097-E56  SNxxxxxxx
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Source           Cage  Slot  F/C    PCHID/Ports or AID             Comment
  06/D9            A25B  D906  0163   AID=0E

  15/D9            A25B  D915  0163   AID=1E
PCHID report for System IB022097
CHPIDSTART
  31160906                      PCHID REPORT                      Nov 07,2007
  Machine: 2097-E56  SNxxxxxxx
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Source           Cage  Slot  F/C    PCHID/Ports or AID             Comment
  06/D9            A25B  D906  0163   AID=0E

  15/D9            A25B  D915  0163   AID=1E

  10/D9            A25B  D910  0163   AID=16
PCHID report for System IB032094
CHPIDSTART
  31161245                      PCHID REPORT                      Nov 07,2007
  Machine: 2094-S08  SNxxxxxxx
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Book/Fanout/Jack Cage  Slot  F/C    PCHID/Ports                    Comment
  0/D5             A19B  D506  0167   AID=0C
0/D7             A19B  D706  0167   AID=0E

  0/D8             A19B  D806  0167   AID=0F
PCHID report for System IB042097
CHPIDSTART
  31161581                      PCHID REPORT                      Nov 07,2007
  Machine: 2097-E12  SNxxxxxxx
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Source           Cage  Slot  F/C    PCHID/Ports or AID             Comment
  06/D7            A25B  D706  0163   AID=0C

  06/D8            A25B  D806  0163   AID=0D
```

Figure 5-16 PCHID reports for sample configuration

Mapping CIB CHPIDs across the fanouts and ports

The CIB CHPIDs should be mapped across fanouts and ports for availability. At this time, this is a manual process as the CHPID Mapping Tool does not perform this mapping. Remove single points of failure within your configuration so that systems or sysplexes are mapped across multiple fanouts and ports. The mapping for this sample configuration is listed in Table 5-2.

Table 5-2 System CHPID assignments across IFB fanouts and ports

System	Adapter ID	PORT	PROD sysplex CHPIDs	DEV sysplex CHPIDs	CHPIDs to/from SDM (IB042097)
IB012097	0E	1	01,05	91	
		2	03	93, 95	
	1E	1	02, 06	92	
		2	04	94,96	
IB022097	0E	1	01	91	
		2	05	95	
	1E	1			53
		2	06	96	
	16	1	02	92	
		2			54
IB032094	0C	1	05	95	
		2	03	93	
	0E	1			61
		2	06	96	
	0F	1			62
		2	04	94	
IB042097	0C	1			53
		2			61
	0D	1			54
		2			62

5.7.1 Defining the local system name

A local system name (LSYSTEM in the IOCP) is required on every server that will use PSIFB links. When the local system name is not present, it will default to the CPC name. We recommend using the CPC name instead of LSYSTEM with the same value specified as the Processor ID. This will help you to avoid naming conflicts. The CPC name is defined in the processor definition of the server as shown in Figure 5-17 on page 156.

```
                        Change Processor Definition


Specify or revise the following values.

Processor ID . . . . . : IB012097
Support level:
XMP, 2097 support
Processor type . . . . . 2097      +
Processor model  . . . . . E56       +
Configuration mode . . . . . LPAR      +

Serial number  . . . . . __________  +
Description  . . . . . ____________________________________

Specify SNA address only if part of an S/390 microprocessor cluster:

Network name . . . . . ________ +
CPC name . . . . . IB012097 +

Local system name  . . . . . ________
```

Figure 5-17 Defining the Local System Name

5.7.2 Defining PSIFB links

This section details the definition steps required to define PSIFB links between IB012097 and IB022097 (as shown in Figure 5-15 on page 152) using HCD. This includes CIB CHPID definitions for just the PROD sysplex.

The following CIB CHPIDs are defined and connected in accordance with Figure 5-15 on page 152:

- IB012097 CHPID 01 connects to IB022097 CHPID 01

Defining the CIB channel paths

To define the CIB channel paths, follow these steps:

1. From the HCD Main Menu, select option **1.3** and press **Enter**. This takes you to the processor list shown in Figure 5-18.

```
 Processor List       Row 1 of 12 More:       >
Command ===> ____________________________________________ Scroll ===> PAGE

Select one or more processors, then press Enter. To add, use F11.


/ Proc. ID Type +   Model +  Mode+ Serial-# + Description
s IB012097 2097     E56      LPAR  __________ ____________________________________
_ IB022097 2097     E56      LPAR  __________ ____________________________________
_ IB032094 2094     S08      LPAR  __________ ____________________________________
_ IB042097 2097     E12      LPAR  __________ ____________________________________
```

Figure 5-18 HCD processor list

2. Select the processor to work with (in our case, IB012097) and type s to work with channel subsystems for this server, as shown in Figure 5-18 on page 157.

```
------------------------------------------------------------------------------
                           Channel Subsystem List                 Row 1 of 4
Command ===> ____________________________________________ Scroll ===> PAGE

Select one or more channel subsystems, then press Enter.  To add, use F11.

Processor ID . . . : IB012097

  CSS Devices in SS0    Devices in SS1
/ ID  Maximum + Actual  Maximum + Actual  Description
s 0   65280     0       65535     0       ____________________________________
_ 1   65280     0       65535     0       ____________________________________
_ 2   65280     0       65535     0       ____________________________________
_ 3   65280     0       65535     0       ____________________________________
```

Figure 5-19 Channel Subsystem List

Type s for the appropriate Channel Subsystem (0, in our example in Figure 5-19 on page 157) and then press Enter to display the Channel Path list for this server.

3. From the Channel Path List press PF11 to add a CHPID, as shown in Figure 5-20 on page 158.

```
                              Add Channel Path


Specify or revise the following values.

Processor ID . . . . : IB012097
Configuration mode . : LPAR
Channel Subsystem ID : 0

Channel path ID  . . . . 01    +            PCHID . . . ___
Number of CHPIDs . . . . 1
Channel path type  . . . CIB   +
Operation mode . . . . . SHR   +
Managed  . . . . . . . . No   (Yes or No)   I/O Cluster ________  +
Description  . . . . . . PROD SYSPLEX to IB022097

Specify the following values only if connected to a switch:
Dynamic entry switch ID   _  + (00 - FF)
Entry switch ID  . . . . __  +
Entry port . . . . . . . __  +
```

Figure 5-20 Adding a CIB Channel Path

4. On the Add Channel Path panel, fill in the appropriate detail and press Enter:
   - The Channel Path ID (`01`, in our configuration).
   - The Channel Path Type (`CIB` for a Coupling over InfiniBand CHPID).
   - The Operation mode (`SHR`, in our configuration). This CHPID will provide connectivity from logical partition IB01PRC1 to those on the IB022097 server.
   - A description (optional).

   Note: The PCHID field is not valid for a CIB CHPID but is still included in the panel.

5. The next panel requires details of the HCA attributes for this CHPID, as shown in Figure 5-21 on page 158. This detail was established in the mapping exercise we performed earlier (see Table 5-2 on page 155).

```
                  Specify HCA Attributes


Specify or revise the values below.

Adapter ID of the HCA  . . 0E  +
Port on the HCA  . . . . . 1   +
```

Figure 5-21 Defining Adapter ID and Port assignments

6. Type in the Adapter ID and PORT details and press Enter. The Access List panel will display, as shown in Figure 5-22 on page 159.

```
                          Define Access List
                                                                  Row 1 of 3
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more partitions for inclusion in the access list.

Channel subsystem ID : 0
Channel path ID  . . : 01     Channel path type  . : CIB
Operation mode . . . : SHR    Number of CHPIDs . . : 1

/ CSS ID Partition Name   Number Usage Description
/ 0      IB01PRC1         3      CF
_ 0      IB01PR01         1      OS
_ 0      IB01PR02         2      OS
```

Figure 5-22 Channel Path Access List

7. Specify / for each LPAR required in the access list. In our configuration it is only the IB01PRC1 coupling facility LPAR.
8. Press Enter and add any additional LPARs to the Candidate List for this CIB channel (none, in our case) until the Channel Path List for this CSS ID is redisplayed as shown in Figure 5-23 on page 159. This shows the unconnected CIB CHPID on IB012097.

```
                             Channel Path List       Row 1 of 1 More:       >
Command ===> _____________________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter. To add use F11.

Processor ID . . . . : IB012097
Configuration mode . : LPAR
Channel Subsystem ID : 0

                    DynEntry Entry +
/ CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
f 01    CIB   SHR   __       __ __   N   No   PROD SYSPLEX to IB022097
```

Figure 5-23 Channel Path list with unconnected CIB CHPID 01

Note: At this point, we return to the processor list and perform the same procedure to define CHPID 01 to the IB022097 processor on AID 0E, PORT1. The LPARs from the PROD sysplex (IB02PR03 and IB02PR04) are added to the Access list.

Connecting the CIB CHPIDs

At this stage, we have defined the CIB CHPIDs for a PSIFB link. However, they cannot be used until they have been connected. Furthermore, an attempt to build a production IODF will fail if there are unconnected CIB CHPIDs.

Follow these steps to connect the CIB CHPIDs:

1. Return to the Channel Path List of one of the associated servers; see Figure 5-23 on page 159. Type f next to the associated CHPID and press Enter to display the CF Channel Path Connectivity List, as shown in Figure 5-24 on page 160.

```
------------------------------------------------------------------------------
                   CF Channel Path Connectivity List               Row 1 of 1
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IBO12097
Source channel subsystem ID . : 0
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
p 01     CIB   SHR  N
```

Figure 5-24 CF Channel Path Connectivity List with unconnected CIB CHPID 01

2. Type p next to the associated CHPID to display the Channel Path Connection Panel shown in Figure 5-25 on page 160.

```
                  Connect to CF Channel Path


 Specify the following values.

 Source processor ID  . . . . . : IBO12097
 Source channel subsystem ID  . : 0
 Source channel path ID . . . . : 01
 Source channel path type . . . : CIB

 Destination processor ID . . . . . IBO22097  +
 Destination channel subsystem ID . . 0   +
 Destination channel path ID  . . . . 01  +

 Timing-only link . . . . . No
```

Figure 5-25 Channel Path Connection Panel

3. On the Channel Path Connection Panel, enter:
   - Destination processor ID (IBO22097, in our configuration)
   - Destination channel subsystem ID (0, in our configuration)
   - Destination channel path ID (01, in our configuration)
   - This is not a Timing-only link, so we accepted the default of No.
4. Press Enter to display a confirmation panel that includes the Control Unit number and devices that are to be generated as a result of the operation (see Figure 5-26 on page 161).

```
                Add CF Control Unit and Devices


Confirm or revise the CF control unit number and device numbers
for the CF control unit and devices to be defined.

Processor ID . . . . . : IB022097
Channel subsystem ID . . . : 0
Channel path ID  . . . . . : 01         Operation mode . . : SHR
Channel path type  . . . . : CIB

Control unit number  . . . . FFDC  +

Device number  . . . . . FD86
Number of devices  . . . . : 7
```

Figure 5-26 Channel path connection confirmation panel

HCD selects the highest unused control unit and device numbers automatically. If necessary, these can be changed to numbers of your choice.

5. Press Enter to complete the operation and the CF Channel Path Connectivity List is redisplayed (see Figure 5-27 on page 161). This now displays our new CHPIDs in connected status.

```
                  CF Channel Path Connectivity List              Row 1 of 1
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IB012097
Source channel subsystem ID . : 0
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 01     CIB   SHR  N     IB022097.0   01     CIB   SHR    CFP
```

Figure 5-27 CF Channel Path Connectivity List with connected CIB CHPIDs 01

The PSIFB link definition is now complete.

Adding additional CHPIDs to an existing PSIFB link

Up to 16 CHPIDs may be configured across the two available ports on a HCA. This is a significant benefit over other coupling links where each PCHID (and channel adapter) is associated with an individual CHPID.

This section details the steps to add additional CHPIDs to an existing PSIFB link. In section “Defining PSIFB links” on page 156 we defined the following CHPIDs for the PROD sysplex:

- IB012097 CHPID 01 on AID 0E, PORT1
- IB022097 CHPID 01 on AID 0E, PORT1

This PSIFB link will provide the connectivity for the DEV sysplex from system IB022097 to the IB01DVC1 coupling facility on system IB012097.

We will use the same AIDs and PORTs to add additional CHPIDs for the DEV sysplex as follows:

- IB012097 CHPID 91 on AID 0E, PORT1
- IB022097 CHPID 91 on AID 0E, PORT1

The process is identical to that followed for the definition of the original CHPIDs on these AIDs and PORTs, as follows:

1. Select the IB012097 processor from the HCD processor list. For reference see Figure 5-18 on page 157.
2. Select the appropriate CSS ID (1 in this example). For reference see Figure 5-19 on page 157.
3. Add the new CHPID (91 in our example) and add to AID 0E, PORT1. For reference see Figure 5-20 on page 158 and Figure 5-21 on page 158.
4. Add partition IB01DVC1 to the Access list (as in Figure 5-22 on page 159) and press Enter to return to the Channel Path List. This will show the new CHPID 91 as unconnected (see Figure 5-28 on page 162.

```
                  CF Channel Path Connectivity List               Row 1 of 1
Command ===> ___________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IB012097
Source channel subsystem ID . : 1
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 91     CIB   SHR  N
```

Figure 5-28 CF Channel Path Connectivity list with unconnected CIB CHPID 91

5. Repeat the process to add CHPID 91 in CSS1 to the IB022097 processor. Add partitions IB02DV03 and IB02DV04 to the access list.
6. Connect CHPID 91 on IB012097 to CHPID 91 on IB022097 as described in “Connecting the CIB CHPIDs” on page 159.
7. The connected CIB CHPIDs will then be displayed on the CF Channel Path Connectivity List as shown in Figure 5-29 on page 163.

```
------------------------------------------------------------------------
                  CF Channel Path Connectivity List              Row 1 of 1
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IB012097
Source channel subsystem ID . : 1
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 91     CIB   SHR  N     IB022097.1   91     CIB   SHR    CFP
```

Figure 5-29 CF Channel Path Connectivity List with connected CIB CHPIDs 91

5.7.3 Defining timing-only PSIFB links

This section explains the steps to follow to define a timing-only PSIFB links. These links are required for connectivity in a Coordinated Timing Network (CTN) when exploiting the Server Time Protocol (STP) capability. When there are only operating system images at both ends of the link (that is, when there is no Coupling Facility on either server), then a timing-only link must be defined to provide the required connectivity.

In this sample configuration, the links that fit this scenario are shown in Figure 5-15 on page 152. IB022097 CHPIDs 53 and 54 connect to IB042097 CHPIDs 53 and 54.

Follow these steps to define the timing-only links for this configuration.

1. Define CIB CHPIDs 53 and 54 to IB022097 and IB042097 using the process documented in “Defining the CIB channel paths” on page 156. These are assigned to Adapter IDs and PORTs as listed in Table 5-2 on page 155. After they are defined, the Channel Path list will reflect the CHPIDs as unconnected. This is shown in Figure 5-30 on page 163.

```
                            Channel Path List       Row 1 of 2 More:       >
Command ===> ____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter. To add use F11.

Processor ID . . . . : IB042097
Configuration mode . : LPAR
Channel Subsystem ID : 0

                    DynEntry Entry +
/ CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
f 53    CIB   SHR   __       __ __   N   No   Timing only link to IB022097
_ 54    CIB   SHR   __       __ __   N   No   Timing only link to IB022097
```

Figure 5-30 Channel Path List with unconnected CHPIDs 53 and 54

2. Connect the CIB CHPIDs between servers IB022097 and IB042097 by entering `f` next to one of the unconnected CHPIDs, as shown in Figure 5-30 on page 163. This will display the CF Channel Connectivity List; see Figure 5-31 on page 164.

```
                    CF Channel Path Connectivity List               Row 1 of 2
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IBO42097
Source channel subsystem ID . : 0
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
p 53     CIB   SHR  N
_ 54     CIB   SHR  N
```

Figure 5-31 CF Channel Connectivity List

3. Entering `p` next to the CHPID that you want to connect displays the Channel Path Connection Panel; see Figure 5-32 on page 164.

```
                 Connect to CF Channel Path


 Specify the following values.

 Source processor ID  . . . . . : IBO42097
 Source channel subsystem ID  . : 0
 Source channel path ID . . . . : 53
 Source channel path type . . . : CIB

 Destination processor ID . . . . . IBO22097  +
 Destination channel subsystem ID . . 0   +
 Destination channel path ID  . . . . 53  +

 Timing-only link . . . . . Yes
```

Figure 5-32 Channel Path Connection panel

4. On the Channel Path connection panel enter the following values:
   - Destination processor ID (`IBO22097`, in our configuration)
   - Destination channel subsystem ID (`0`, in our configuration)
   - Destination channel path ID (`53`, in our configuration)
   - Timing-only link, specify `Yes`
5. Press Enter to display a confirmation panel as shown in Figure 5-33 on page 165. This includes the Control Unit number that will be generated as a result of the operation. HCD selects the highest unused control unit, but this may be changed to a number of your choice.

Note: No Device numbers are generated or used by a Timing only link, so the field is left blank. Any attempt to add a value in this field will be rejected as invalid.

```
                  Add CF Control Unit and Devices


 Confirm or revise the CF control unit number and device numbers
 for the CF control unit and devices to be defined.

 Processor ID . . . . . : IB042097
 Channel subsystem ID . . . : 0
 Channel path ID  . . . . . : 53         Operation mode . . : SHR
 Channel path type  . . . . : CIB

 Control unit number  . . . . FFDA  +

 Device number  . . . . . ____
 Number of devices  . . . . : 0
```

Figure 5-33 Channel path connection confirmation panel: Timing-only link

6. Press Enter to complete the connection and return to the CF Channel Path Connectivity List. This will now show the connected PSIFB Timing-only link with a control unit type of STP, as shown in Figure 5-34 on page 165.

```
  Goto  Filter  Backup  Query  Help
------------------------------------------------------------------------------
                    CF Channel Path Connectivity List               Row 1 of 2
Command ===> ____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IB042097
Source channel subsystem ID . : 0
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 53     CIB   SHR  N     IB022097.0   53     CIB   SHR    STP
_ 54     CIB   SHR  N
```

Figure 5-34 CF Channel Path Connectivity List with connected Timing-only link

7. The PSIFB Timing-only link definition is now complete for CHPID 53. You can follow the same process for CHPID 54.

5.7.4 Defining self-coupling PSIFB links

This section explains the definition steps required to define self-coupling PSIFB links. The sample configuration uses self-coupling links on the IB012097 server as shown in Figure 5-15 on page 152.

The associated CHPIDs are as follows:

- IB012097 CHPIDs 05 and 06 provide self-coupling connectivity for the PROD sysplex
  - z/OS LPARs IB01PR01 and IB01PR02 to CF LPAR IB01PRC1
- IB012097 CHPIDs 95 and 96 provide self-coupling connectivity for the DEV sysplex
  - z/OS LPARs IB01DV01 and IB01DV02 to CF LPAR IB01DVC1

The process required to define self-coupling PSIFB links is very similar to that of external links. The only difference is that connected CHPIDs are connected to links within the same server.

This example defines CHPIDs 95 and 96 to establish the PSIFB link for the DEV sysplex. They are defined on AID 0E, PORT2 and AID, 1E Port2, as illustrated in Figure 5-2 on page 155.

Defining the CIB channel paths

Follow these steps to define CIB channel paths.

1. From the HCD main Menu, select option **1.3** to take you to the processor list (see Figure 5-18 on page 157). Select the IB012097 processor and enter `s` to work with channel subsystems for this server.
2. Specify `s` next to the appropriate CSS ID (`1` for the DEV sysplex) and press Enter to display the Channel Path list for this server.
3. From the Channel Path list press PF11 to add a new CHPID, as shown in Figure 5-35 on page 166.

```
                          Add Channel Path


Specify or revise the following values.

Processor ID . . . . : IB012097
Configuration mode . : LPAR
Channel Subsystem ID : 1

Channel path ID  . . . . 95    +            PCHID . . . ___
Number of CHPIDs . . . . 1
Channel path type  . . . CIB   +
Operation mode . . . . . SHR   +
Managed  . . . . . . . . No   (Yes or No)   I/O Cluster ________  +
Description  . . . . . . Self Cple CIB (DEV from IB01DVC1)

Specify the following values only if connected to a switch:
Dynamic entry switch ID  __  + (00 - FF)
Entry switch ID  . . . . __  +
Entry port . . . . . . . __  +
```

Figure 5-35 Adding an internal CIB channel path

4. On the Add Channel Path panel, fill in the appropriate details and press Enter:
   - The Channel Path ID (`95`, in our configuration)
   - The Channel Path type (CIB for a Coupling over InfiniBand CHPID)

– The Operation mode (SHR, in our configuration. This CHPID defines the IB01DVC1 CF and will connect and be defined to IB01DV01 and IB01DV02 z/OS LPARs).
– A description (optional).

Note: Although the PCHID field is not valid for a CIB CHPID, it is shown in the panel.

5. On the next panel, specify the HCA details as illustrated in Figure 5-36 on page 167. These details are taken from the results in Table 5-2 on page 155.

```
                  Specify HCA Attributes


Specify or revise the values below.

Adapter ID of the HCA  . . 0E  +
Port on the HCA  . . . . . 2   +
```

Figure 5-36 Defining Adapter ID and Port Assignments for an internal CIB CHPID

6. Press **Enter** to display the Access List, as shown in Figure 5-37 on page 167.

```
                          Define Access List
                                                                  Row 1 of 3
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more partitions for inclusion in the access list.

Channel subsystem ID : 1
Channel path ID  . . : 95     Channel path type  . : CIB
Operation mode . . . : SHR    Number of CHPIDs . . : 1

/ CSS ID Partition Name   Number Usage Description
/ 1      IB01DVC1         3      CF
_ 1      IB01DV01         1      OS
_ 1      IB01DV02         2      OS
```

Figure 5-37 Channel Path Access List

7. Specify / for each LPAR required in the access list (IB01DVC1, in this example).
8. Press **Enter** to add any additional LPARs to the Candidate list for this CHPID (there are none in this example) until the Channel Path List of this CSS ID is redisplayed.

Note: At this point, we returned to the processor list and performed the same procedure to define CHPID 96 to the IB012097 processor on an AID and PORT that was different from where we had defined CHPID 95 (AID 1E, PORT2 for CHPID 95). We added z/OS LPARs from the DEV sysplex (IB01DV01 and IB01DV02) to the Access list for CHPID 96.

9. The Channel Path List then displayed the newly defined CHPIDs (95 and 96) in unconnected status, as shown in Figure 5-38 on page 168.

```
                              Channel Path List       Row 1 of 3 More:       >
Command ===> ___________________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter. To add use F11.

Processor ID . . . . : IBO12097
Configuration mode . : LPAR
Channel Subsystem ID : 1

                     DynEntry Entry +
/ CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
_ 91    CIB   SHR   __       __ __   Y   No   DEV sysplex to IBO22097
f 95    CIB   SHR   __       __ __   N   No   Self Cple CIB (DEV from IBO1DVC1)
_ 96    CIB   SHR   __       __ __   N   No   Self C CIB (DEV from IBO1DV01/02)
```

Figure 5-38 Channel Path List with unconnected CIB CHPIDs 95 and 96

Connecting the CIB CHPIDs

Follow these steps to connect the CIB CHPIDs.

At this point the self-coupling CIB CHPIDs 95 and 96 have been defined, but until they are connected they cannot be used. Furthermore, an attempt to build a production IODF would fail until these CHPIDs have been connected.

1. From the Channel Path list shown in Figure 5-38 on page 168, select `f` next to one of the unconnected CHPIDs to display the CF Channel Path Connectivity List as shown in Figure 5-39 on page 168.

```
                    CF Channel Path Connectivity List               Row 1 of 3
Command ===> ___________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IBO12097
Source channel subsystem ID . : 1
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 91     CIB   SHR  N     IBO22097.1   91     CIB   SHR    CFP
p 95     CIB   SHR  N
_ 96     CIB   SHR  N
```

Figure 5-39 CF Channel Path Connectivity List with unconnected CHPIDs 95 and 96

2. Select `p` next to the associated CHPID to display the Channel Path Connection panel shown in Figure 5-40 on page 169.

```
                Connect to CF Channel Path


Specify the following values.

Source processor ID  . . . . . : IBO12097
Source channel subsystem ID  . : 1
Source channel path ID . . . . : 95
Source channel path type . . . : CIB

Destination processor ID . . . . . IBO12097  +
Destination channel subsystem ID . . 1   +
Destination channel path ID  . . . . 96  +

Timing-only link . . . . . No
```

Figure 5-40 Channel Path Connection panel

3. On the Channel Path Connection panel, enter the following values:
   - Destination processor ID. This is the same as the Source Processor ID (IBO12097, in our configuration) because it is a self-coupling CIB link.
   - Destination Channel Subsystem ID (1, in our configuration).
   - Destination channel path ID (96, in our configuration).
   - This is not a Timing-only link, so in our case we accepted the default of No.
4. Press **Enter** to display a confirmation panel, as shown in Figure 5-41 on page 169. This shows the control unit and device numbers that will be generated as a result of the operation.

```
                Add CF Control Unit and Devices


Confirm or revise the CF control unit number and device numbers
for the CF control unit and devices to be defined.

Processor ID . . . . . : IBO12097
Channel subsystem ID . . . : 1
Channel path ID  . . . . . : 96         Operation mode . . : SHR
Channel path type  . . . . : CIB

Control unit number  . . . . FFD8  +

Device number  . . . . . FD78
Number of devices  . . . . : 7
```

Figure 5-41 Channel Path Connection Confirmation panel

HCD selects the highest unused control unit and device numbers automatically. If necessary, you can change these to numbers of your choice.

5. Press **Enter** to complete the operation. The CF Channel Connectivity List will be redisplayed. Our CHPIDs were displayed in connected status, as shown in Figure 5-42 on page 170.

```
                  CF Channel Path Connectivity List               Row 1 of 3
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IB012097
Source channel subsystem ID . : 1
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode   Type
_ 91     CIB   SHR  N     IB022097.1   91     CIB   SHR    CFP
_ 95     CIB   SHR  N     IB012097.1   96     CIB   SHR    CFP
_ 96     CIB   SHR  N     IB012097.1   95     CIB   SHR    CFP
```

Figure 5-42 CF Channel Path Connectivity List with connected CIB CHPIDs 95 and 96

The self-coupling PSIFB link definition is now complete.

5.7.5 IOCP Reference for PSIFB links

This section provides a sample of IOCP statements from the configuration described in Figure 5-15 on page 152.

Note: Various IOCP statements (for example, RESOURCE) have been removed from these examples to improve the readability of the PSIFB link detail.

Sample IOCP for PSIFB links

This sample lists the IOCP statements for the following PSIFB links between servers IB012097 and IB022097:

- IB012097 CHPID 01 on AID 0E, PORT1
- IB022097 CHPID 01 on AID 0E, PORT1

Figure 5-43 on page 171 shows the IOCP for each of these servers.

```
IOCP Statements from IB012097:
 ID    MSG2='IB01.IODF34.WORK - 2007-11-09 16:45',             *
       SYSTEM=(2097,1),LSYSTEM=IB012097,                       *
       TOK=('IB012097',00800001DE5020971645116301073l3F00000000*
       ,00000000,'07-11-09','16:45:11','........','........')
  CHPID PATH=(CSS(0),01),SHARED,PARTITION=((IB01PRC1),(=)),     *
       CPATH=(CSS(0),01),CSYSTEM=IB022097,AID=0E,PORT=1,       *
       TYPE=CIB
IOCP Statements from IB022097:
 ID    MSG2='IB01.IODF34.WORK - 2007-11-09 16:45',             *
       SYSTEM=(2097,1),LSYSTEM=IB022097,                       *
       TOK=('IB022097',00800001DE5020971645116301073l3F00000000*
       ,00000000,'07-11-09','16:45:11','........','........')
  CHPID PATH=(CSS(0),01),SHARED,                               *
       PARTITION=((IB02PR03,IB02PR04),(=)),CPATH=(CSS(0),01),  *
       CSYSTEM=IB012097,AID=0E,PORT=1,TYPE=CIB
  CNTLUNIT CUNUMBR=FFDC,PATH=((CSS(0),01)),UNIT=CFP
  IODEVICE ADDRESS=(FD86,007),CUNUMBR=(FFDC),UNIT=CFP
```

Figure 5-43 IOCP for PSIFB Link between IB012097 and IB022097

Sample IOCP for timing-only PSIFB links

This sample lists the IOCP statements for the following PSIFB links between servers IB022097 and IB042097:

- IB022097 CHPID 53 on AID 1E PORT 1
- IB042097 CHPID 53 on AID 0C PORT 1

Example 5-44 shows the IOCP for each of these servers. Note that the Timing-only link is reflected by a CNTLUNIT type of STP for which no devices are generated.

```
IOCP Statements from IB022097:
 ID    MSG2='IB01.IODF34.WORK - 2007-11-09 16:45',             *
       SYSTEM=(2097,1),LSYSTEM=IB022097,                       *
       TOK=('IB022097',00800001DE5020971645116301073l3F00000000*
       ,00000000,'07-11-09','16:45:11','........','........')
 CHPID PATH=(CSS(0),53),SHARED,                                *
      PARTITION=((IB02PR03,IB02PR04),(=)),CPATH=(CSS(0),53),   *
      CSYSTEM=IB042097,AID=1E,PORT=1,TYPE=CIB
 CNTLUNIT CUNUMBR=FFD9,PATH=((CSS(0),53,54)),UNIT=STP
IOCP Statements from IB042097:
 ID    MSG2='IB01.IODF34.WORK - 2007-11-09 16:45',             *
       SYSTEM=(2097,1),LSYSTEM=IB042097,                       *
       TOK=('IB042097',00800001DE5020971645116301073l3F00000000*
       ,00000000,'07-11-09','16:45:11','........','........')
 CHPID PATH=(CSS(0),53),SHARED,PARTITION=((IB04SDM),(=)),      *
       CPATH=(CSS(0),53),CSYSTEM=IB022097,AID=0C,PORT=1,       *
       TYPE=CIB
CNTLUNIT CUNUMBR=FFDA,PATH=((CSS(0),53,54)),UNIT=STP
```

Figure 5-44 IOCP for Timing only PSIFB link between IB022097 and IB042097

5.8 CHPID Mapping Tool support

When planning and configuring a System z server, you must plan for maximum server and device availability. The CHPID Mapping Tool (CMT) provides an availability mapping function to assign CHPIDs across control units and avoid such single points of failure.

Full documentation is available within the CHPID Mapping Tool. Full usage guidelines are available in *z10 EC Configuration Setup*, SG24-7571.

Support for PSIFB links

The CMT provides the capability to identify potential availability exposures attributable to the mapping of CHPIDs across Adapter IDs and PORTs. To do so, follow the process shown in Figure 5-45 on page 172.

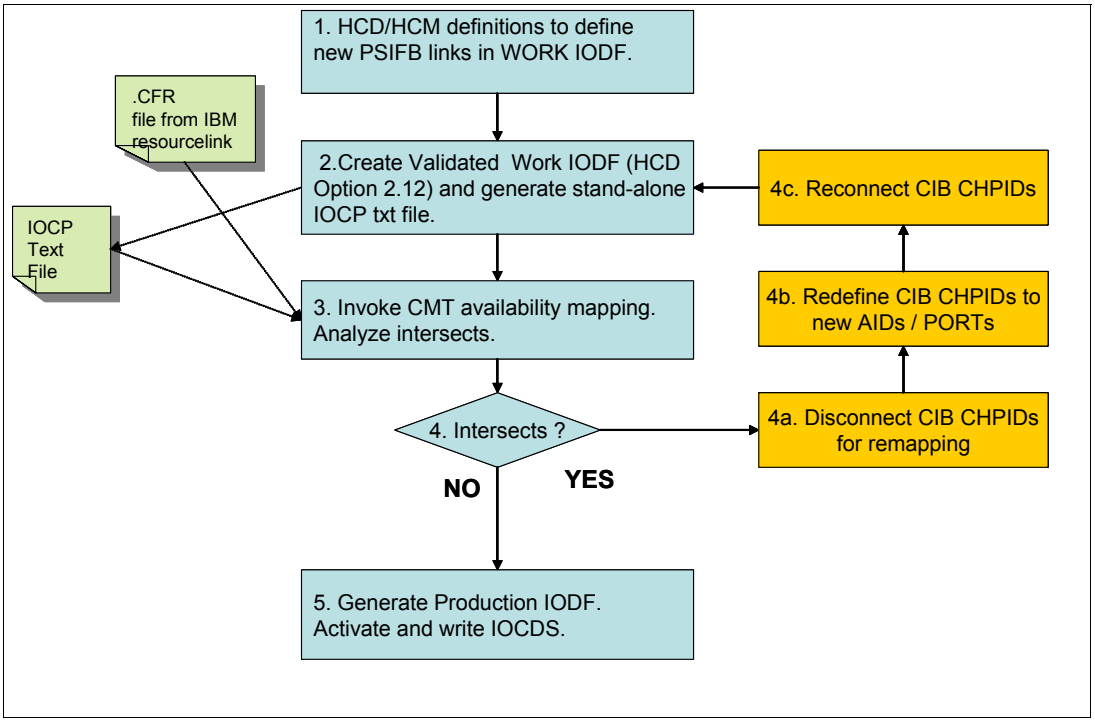


Figure 5-45 CHPID Mapping Tool process for PSIFB links

This section explains the stages shown in Figure 5-45.

Stage 1

Use HCD or HCM to define and connect the CIB CHPIDs to create the PSIFB links. For reference, see “Sample configuration with PSIFB links” on page 152.

Stage 2

Before you generate a stand-alone IOCP file, generate a Validated Work IODF using HCD option 2.12, as shown in Figure 5-46 on page 173.

```
      Activate or Process Configuration Data


Select one of the following tasks.

12  1.  Build production I/O definition file
    2.  Build IOCDS
    3.  Build IOCP input data set
    4.  Create JES3 initialization stream data
    5.  View active configuration
    6.  Activate or verify configuration
        dynamically
    7.  Activate configuration sysplex-wide
    8.  Activate switch configuration
    9.  Save switch configuration
    10. Build I/O configuration statements
    11. Build and manage S/390 microprocessor
        IOCDSs and IPL attributes
    12. Build validated work I/O definition file
```

Figure 5-46 HCD: Building the Validated Work IODF

IOCP file

Next, download the IOCP file to the workstation where the CHPID Mapping Tool will be run. Download the file as TEXT using your 3270 emulation program.

Stage 3

Invoke the CHPID Mapping Tool, load the CFR, and import the IOCP file. Perform availability mapping. (You can also use the Hardware Configuration Manager (HCM).)

Stage 4

Analyze the output for intersects across Adapter IDs and PORTs. You can display the intersects by using the CHPID Mapping Tool as shown in the CMT panel sample in Figure 5-47 on page 174, where `C` indicates that two channels use the same channel card.

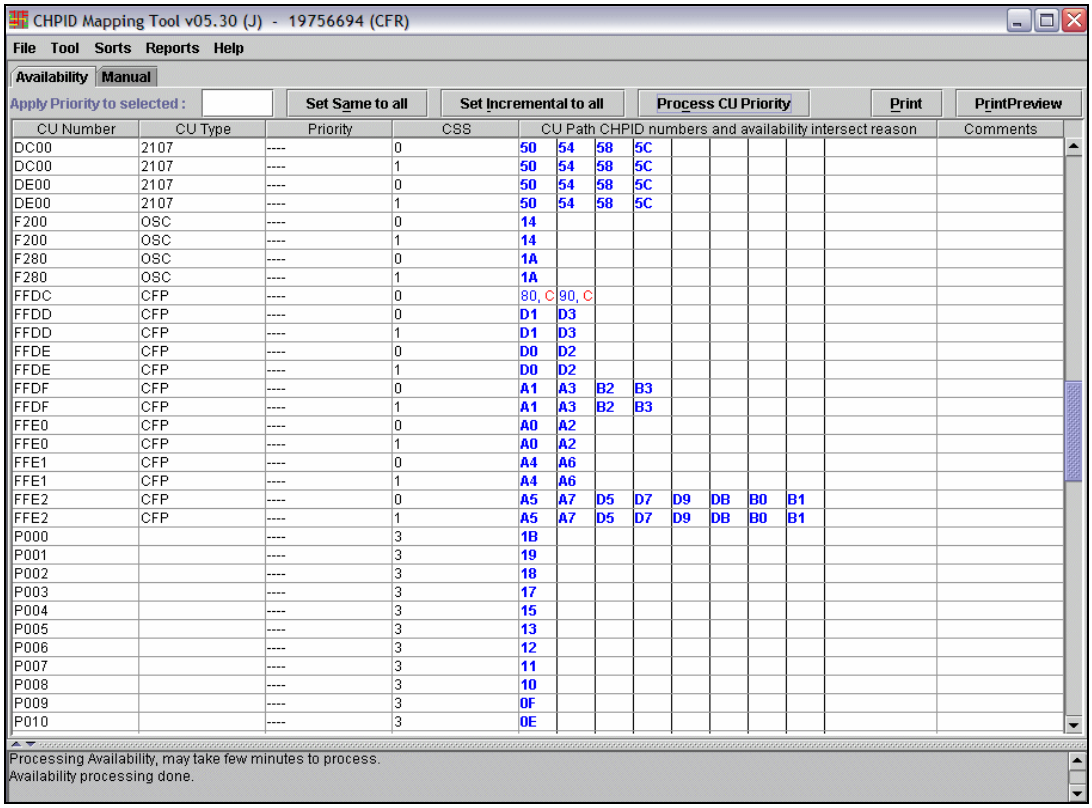


Figure 5-47 Sample CMT Panel showing detected intersects

If no unacceptable intersects exist, go to “Stage 5” on page 175. If there are such intersects, then follow steps 1 through 3.

1. Disconnect the CIB CHPIDs that are to be remapped. Perform this task from the CF Channel Path Connectivity List shown in Figure 5-48 on page 174. Type `n` next to each CHPID to be disconnected.

```
                  CF Channel Path Connectivity List               Row 1 of 4
Command ===> ___________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : IBO22097
Source channel subsystem ID . : 0
Source partition name . . . . : *

  -------Source--------   -----------Destination--------   -CU-
/ CHPID  Type  Mode Occ   Proc.CSSID   CHPID  Type  Mode    Type
n 01     CIB   SHR  N     IBO12097.0   01     CIB   SHR     CFP
_ 02     CIB   SHR  N     IBO12097.0   02     CIB   SHR     CFP
_ 53     CIB   SHR  N     IBO42097.0   53     CIB   SHR     STP
_ 54     CIB   SHR  N     IBO42097.0   54     CIB   SHR     STP
```

Figure 5-48 CF Channel Path Connectivity List

2. Redefine the CHPIDs across new Adapter IDs and PORTs; refer to “Defining PSIFB links” on page 156 for more detail.

3. Reconnect the CIB CHPIDs from the CF Channel Path Connectivity List; refer to “Connecting the CIB CHPIDs” on page 159 for more detail. To revalidate the configuration for intersects, return to Stage 2.

Stage 5

After you remove all the unacceptable intersects, you can build the production IODF. Use HCD to perform dynamic reconfiguration to activate the IODF and write the IOCDS.

5.9 Operations

During operations, some commands display a new CHPID type of CIB along with the following new messages:

```
IOS500I 03130 REASON=&NUM.,CANNOT CHANGE THE CONNECTED SYSTEM NAME FOR CIB
CHPID &XX. IN CSS &CC.
IOS500I 03131 NOTE = &NUM., CIB PORT OR ADAPTER NOT INSTALLED FOR CHPID &XX.
IOS500I 03132 NOTE = &NUM.,THE CONNECTED SYSTEM NAME WILL BE CHANGED FOR CIB
CHPID &XX. IN CSS &CC.
IOS500I 04080 DESCTEXT=THE CONNECTED SYSTEM (CSYSTEM) NAME SPECIFIED IS
INCORRECT
IOS500I 04081 ESCTEXT= CIB CSYSTEM NAME CHANGES MAY AFFECT FUTURE CONNECTIVITY
```

5.9.1 Summary

As a summary, Figure 5-49 on page 175 illustrates the z10 EC Parallel Sysplex coexistence and coupling connectivity.

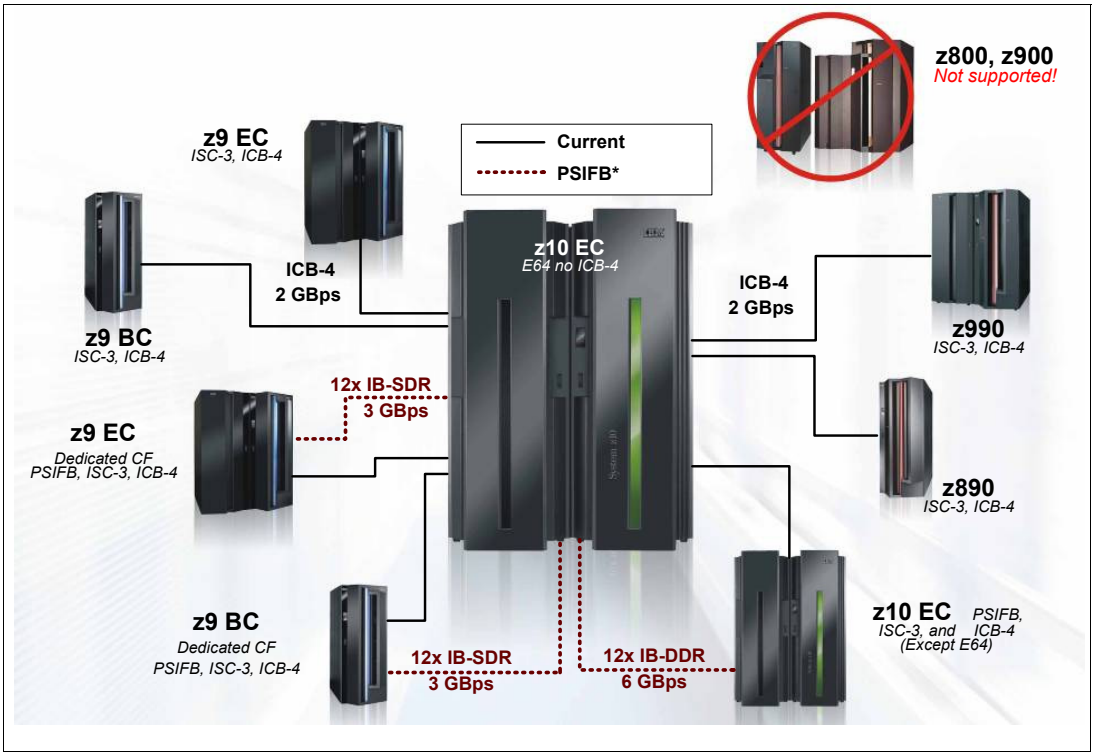


Figure 5-49 z10 EC Parallel Sysplex coexistence and coupling connectivity

6

z/OS Capacity Provisioning

z/OS Capacity Provisioning allows installations to manage processing capacity more reliably, more easily, and faster. Replacing manual monitoring with autonomic management, or supporting manual operation with recommendations can ensure that sufficient processing power will be available with the least possible delay. With the z/OS Capacity Provisioning function, it is possible to manage processing capacity via policy-based automation.

The activation of On/Off CoD on z10 EC can be simplified or automated by using z/OS Capacity Provisioning (available with z/OS V1R10 and z/OS V1R9). This capability enables the monitoring of multiple systems based on Capacity Provisioning and Workload Manager (WLM) definitions. When the defined conditions are met, z/OS can suggest capacity changes for manual activation from a z/OS console or the system can add or remove temporary capacity automatically and without operator intervention.

This chapter describes the following:

- z10 EC and Capacity Provisioning
- Capacity Provisioning domain
- Workload condition
- Capacity Provisioning hardware requirements
- Capacity Provisioning software requirements
- Capacity Provisioning manager summary

6.1 z10 EC and Capacity Provisioning

z/OS Workload Manager (WLM) manages the workload by goals and business importance (as defined by an installation WLM policy) on each z/OS system in a Parallel Sysplex. WLM metrics (as resource delays and performance index) are available through existing interfaces and are reported through RMF Monitor III, with one RMF gatherer per z/OS system.

Prior to z/OS V1R9, WLM controls, through the IRD feature, the distribution of physical CPUs among the z/OS systems contained in an LPAR cluster. This task was executed by the following WLM functions:

- WLM vary logical CPU management, where the number of logical CPUs in a LP is dynamically controlled by WLM
- WLM vary weight management, where, depending on whether goals are being achieved or not, the LP weights are dynamically modified taking CPU resource from an LP and delivering it to others in the same server.

With the z10 EC provisioning capability combined with the Capacity Provisioning Management (CPM) component in z/OS, it is possible to control the activation of On/Off Capacity on Demand in a new flexible and automated process.

Previously, WLM was only able to take CPU from an LP and give it to other LPs to decrease the CPU delays, causing important unhappy transactions. Now the CPM (WLM is included in such function) is able to activate spare CPUs through On/Off Capacity on Demand to fix the previously described situation.

Background

Unexpected workload spikes may exceed available capacity such that Service Level Agreements cannot be met. While business needs may not justify a permanent upgrade, a temporary upgrade may be justified. The z10 EC provides improved and integrated On/Off CoD and CBU concepts:

- Faster activation and improved robustness
- Can be partially activated and combined

The new Capacity on Demand architecture offers increased flexibility and capabilities over previous systems. Unlike previous systems, where only one temporary entitlement record (TER) could be active at a given time, with the new Capacity on Demand architecture, you can have up to four different TERs active at the same time. In addition, the new architecture allows for concurrent permanent upgrades while temporary capacity is active, as follows:

- System z Capacity Provisioning allows customers to manage processing capacity more reliably, more easily, and faster.
- Replacing manual monitoring with autonomic management, or supporting manual operation with recommendations can ensure that sufficient processing power will be available with the least possible delay.
- Demonstrates the superior vertical scalability of System z.

Capacity Provisioning infrastructure overview

The z/OS provisioning environment with all its components is shown in Figure 6-1.

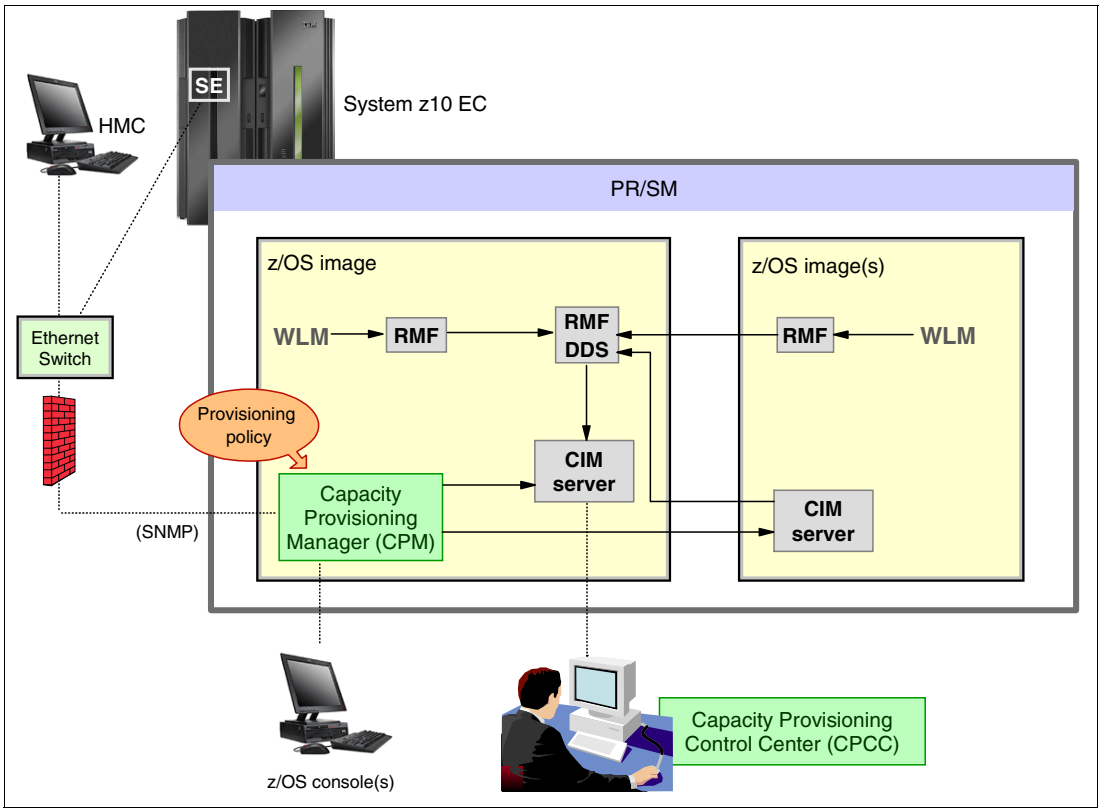


Figure 6-1 z/OS Capacity Provisioning

WLM and RMF support

WLM manages by goals and makes its metrics available through existing interfaces, as follows:

- One RMF gatherer per z/OS system.
- The RMF distributed data server (DDS) per sysplex. From the distributed data server (DDS) facility where all data is captured, RMF can send data to an RMF focal point.
- The RMF Common Interface Module (CIM) providers and associated CIM models publish the RMF Monitor III data.

  CIM is a standard data model developed by a consortium of major HW/SW vendors (including IBM) called Distributed Management Task Force (DMTF), a part of the Web Based Enterprise Management (WBEM) initiative. It includes a set of standards and technologies that provide management solutions for distributed network environments. CIM creates an interface (API) where applications, for example, can ask system management questions such as: how many jobs are running in the system? This query must be converted to an API known by the running operating system. In z/OS CIM is implemented through the Common Event Adapter (CEA) address space.
- Then the Capacity Provisioning Manager (CPM), a function inside z/OS, retrieves critical metrics from one or more z/OS systems through the Common Information Model (CIM) structures and protocol. Depending on such metrics, CPM communicates to (local or remote) support elements and HMCs, respectively, via the SNMP protocol to access On/Off Capacity on Demand. The control over the Provisioning Infrastructure is executed

by the CPM through Capacity Provisioning Policy (CPP), which controls the Capacity Provisioning Domain (CPD).

- Capacity Provisioning Policy (CPP) is created and managed by the Capacity Provisioning Control Center (CPCC), which resides on a workstation providing a system programmer front end to administer such policies. CPCC is a Graphical User Interface (GUI) component. These policies are not the ones managed by WLM and kept in the WLM couple data set. The CPCC is not required for regular CPM operation.
- Refer to “Capacity Provisioning domain” on page 181 to learn about CPD.

Capacity Provisioning policy

A Capacity Provisioning policy may consist of multiple rules based on a variety of things, such as specific applications (bank transactions, for example).

The “maximum provisioning scope” defines the maximum additional capacity (expressed in MSUs, zIIPs, and zAAPs) that may be activated at any time for all contained rules.

“Provisioning Condition” is simply a group of time and workload conditions that can be referred to as:

- WLM service class conditions
- Time conditions (start/deadline/end)
- Workload (critical workload conditions)

“Provisioning Scope” defines the maximum capacity (expressed in MSUs, zIIPs, zAAPs) that may be activated.

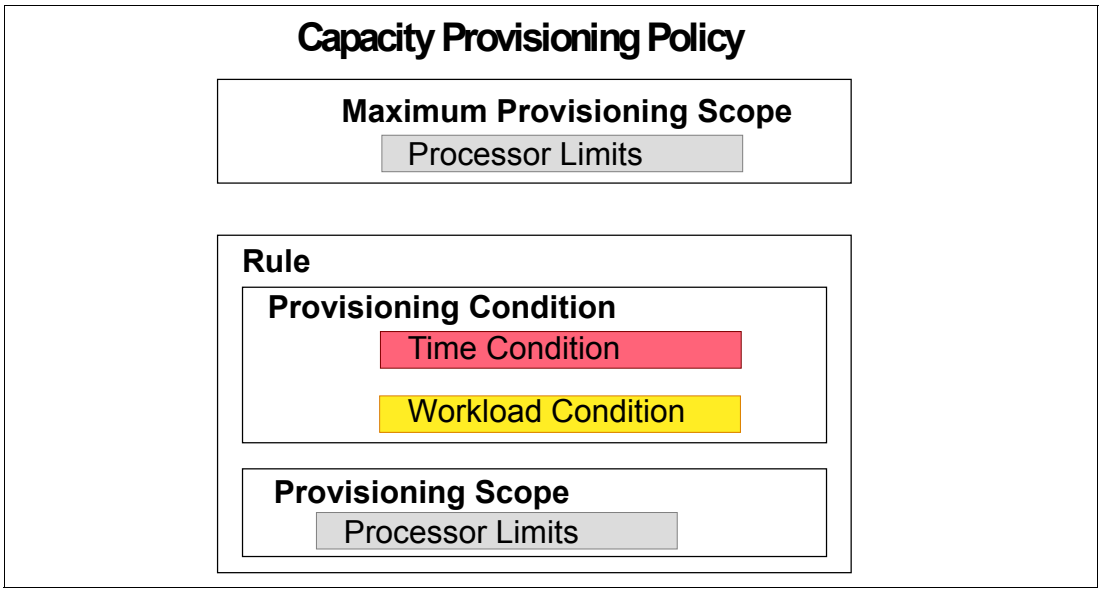


Figure 6-2 Capacity Provisioning Policy

Capacity management

CPM differentiates between types of Provisioning Requests:

- Manual through HMC or CPM (via command)
- Scheduled (time condition without workload condition)
- Conditional (based on workload condition)

Different capacity demands can be expressed as number of zAAPs, number of zIIPs, or general purpose capacity:

- Needs to consider different capacity levels (processor speeds) for sub-capacity processors
- Speed demand for higher capacity levels
- Unqualified demand (capacity level or number of GCPs)

6.1.1 Components of capacity provisioning

Capacity provisioning mainly consists of the following components:

- The Capacity Provisioning Manager (CPM) is the server program that monitors the defined systems and CPCs and takes actions as appropriate and authorized by the policies.
- The Capacity Provisioning Control Center (CPCC) is the Graphical User Interface (GUI) component. It is the interface through which administrators work with provisioning policies and domain configurations.

  Optionally, you can use the CPCC to transfer provisioning policies and domain configuration files to the CPM, or to query the Capacity Provisioning Manager status.

  The CPCC is installed and used on a Microsoft Windows workstation. It is not required for regular operation of the CPM.

6.2 Capacity Provisioning domain

Capacity Provisioning Domain (CPD) represents the set of servers (CECs) that are controlled by the Capacity Provisioning Manager; see Figure 6-3 on page 182. The HMCs of the CPCs within a CPD must be connected to the same processor LAN. Parallel Sysplex members can be part of a CPD. There is no requirement that all z/OS members of a Parallel Sysplex must be part of the CPD, but participating z/OS members must all be part of the same CPD.

Administrators work through the CPCC interface to define domain configurations and provisioning policies, but this is not needed during production. The CPCC is installed on a Microsoft Windows workstation.

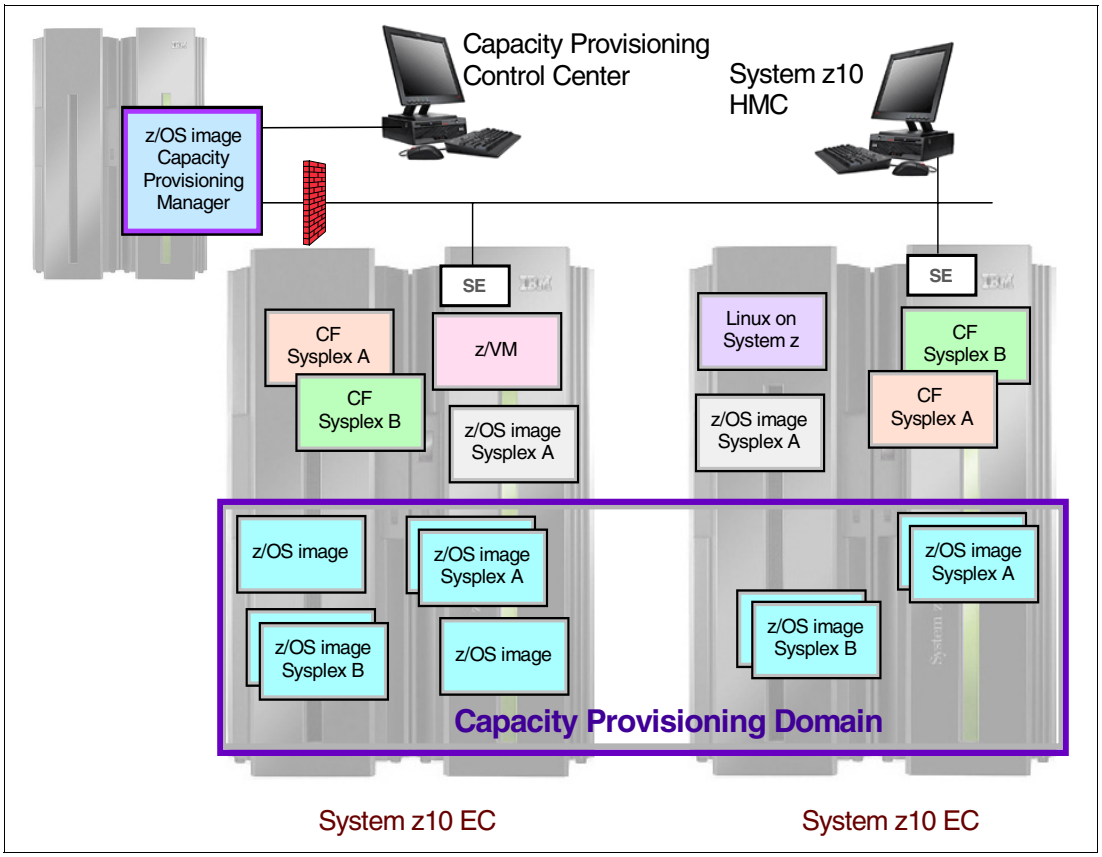


Figure 6-3 Capacity Provisioning Domain

CPM processing modes

The CPM operates in either of the following four modes, allowing for different degrees of automation:

Manual This mode is basically a command-driven mode where no CPM policy is active.

Analysis With this mode, CPM processes the capacity provisioning policy and informs the operator when a provisioning or deprovisioning action would be due according to the criteria specified in the policy. No checking whether On/Off CoD record is available

It is up to the operator either to ignore that information or to perform the up/downgrade manually (using the HMC/SE or the available CPM commands).

Confirmation With this mode, CPM processes the policy as well as the On/Off CoD record to be used for capacity provisioning. Every provisioning action needs to be authorized (confirmed) by the operator

Autonomic This mode is similar to confirmation mode, except that no human (operator) intervention is required.

In all modes, various reports will be available with information about workload and provisioning status, and the rationale for provisioning recommendations.

CPM commands

A CPM command user interface is through the z/OS system console. It is not possible to change the configuration and rules from a console. So CPCC and the console have no common commands and the CP Control Center has only partial use of commands.

The CPCC defines the configuration, policy, and rules and can upload them to CPM. To activate them, it is necessary to use an MVS console to run a command. The CPM server uses three types of input data sets that contain different types of information, as follows:

- The domain configuration defines the topology and connections, such as the CPCs and z/OS systems that are to be managed by the server.
- The policy contains the information as to:
  - Which work is provisioning eligible, under which conditions and during which time frames
  - How much capacity may be activated when the work suffers due to insufficient processing capacity
- The PARM data set contains setup instructions using UNIX environment variables and various processing options that may be set by an installation.

CPM policy

The provisioning policy defines the circumstances under which additional capacity may be provisioned. There are three elements in the criteria:

- **When** provisioning is allowed - a time condition:
  - Start time - indicates when provisioning can begin.
  - Deadline - provisioning of additional capacity no longer allowed.
  - End time - deactivation of additional capacity should begin.
- **Which** work qualifies for provisioning, parameters included - a workload condition:
  - The z/OS systems that may execute eligible work
  - Importance filter - eligible service class periods, identified by WLM importance
  - Performance Indicator (PI) criteria:
    - Activation threshold - the PI of service class periods must exceed the activation threshold for a specified duration before the work is considered to be suffering.
    - Deactivation threshold - the PI of service class periods must fall below the deactivation threshold for a specified duration before the work is considered to be no longer suffering.
  - Included service classes - eligible service class periods
  - Excluded service classes - service class periods that should not be considered
- **How much** additional capacity may be activated - provisioning scope. Specified in MSUs, number of zAAPs, and number of zIIPs - one specification per CPC that is part of the Capacity Provisioning domain.

Capacity Provisioning manager rules

The following rules apply to the CPM:

- The domain configuration defines CPCs and z/OS systems that are controlled by a CPM instance.
- Sysplexes do not have to be completely contained in a domain but must not belong to more than one domain.

- Multiple sysplexes and hence multiple WLM service definitions may be involved.
- One active Capacity Provisioning Policy (CPP) per Domain at a time.
- More than one policy can exist for different purposes.

6.3 Workload condition

The workload condition identifies the work that may trigger the activation of additional capacity when that work does not achieve its goal due to insufficient capacity, and additional capacity would help. Its parameters are:

- Sysplex systems - The z/OS systems that may run eligible work
- Importance Filter - Eligible service class periods, identified by WLM importance
- PI criteria:
  - Activation threshold - The PI of service class periods must exceed the activation threshold for a specified duration the work is considered to require help.
  - Deactivation threshold - The PI of service class periods must fall below the deactivation threshold for a specified duration the work is considered to no longer require help.
- Included service classes - Eligible service class periods.

  Extends the set of service class periods with qualified work (extends the default set of default eligible service classes) and may specify different PI criteria.
- Excluded service classes - Identifies service class periods that should not be considered.
- If specifications exist on multiple levels then the service class periods, as derived from the importance filter, are merged with the explicitly defined (included) service class period. Finally, the excluded service class periods (if any) are removed from the previous set.

If no workload condition is specified, the full capacity is activated and deactivated unconditionally at the start and end times of the time condition (scheduled activation, deactivation).

6.4 Capacity Provisioning hardware requirements

The following hardware is required in order to properly use Capacity Provisioning:

- One or more z10 EC servers.
- If temporary capacity should be controlled by the Capacity Provisioning Manager, when the manager is running in the confirmation or autonomic mode, or if provisioning actions are performed through CPM commands in either mode, temporary capacity needs to be available.

  It requires the CIU enablement feature and On/Off CoD enablement as well as a valid On/Off CoD record for temporary general purpose processor, zAAP, or zIIP capacity.
- The workstation for the control center needs:
  - INTEL Pentium® or equivalent processor with 512 MB of memory (1 GB recommended).
  - Available disk space of 150 MB.

– Microsoft Windows XP Professional, Service Pack 2 or later. Screen resolution should be 1024x768 or higher.

Capacity Provisioning configuration dependencies and restrictions

While *observed* systems must be running z/OS V1R9 or higher it is allowable that other operating systems or the Coupling Facility Control Code (CFCC) are active in other LPARs. Consider the following situations:

- Observed systems that are running as guests under z/VM are not supported.

  Also, the Capacity Provisioning Manager should run on a system that is not running as a z/VM guest.
- An observed system may run in a shared or dedicated LPAR. An LPAR with dedicated processors can only generate demand for a higher general purpose processor capacity level. If the processor is not a sub-capacity processor, that is. it is already operating at its maximum capacity level, no additional demand will be recognized.
- Also, for a dedicated LPAR, no demand for additional special purpose processors will be recognized.
- Demand for additional physical processors—as opposed to increased capacity level—for a shared CP, zAAP, or zIIP processor can only be recognized if the current sum of logical processors is greater than or equal to the target number of physical processors in the respective pool.

  Rationale: Capacity Provisioning currently does not configure reserved or offline processors online.
- Observed systems may have general purpose CPs, zAAPs zIIPs, or any combination thereof, configured. Other processor types in the physical configuration are allowable.
- Demand for zAAP processors can be recognized if at least one zAAP is already online to the system.
- Demand for zIIP processors can be recognized if at least one zIIP is already online to the system.
- The additional physical capacity will be distributed through PR/SM™ and the operating system. In general, the additional capacity will be available to all LPARs.

  Facilities such as defined capacity (soft capping) or initial capping (hard capping) can be used to control the use of capacity.
- IBM recommends to avoid defining provisioning conditions for service classes that are associated with WLM resource groups for which a capacity maximum is in effect.

The following are prerequisites for installation:

- z/OS RMF must be set up and customized (including Distributed Data Server (DDS)).
- z/OS CIM Server must be set up (z/OS Base element since z/OS V1R7).

Customization of capacity provisioning must be performed as described in Chapter 3 of *z/OS MVS Capacity Provisioning User's Guide*, SA33-8299.

This customization is required on the following systems:

- The observed z/OS systems - These are the systems in one or multiple sysplexes that are to be monitored (see description of Capacity Provisioning Domain).
- Runtime systems - These are the systems where the Capacity Provisioning Manager is running, or to which the server may fail over after server or system failures.

6.5 Capacity Provisioning software requirements

The following software is required:

- Systems with z/OS V1rr9 plus APAR OA20824 or above can be monitored or used to run the Capacity Provisioning Manager.
- z/OS Resource Measurement Facility (RMF), an optional element of z/OS, must be enabled (or an equivalent product, including equivalent CIM RMF provider capability).
- The z/OS security product needs to support creation of passtickets (R_GenSec) and evaluation through the SAF interfaces.
- When using a security product other than IBM Security Server (Resource Access Control Facility, RACF), check with your vendor.
- TCP (SNMP) connectivity from the hosting system (where the CPM server is running) to the Hardware Management Console or support elements, respectively.
- IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N98); currently, no other levels are supported.
- Capacity Provisioning utilizes the CIM Server which is a z/OS base element.

6.5.1 Installation

Installation of the Capacity Provisioning function on z/OS requires the following:

- Setting up and customizing z/OS RMF, including the Distributed Data Server (DDS).
- Setting up the z/OS CIM Server (included in z/OS base since V1 R7).
- Perform Capacity Provisioning customization as described in *z/OS Capacity Provisioning User's Guide*, SA33-8299.

Exploitation of the Capacity Provisioning function requires:

- TCP/IP connectivity to observed systems
- RMF Distributed Data Server must be active
- CIM Server must be active
- Security and CIM customization
- Capacity Provisioning Manager customization

In addition, the Capacity Provisioning Control Center has to be downloaded from the host and installed on a PC server. This application is only used to define policies, it is not required for regular operation.

Customization of the capacity provisioning function is required on observed z/OS systems. These are the systems in one or multiple sysplexes that are to be monitored. Runtime systems are the systems where the Capacity Provisioning Manager is running, or to which the server may fail over after server or system failures.

6.6 Capacity Provisioning manager summary

Capacity Provisioning provides faster, more reliable methods to activate On/Off Capacity on Demand.

The following are quite useful features of this new component with z/OS V1R10:

- Manual mode allows activation and deactivation of physical general purpose capacity, zAAPs, and zIIPs.
- Analysis mode to receive suggestions on how to address capacity bottlenecks.
- Confirmation and automation mode to automate recognition and enabling of capacity changes with or without operator confirmation.
- Can help optimize the activated resources.
- Solution integrates z10 EC enhancements, as well as new and existing z/OS components: Capacity Provisioning FMID, WLM, RMF, CIM.
- Different degrees of automation available, ranging from manual to autonomic.

7

New features of XL C/C++

This chapter describes the new features of XL C/C++ for z/OS V1R10.

These are application enablement features:

- Mixing AMODE31/64 under Metal C
- #pragma insert_asm
- Saved Option String
- C++ TR1 Libraries
- SQL data type initialization macros
- DFP support for SQL host variable
- Copyright CSECT
- Function offset in pseudo-assembly listing

These are performance features:

- Improve C++ Standard Library Performance
- Prefetch Built-in Functions
- Improve dbx Start-up Time

7.1 Application enablement features

The C/C++ compiler is not capable of AMODE mixing primarily due to its dependency on LE's storage management facility (64-bit LE control blocks are not accessible by ILP32 programs, and vice versa).

With the METAL option, the storage locality is entirely managed by the user code.

- Stack frame is obtained by user-supplied prolog code.
- The compiler assumes that it is the user's responsibility to ensure that the addressability of the storage is consistent with the AMODE switches.

The current compiler support is limited to calling an external function defined in a separate CU. C/C++ does not support mixing AMODE31 and AMODE64 compilation units in a single application, while HLASM provides facilities to switch between AMODE31 and AMODE64.

z/OS V1R10 support

With z/OS V1R10 Metal C, a user can create LE-independent code via the METAL option. This feature instructs the compiler to generate AMODE switching code whenever necessary, and so the user can now call external LP64 functions from ILP32 functions, and vice versa.

The compiler needs to know when to generate AMODE switching code. New attributes are necessary for identifying the AMODE of the functions:

- Defining LP64 functions in an ILP32 compilation unit is not allowed (and vice versa).
- Dereferencing LP64 pointers in an ILP32 compilation unit is not allowed (vice versa is okay).

The new function and type attributes are `__attribute__((amode31))` and `__attribute__((amode64))`; they are specified as follows:

```
void foo (void) __attribute__((amode64));
typedef void (*fp)(void) __attribute__((amode31));
```

There is a new pointer qualifier in Metal C, `__ptr64`, specified as follows:

```
int *__ptr64 p;
```

AMODE switching support

To take advantage of the Metal C AMODE switching support, be aware of the following:

- The called and calling programs must be in separate source files. Mixing addressing modes within a single C source file is not supported.
- The compiler adjusts the save area size of the calling program to accommodate the requirements of the called program.
- Before the call, the compiler switches the addressing mode of the calling program to that of the called program. On return from the call, the compiler restores the original addressing mode of the calling program.
- The implicit sizes of types *long* and *pointer* are determined by the addressing mode of the target program.

Coding with AMODE support

Figure 7-1 shows the contents of two C source files, one for each addressing mode:

- initval64() creates a 64-bit int pointer that holds the value of 5.
- getval64() reads the 64-bit int pointer and returns the value that it holds.
- main() is an AMODE31 function that calls the above functions to get the content of the 64-bit int pointer.

amode31.c:

```
int *__ptr64 initval64(void) __attribute__((amode64));
int getval64(int *__ptr64)  __attribute__((amode64));

int main ()
{
  int *__ptr64 p;
  p = initval64();
  return getval64(p);
}
```

amode64.c:

```
int val;

int *initval64(void) {
  val = 5;
  return &val;
}

int getval64(int *p) {
  return *p;
}
```

Figure 7-1 AMODE31 program that calls an AMODE64 program

Compiling source code

Figure 7-2 on page 192 is a z/OS UNIX OMVS shell session. It shows the successive commands that need to be issued. A description of the commands follows:

- The first command compiles file amode31.c and generates an HLASM source file from the C source file. Issuing the **xlc** command must be invoked with the **-qmetal** option and the **-S** flag, as follows:

  ```
  xlc -S -qmetal amode31.c
  ```

- The next command does the equivalent for amode64.c, which contains the two functions called by the main program contained in amode31.c. Notice that the **-q64** flag has to be specified in order for the compiler to know that the functions in amode64.c are to be running in AMODE=64.

  ```
  xlc -q64 -S -qmetal amode64.c
  ```

- The next two commands assemble the HLASM source files generated by the C compiler in the two previous commands:

  ```
  as amode31.s
  as amode64.s
  ```

- The next command does a link-edit of the two object files generated by HLASM and generates an executable for which a MAIN entry point is specified:

  ```
  ld amode31.o amode64.o -eMAIN
  ```

- The next command executes the program:

  ```
  a.out
  ```

- Execution is followed by the determining the returned value using the **echo** command, shown as 5. This shows that we have actually been in the AMODE=64 world from the MAIN, itself running in AMODE=31, and back. The command is:

```
     echo $?
```

```
LAFITTE @ SC70:/u/lafitte>
LAFITTE @ SC70:/u/lafitte>ls
amode31.c    amode64.c
LAFITTE @ SC70:/u/lafitte>xlc -S -qmetal amode31.c
LAFITTE @ SC70:/u/lafitte>xlc -q64 -S -qmetal amode64.c
LAFITTE @ SC70:/u/lafitte>as amode31.s
 Assembler Done No Statements Flagged
LAFITTE @ SC70:/u/lafitte>as amode64.s
 Assembler Done No Statements Flagged
LAFITTE @ SC70:/u/lafitte> ld amode31.o amode64.o -eMAIN
LAFITTE @ SC70:/u/lafitte>a.out
LAFITTE @ SC70:/u/lafitte>echo $?
5
LAFITTE @ SC70:/u/lafitte>
LAFITTE @ SC70:/u/lafitte>ls
a.out       amode31.o   amode64.c   amode64.s
amode31.c   amode31.s   amode64.o
```

Figure 7-2 Compile, assemble, and go

7.1.1 #pragma insert_asm

Some macro invocation requires the use of an accompanying macro to map out the storage of the parameter area. These “mapping macros” are typically placed at the end of the assembler program.

__asm() places assembler statements within the function body, and has not been suitable for this purpose.

#pragma insert_asm accepts a user-specified string and the compiler inserts it to the end of the generated HLASM code.

Some macro invocations can now be used through this facility. The new #pragma now inserts the text string just before the END statement in the generated HLASM code.

7.1.2 Saved option string

Two common questions asked when diagnosing an application problem are:

- What are the compilation options used?
- What is the level of the compiler used?

Nowadays, this information can only be obtained via a combination source listing, build script and PHASEID information—when they can be found.

Saved option string (SOS) support is a compact representation of compiler options and PHASEID information that is always embedded in the object file. SOS is also present in the executable module. SOS is always generated (not an option). SOS is stored in the object in binary form. It can be located via the PPA2 block.

Consequently, it is now possible to diagnose some run-time problems with the SOS information. Debuggers can use this information to identify the compiler and options used for each compilation unit.

The information depicted in Figure 7-3 is available in the SOS within the executable module (the information is stored in binary format; the following text is generated by translating the binary information).

```
Compile Unit #1:
Timestamp:
00000000  f2f0f0f8 f0f3f0f4 f2f1f1f2 f4f8f0f1  |2008030421124801|
00000010  f0c1f0f0                             |0A00            |
Saved Option String:
sos_words         = 8
sos_version       = 1
sos_arch          = ARCH(5)
sos_tune          = TUNE(5)
sos_csect         = NOCSECT
sos_version_info  = 8
sos_locale_ccsid  = IBM-1047
sos_lit_ccsid     = 0x0000
sos_wlit_ccsid    = 0x0000
sos_target_rel    = 0x410A0000
sos_initauto_val  = 0x00000000
sos_enumsize      = ENUMSIZE(SMALL)
sos_round         = ROUND(Z)
...
CCNDRVR           = N20080304.zosdev
CCNEOPTP          = N20080304.zosdev
CCNEP             = N20080304.zosdev
CCNETBY           = D0303
```

Figure 7-3 Saved option string

7.1.3 C++ TR1 library support

The document “ISO/IEC TR 19768 -- Technical Report on C++ Library Extensions” (informally named “TR1”) specifies ten libraries to supplement the existing C++ standard library.

The following TR1 libraries are supported in z/OS V1R10 C++:

- Smart pointers: shared_ptr and weak_ptr
- Function objects
- Random number generators
- Tuples
- Fixed size arrays
- Regular expressions
- Unordered associative containers
- type_traits

The macro __IBMCPP_TR1__ must be defined to non-zero to gain access to TR1 library features.

New header files are added:

- <array>

- <random>
- <regex>
- <tuple>
- <unordered_set>
- <unordered_map>
- <type_traits>

New declarations are added to existing header files:

- <memory>
- <functional>
- <utility>

Two items are missing:

- Additions for C99 compatibility
- Special math functions

  This library will be the subject of a C Technical Report, and will therefore become a common C/C++ facility (because it is specified for both languages), rather than a C++-only library.

With the following notes:

- All TR1 libraries live in the std::tr1 namespace.
- TARGET(zOSV1R10) must be used (default TARGET). The header file will issue an error message if the required TARGET level is not achieved: `C++ TR1 libraries can only be used with z/OS V1R10 or above`.
- The type_traits and random libraries work on both IEEE and HEX floating point mode, but DFP mode is not yet supported.
- The regular expressions library works in both ASCII and EBCDIC.

7.1.4 SQL data type initialization macros

To initialize the DB2 data type (VARBINARY, BLOB, CLOB, DBCLOB), it is necessary to understand the underlying structure of the data types. However, the user should not need to be bothered with this kind of detail.

With the SQL option, the compiler can generate a set of functions such as macros to initialize the above mentioned data types; see Figure 7-4 on page 195.

The macros are already used in the iSeries®, Linux, UNIX, and Windows versions of the DB2 preprocessor. This feature makes porting from those platforms and operating systems easier. These macros are only predefined if the SQL option is specified. The macros are unprotected, meaning that users are free to undefine and/or redefine them.

```
EXEC SQL INCLUDE SQLCA;
int main(void) {
   EXEC SQL BEGIN DECLARE SECTION;
   SQL TYPE IS VARBINARY(100) myvar =
     SQL_VARBINARY_INIT("abc");
   SQL TYPE IS BLOB(100) myvar1 = SQL_BLOB_INIT("abc");
   SQL TYPE IS CLOB(100) myvar2 = SQL_CLOB_INIT("abc");
   SQL TYPE IS DBCLOB(100) myvar3 = SQL_DBCLOB_INIT(L"abc");
   EXEC SQL END DECLARE SECTION;

   return 55;
```

Figure 7-4 SQL data type initialization macros

The actual definition of the macros is:

```
#define SQL_VARBINARY_INIT(s) {sizeof(s)-1, s}
#define SQL_BLOB_INIT(s) {sizeof(s)-1, s}
#define SQL_CLOB_INIT(s) {sizeof(s)-1, s}
#define SQL_DBCLOB_INIT(s) {(sizeof(s)/2)-1, s}
```

7.1.5 DFP support for SQL host variable

SQL for DB2 V9 provides support for DFP types through the DECFLOAT data type. The C/C++ compiler provides DFP support in z/OS V1R9. However, the DFP type cannot be designated as DB2 host variables. Through the use of the DFP and SQL options, DFP type Identifiers can be designated as host variables and used in embedded SQL statements. The DFP data type is useful in financial applications where most calculations are performed using decimal arithmetic. This feature allows a user to write C/C++ applications with embedded SQL statements for DB2 databases containing DFP data.

This support is also available in z/OS V1R9 through compiler PTFs UK33668, UK33740, UK33741, and UK33742.

```
_Decimal64 dfp_id1;
EXEC SQL BEGIN DECLARE SECTION;
_Decimal64 dfp_hv;
EXEC SQL END DECLARE SECTION;
_Decimal64 dfp_id2;
```

The identifiers dfp_id1 and dfp_id2 cannot be used in embedded SQL statements as Host Variables. Only dfp_hv can be used as an SQL host variable. In an embedded SQL statement, users may then use the DFP host variable like any other host variable.

7.1.6 Copyright CSECT

A #pragma comment (copyright) inserts a user-specified text string into an object file. If multiple source files are compiled with the same pragma, the resulting executable module will contain multiple copies of the same text.

To avoid this, place all copyright statements into a single CSECT. The user can customize the name of the CSECT.

Putting the copyright statement into its own CSECT makes it easy to look for the copyright string.

If multiple CSECTs have the same name, the binder arbitrarily picks one for the bounded module. This reduces the size of the executable module where text is placed by the compiler into a CSECT section named csect_name in the generated object module.

```
#pragma comment (csect_copyright, csect_name, "text")
```

7.1.7 Function offset in pseudo-assembly listing

When the LIST option is active, the compiler displays a pseudo-assembly listing, and an offset is attached on each instruction.

When OFFSET is not active, it contains the relative offset from the top of the CSECT. When OFFSET is active, it contains the relative offset from the top of the function. But, so far, it has not been possible to display both at once. When OFFSET is active, each function reports its relative offset from the top of the CSECT. This information is attached at the end of each function.

Given this information, a user can easily calculate the CSECT relative offset for each instruction.

```
OFFSET OBJECT CODE      LINE# FILE#   P S E U D O   A S S E M B L Y   L I S T I N G


                 000001 |     *  int main ()
000000              000001 |      main    DS     0D
000000 47F0 F022      000001 |             B      34(,r15)
000004 01C3C5C5                             CEE eyecatcher
...

                 ***  General purpose registers used: 1001100000001111
                 ***  Floating point  registers used: 0000000000000000
                 ***  Size of register spill area: 128(max) 0(used)
                 ***  Size of dynamic storage: 224
                 ***  Size of executable code: 98
                 ***  CSECT Offset: 136 : 0x88
```

Figure 7-5 Function offset in a pseudo-assembly listing

7.1.8 Improve C++ Standard Library performance

While investigating SPECINT 2006 for xalancbmk, a performance improvement was spotted that can speed up the benchmark by 200% by defining a global placement new operator in the C++ header file <new>. Then code paths that make use of the global placement new are expected to experience a runtime performance gain.

Existing non-C++ standard compliant code may need to get the old behavior by specifying:

```
__IBM_ALLOW_OVERRIDE_PLACEMENT_NEW
```

Code that is considered a C++ implementation such as RogueWave C++ library will need to define the macro to enable defining global placement new, as follows:

```
__IBM_ALLOW_OVERRIDE_PLACEMENT_NEW
```

Code that is not considered a C++ implementation, such as Jikes, which defines global placement new is not standard-compliant and can get back the pre-z/OS V1R10 behavior by defining the same macro.

Placement new

Placement new makes it possible to construct an object or an array of objects at a predetermined memory position. It has many useful applications, including building a custom-made memory pool or a garbage collector. Additionally, it can be used in mission-critical applications because there is no danger of allocation failure; the memory that is used by placement new has already been allocated. Placement new is also faster because the construction of an object on a preallocated buffer takes less time.

7.1.9 Prefetch built-in functions

The z10 EC provides PREFETCH DATA and PREFETCH DATA RELATIVE LONG instructions. The C/C++ compiler does not insert these instructions automatically under ARCH(8). Built-in functions are available to allow C/C++ programmers to manually insert these instructions into their code.

These new instructions affect the prefetching/releasing of storage into/from the data or instruction cache. Generally, main storage may include one or more smaller and faster access buffers, sometimes called caches. A cache is usually physically associated with a CPU or I/O processor. The effects, except on performance , of the physical construction and use of distinct storage media are generally not observable by the program.

Separate caches may be maintained for instructions and for data operands. Information in a cache is maintained in contiguous bytes on an integral boundary called a cache block or cache line (or line, for short). A model may provide the EXTRACT CACHE ATTRIBUTE instruction which returns the size of a cache line in bytes. A model (such as the z10 EC) may also provide the PREFETCH DATA and PREFETCH DATA RELATIVE LONG instructions which effect the prefetching of storage into the data or instruction cache or the releasing of data from the cache. These instructions are introduced with the z10 EC model along with other functionalities to manage the polarization of the CPU topology.

These functions are made for programmers who want to exploit z10 EC instructions to improve application performance:

- void __dcbt (const void* addr);

  Prefetch the cache line containing the specified address into the cache for fetch access.

- void __dcbtst (const void* addr);

  Prefetch the cache line containing the specified address into the cache for store access.

- void __dcbst (const void* addr);

  Release the cache line containing the specified address from store access; retain the data in the cache line for fetch access.

- void __dcbf (const void* addr);

  Release the cache line containing the specified address from all accesses.

Important: #include <builtins.h> to access built-in functions.

More generally, ARCH(8) produces code that uses instructions available on the 2097-xxx models (IBM System z10 EC) in z/Architecture® mode. Specifically, these ARCH(8) machines and their follow-ons add instructions supported by the general instruction extension

facility, which may be exploited by the compiler. Also, these machines add instructions supported by the decimal floating-point facility, which are generated if the DFP compiler option is specified and there are decimal floating-point data types in the source code.

General-instructions-extension facility

The general-instructions-extension facility may be available on a model implementing z/Architecture. The facility provides the following new instructions:

- ADD LOGICAL WITH SIGNED IMMEDIATE
- COMPARE AND BRANCH
- COMPARE AND BRANCH RELATIVE
- COMPARE AND TRAP
- COMPARE HALFWORD RELATIVE LONG
- COMPARE IMMEDIATE AND BRANCH
- COMPARE IMMEDIATE AND BRANCH RELATIVE
- COMPARE IMMEDIATE AND TRAP
- COMPARE LOGICAL AND BRANCH
- COMPARE LOGICAL AND BRANCH RELATIVE
- COMPARE LOGICAL AND TRAP
- COMPARE LOGICAL IMMEDIATE AND BRANCH
- COMPARE LOGICAL IMMEDIATE AND BRANCH RELATIVE
- COMPARE LOGICAL IMMEDIATE AND TRAP
- COMPARE LOGICAL RELATIVE LONG
- COMPARE RELATIVE LONG
- EXTRACT CACHE ATTRIBUTE
- LOAD HALFWORD RELATIVE LONG
- LOAD LOGICAL HALFWORD RELATIVE LONG
- LOAD LOGICAL RELATIVE LONG
- LOAD RELATIVE LONG
- MULTIPLY SINGLE IMMEDIATE
- PREFETCH DATA
- PREFETCH DATA RELATIVE LONG
- ROTATE THEN AND SELECTED BITS
- ROTATE THEN EXCLUSIVE OR SELECTED BITS
- ROTATE THEN INSERT SELECTED BITS
- ROTATE THEN OR SELECTED BITS
- STORE HALFWORD RELATIVE LONG
- STORE RELATIVE LONG

Additionally, a certain number of instructions have been enhanced to include additional formats.

The various enhancements of IBM z/OS XL C/C++ are reported in Figure 7-6.

XLC Arch ( )	Hardware facility	Hardw. arch.	machine models	XLC min level	XLC option	Note
8	general instruction extensions facility	z/Arch	z10	1.10		
	AR-mode	31/64		1.9	METAL	no LE Envrt
	decimal floating point	z/Arch	z9-GA3	1.8	DFP	LANGLVL (EXTENDED)
7	extended-immediate facility	z/Arch	z9			x- xlation 3
6	long/displacement facility		z990 z890			x - xlation 2
5	64-bit mode	ESAME	z900 z800	1.2		default for TARGET(z/OS 7) and above
4	long long operations	ESA/390	z900 z800			
3	IEEE (binary) floating-pt		G5-G6			
2	Branch Relative and Halfword Immediate		(G2-G4)			
1	Logical String Assist		G1 9021			
0	produced code is executable on all models					

Figure 7-6 Overall enhancements of IBM z/OS XL C/C++

7.1.10 Improve dbx start-up time

When dbx debugs a large C/C++ application, it has to load all the debug side files before displaying the first user prompt. This process can take a significant amount of time, and must be experienced each time the debug session is restarted.

The new utility dbgld can be used to create a module level debug side file; see Figure 7-7 on page 200. dbx can use this information to load only the debug information that it needs, thereby significantly reducing the debugger start-up time.

One debug side file per module is maintained.

The dbgld utility has the following syntax:

```
dbgld [ option ] file
```

- `option` may be:
  - -v Write the version information to stderr.
- `file` is the module name, and may be:
  - The absolute path name of a z/OS UNIX file
  - The relative path name of a z/OS UNIX file
  - A fully qualified MVS data set (sequential data set or PDS member)

```
LAFITTE @ SC70:/u/lafitte>xlc -g hello.c
LAFITTE @ SC70:/u/lafitte>dbgld a.out
LAFITTE @ SC70:/u/lafitte>dbx a.out
FDBX0089: dbx for z/OS with 64-bit support.
FDBX0399: Compiled: Mary 20 2008 at 10:36:02 (v1.10)
FDBX0400: OS level: 10.00 01; LE level: 4.1.10; (Local)
FDBX6499: CDA levels: ELF=D0814.20080401, DWARF=D0814.20080401,
                      DDPI=D0814.20080401
FDBX0100: Type 'help' for help.
FDBX0048: Set an event like 'st in main' then use 'c' to start debugging.
FDBX6432: Processing load module "a.out"
FDBX6421: Loaded debug data from "/u/lafitte/hello.dbg"
FDBX0150: Debug target is 31-bit.

LAFITTE @ SC70:/u/lafitte>ls a.*
a.out       a.out.mdbg
LAFITTE @ SC70:/u/lafitte>
```

Figure 7-7 dbx start-up time enhanced

The output of the dbgld utility is a file with the name of the module followed by an .mdbg extension. The file will always be written in the current directory. For example, if the module name is /mypath/mymodule, a file called mymodule.mdbg will be created in the current directory. If the file already exists, it will be overwritten.

dbx requires z/OS V1R10 CDA runtime to process the module level debug side file.

8

Workload Manager

Current SRM enqueue promotion support is strictly efficiency based. There is no consideration given to the current Workload Manager (WLM) policy or the business importance of individual waiters for resources that are under contention. The current support also considers only GRS-managed resources. Known problems exist with DB2 and SRM where subsystem and occasionally system outages occur because critical work is waiting for types of resources such as the RACF database or DB2 locks via IRLM.

Typically, the resources needed by applications are being held by work that is simply not being dispatched. The resource holder may be of low importance, or may have been reset-quiesced by the operator.

This chapter describes the functional enhancements with WLM in z/OS V1R10:

- WLM and SRM resource contention
  - Changes to IWNCNTN service
  - RMF support
- WLM pageable storage management
- WLM resource delays

8.1 WLM and SRM resource contention

With z/OS V1R10, there are changes in enqueue management to integrate the importance of the waiters for the resource into the decision to reallocate CPU to the resource holder so that it can finish and release the resource. Awareness of contention situations in authorized resource managers other than GRS, such as DB2, becomes part of the change in this release. Over several releases, this problem has been addressed and in this release changes in enqueue management are made in the SRM resource contention model to resolve resource contentions.

Resource contention

Some contention can naturally be abnormal, lasting longer than is usual for the resource and/or resource manager. Here, some level of intervention may be warranted to ensure, for example, that resource holders are not blocked for long periods due to circumstances such as lack of CPU.

Unimportant work may be dispatched only occasionally on a busy system, yet eventually it may still require an important lock or latch. Once it has acquired that resource, it should have an advantage if unimportant work is treated well for a while. Ideally it will release the resource before more important work requires the same resource.

Abnormal contention may signal that a problem has occurred where either a resource manager or system experiences application deadlocks. Also, it could be a long sequence of database updates without a COMMIT or it may simply be that the resource holder is executing a large or long-running request. The base implementation of abnormal contention management is SYSEVENT ENQHOLD, which is used to identify the holder of a resource causing contention that SRM may boost the service to the resource holder to help resolve the contention more quickly. A holder can be either an address space or an enclave.

When the contention becomes chronic, this situation requires significant intervention. This includes the business importance of the work as a promotion criterion. Together, WLM and SRM try to promote the resource holder to the business goal of the most important waiter. This requires topology information in WLM and SRM about the holders and waiters for a resource.

To address this problem in this release, resource managers can now prevent subsystem and or system outages by resolving chronic resource contentions signaled to WLM. In addition, resource managers are now aware of contention deadlocks because of WLM and SRM knowledge of the complete contention.

8.1.1 WLM IWMCNTN service

The existing IWMCNTN service allows resource managers to notify WLM of changes to the list of resources, work units, or transactions involved with resources that have been in contention (waiters exist) for longer than a resource manager-defined interval. The interval should be chosen so that only contention which has lasted long enough to be considered chronic for the issuing resource manager results in calling this service. The key services provided are to do the following:

- Signal WLM about chronic resource contentions
- Maintain the internal resource topology
- Identify holders and waiters of resources
- Notify WLM of the changes to the list of resources and work units

IWMCNTN service call

Figure 8-2 on page 204 shows a chronic resource contention deadlock. A resource manager issues a IWMCNTN service call (Figure 8-1) to determine such a condition using the following input parameter information:

- Subsystem type and instance name to identify the resource manager
- Resource identifier to identify the resource under contention
- Invocation type:
  - Update - Apply the request list information to the resource topology
  - Replace - Same as update, except that any local resource topology is discarded first
  - EndOfContention - Discard topology information for this resource
- List of notification requests, where each entry is specifying:
  - A work unit (STOKEN, TCB, enclave token)
  - Whether the work unit is holding the resource or waiting for the resource
  - Whether the work unit is to be added to or be deleted from the resource topology

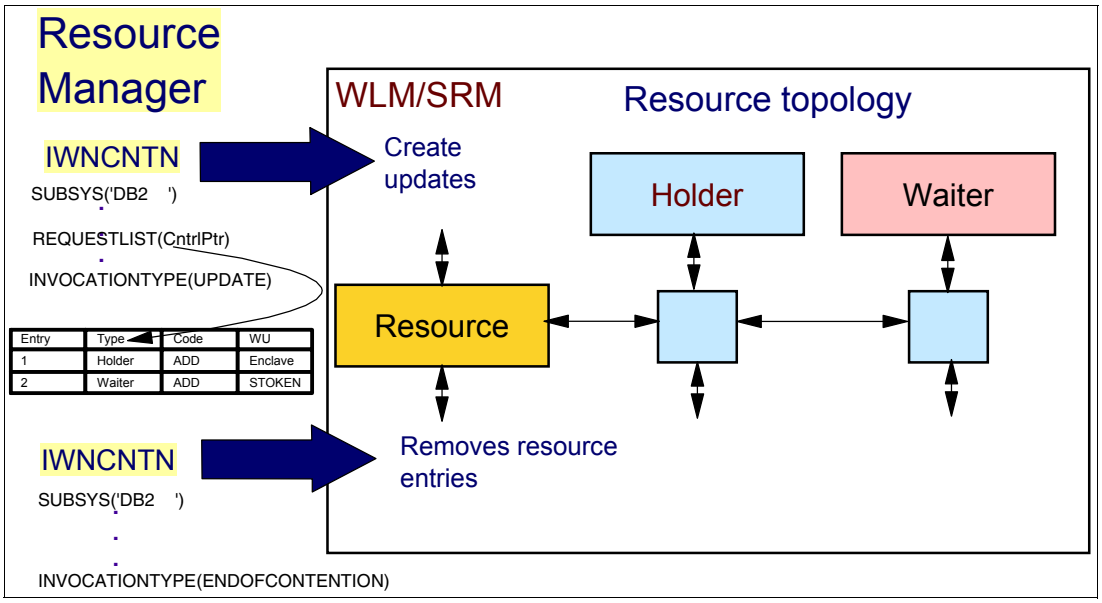


Figure 8-1 Resource manager using IWNCNTN

Note: The major exploiters of this new function are primarily DB2 and all potential resource managers.

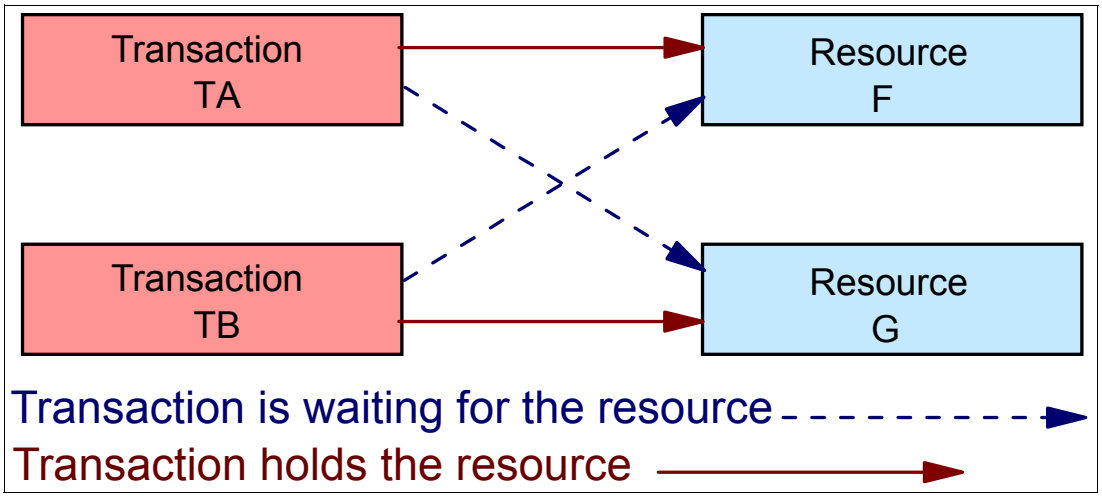


Figure 8-2 Chronic resource contention deadlock

New reason and return codes

Together, WLM and SRM reject chronic resource contention notifications that would cause deadlocks in the resource topology. The causing IWMCNTN notification is rejected with a return and reason code combination. The following return codes and reason codes are new with z/OS V1R10:

Return code **4** - Successful completion, unusual conditions noted.

Reason codes **0448** - The specified chronic resource contention may have caused a deadlock: The holder of resource (A) is waiting for resource (B), which is currently held by another holder, which is waiting for resource (A).

Return code **8** - Unusual conditions noted.

08A5 - The specified chronic resource contention is not stored in the topology. The DELETE request for this request list entry was not processed. **Action:** Correct the DELETE request.

08A8 - The specified chronic resource contention is already stored in the topology. The ADD request for this request list entry was not processed. **Action:** Correct the ADD request.

08AF - The specified chronic resource contention caused a deadlock. The holder of resource (A) is waiting for resource (B), which is currently held by another holder, which is waiting for resource (A). The request list entry was not processed. **Action:** Remove the deadlock.

8.1.2 New IEAOPTxx parameter

After analyzing the results of the resource manager IWNCNTN requests, WLM and SRM analyze the resource topology across resources and promote all resource holders that are blocking work, to the dispatch priority of the most important waiters.

A new IEAOPTxx parmlib member parameter, MaxPromoteTime, is introduced that limits the time that a resource holder is allowed to run promoted.

MaxPromoteTime Specifies the time, which is a multiple of 10 seconds, that a resource holder is allowed to run promoted when it is possibly causing resource contention. A resource holder is either an address space or an enclave. During this “resource contention residency” time, the resource holder

runs with the highest priority of all resource waiters to guarantee the needed importance. Also during this interval, the address space (including the address space associated with an enclave) is not considered for swap-out based on recommendation value analysis.

Value Range: 0-1000

Default Value: 6

MaxPromoteTime=0 causes SRM to not promote any resource holder. If MaxPromoteTime=12, SRM allows an address space or enclave to run 12*10 seconds (2 minutes) promoted. If the time exceeds, the promotion is canceled.

SDSF support for PROMOTED new column

SDSF provides a new column that shows whether displayed address spaces via the DA command or enclaves via the ENC command have a promoted status of YES or NO.

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF DA SC70     SC70     PAG  0  CPU/L/Z   3/  3/  0  LINE 66-70 (70)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   JOBNAME  me zACP-Time GCP-Use% zAAP-Use% SzAAP% SzIIP% Promoted zAAP-NTIM
      TN3270   00      0.00     0.65      0.00      0      0 NO
      D9D4DIST 00      0.00     0.00      0.00      0      0 NO
      D9D4ADMT 00      0.00     0.23      0.00      0      0 NO
      HERING   00      0.00     0.00      0.00      0      0 NO
      DFHSM70  00      0.00     0.00      0.00      0      0 NO

Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF ENCLAVE DISPLAY  SC70     ALL                     LINE 1-4 (4)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   NAME             P-Time zIIP-Time zICP-Time Promoted zAAP-NTime zIIP-NTi
      2400000002         0.00      0.00      0.00 NO             0.00       0.
      2000000001         0.00      0.00      0.00 NO             0.00       0.
```

Figure 8-3 SDSF support for new PROMOTED

SMF type 30 record update

There is a new field in the processor accounting section of the SMF type 30 record that contains the CPU time consumed while being promoted, as shown in Figure 8-4.

This new field gives the CPU time consumed for an address space or job while promoted because of chronic resource contention (in 1.024 millisecond units). For interval records, this field contains only the time consumed during the interval itself.

Offsets	Name	Length	Format	Description
148 94(hex)	SMF30CRP	4	binary	Cumulative CPU time consumed for an address space or job while promoted due to chronic resource contention (in 1.024 millisecond units).

Figure 8-4 New SMF type 30 record field

IPCS support

A new WLMDATA CONTENTION subcommand is introduced to print the contents of the resource topology. For each report type, you can select one or more of the following levels:

SUMMARY Displays summary information for each requested report type. SUMMARY is the default if no level is specified.

EXCEPTION Displays diagnostic information for error or exceptional conditions for each requested report type.

DETAIL Displays detailed information for each requested report type.

The reports provide the following types of information:

- The Contention Report requests information that is associated with the resource contention topology function. The resource contention topology is the workload manager's internal view of the list of resources, work units, or transactions involved with resources that have been in contention for longer than a resource manager interval. Resource managers use the IWMCNTN service to notify WLM of changes that cause WLM to maintain or update the topology.

  WLMDATA Summary CONTENTION prints (see Figure 8-5 on page 206):

  – Resources in contention table

  – Transactions with contention table

```
*****  CONTENTION SUMMARY REPORT  *****
   Resources in contention table
   ----------------------------
   RSRCE    Scope SS    SS         HT     WT  RID
   Address  S/M   Type  Name                 length
                  -------------------------------------------------
                  ResourceID  (first 50 bytes)
   ---------------------------------------------------------------
   025E6220   S   MSUB  MYSUBSYS  0001   0001 000D
                  RESOURCE_TEST.....................................
   Transactions with contention table
   ---------------------------------
   TRXNE    Type  Index Token              HR    WR
   Address  A/E
   ------------------------------------------------
   02312250   A   0028  000000A000000008  0000   0001
   02312208   A   002A  000000A800000008  0001   0000
```

Figure 8-5 IPCS WLMDATA CONTENTION SUMMARY report

- Resource, transaction, and contention elements without chaining information; see Figure 8-6 on page 207 and Figure 8-7 on page 208.

```
Resource element information.......................025E6220
  Resource Description
    Scope.............Single System
    Subsystem type.....MSUB          Subsystem name...MYSUBSYS
    ResourceID length..................................000D
    ResourceID........RESOURCE_TEST.....................
                  ==>...................................
                  ==>...................................
Transactions that hold this resource
  Transaction element information.....................02312208
    Transaction identifier
      Type........Address space     Index.................002A
      Token.......000000A800000008
    Contention element information...................022E7070
      Entity
        Type.................01      AStoken...000000A800000008
        TCB address...00000000      Etoken....0000000000000000
Transactions that are waiting for this resource
  Transaction element information.....................02312250
    Transaction identifier
      Type........Address space     Index.................0028
      Token.......000000A000000008
    Contention element information...................022E70AC
      Entity
        Type.................02      AStoken...000000A000000008
        TCB address...00000000      Etoken....0000000000000000
```

Figure 8-6 Resource, transaction, and contention elements without chaining information

```
Transactions with contention
 ---------------------------
   Transaction element information.....................02312250
     Transaction identifier
       Type.........Address space     Index.............0028
       Token.........000000A000000008
   Resources the transaction is holding
   Resources the transaction is waiting for
     Resource element information.....................025E6220
       Resource Description
         Scope............Single System
         Subsystem type...MSUB          Subsystem name...MYSUBSYS
         ResourceID length................................000D
         ResourceID......RESOURCE_TEST.....................
                      ==>.................................
                      ==>.................................
Contention element information...................022E70AC
         Entity
           Type.............02      AStoken...000000A000000008
           TCB address...00000000      Etoken....0000000000000000
   Transaction element information.....................02312208
     Transaction identifier
       Type.........Address space     Index.............002A
       Token.........000000A800000008
..............
```

Figure 8-7 Resource, transaction, and contention elements without chaining information

- WLMDATA Detail CONTENTION prints:
  - Global contention information, see Figure 8-8 on page 208.
  - Resource, transaction, and contention elements with chaining information, see Figure 8-9 on page 209 and Figure 8-10 on page 209.

```
Global contention information
-----------------------------
  Resource topology anchors in SRM control table........0247A418
    Resource element anchor
      First.............025E6220     Last.............025E6220
    Transaction element anchor
      First.............02312250     Last.............02312208
  Cell pool IDs
    Resource element Cell Pool ID.....................025E61F8
    Transaction element Cell Pool ID..................023121E0
    Contention element Cell Pool ID...................022E7048
```

Figure 8-8 WLMDATA Detail CONTENTION Global contention information

```
Resources in contention
-----------------------
  Resource element information.........................025E6220
    Resource Description
      Scope..............Single System
      Subsystem type.....MSUB          Subsystem name...MYSUBSYS
      ResourceID length...................................000D
      ResourceID........RESOURCE_TEST.....................
                       ==>................................
                       ==>................................
Queue Information
      Resource element links
        Previous.........0247A644     Next.............0247A644
      Anchor of Holder contention elements
        First............022E7070     Last.............022E7070
      Anchor of Waiter contention elements
        First............022E70AC     Last.............022E70AC
```

Figure 8-9 Resources in contention

```
Transactions that hold this resource
  Transaction element information.....................02312208
    Transaction identifier
      Type........Address space     Index................002A
      Token.......000000A800000008
    Queue Information
      Transaction element links
        Previous.......0247A650     Next.............02312250
      Anchor of Holder contention elements
        First..........022E7070     Last.............022E7070
      Anchor of Waiter contention elements
        First..........02312228     Last.............02312228
    Contention element information...................022E7070
      Entity
        Type.............01      AStoken...000000A800000008
        TCB address...00000000      Etoken....0000000000000000
      Queue Information
        Contention element links
          Transaction element address.................02312208
          Resource    element address.................025E6220
        Transaction anchored contention element queue
          Previous.....0231221C     Next.............0231221C
        Resource anchored contention element queue
          Previous.....025E6230     Next.............025E6230
```

Figure 8-10 Transactions in contention

- WLMDATA Exception CONTENTION prints:
  - Validation and exception details for resource, transaction, and contention elements

8.1.3 RMF support for resource contention

To support the new resource contention enhancements, RMF provides the temporary dispatching priority promotion times for workloads and an indication for jobs running with temporarily promoted priority. The purpose of this is to integrate service times related to temporary dispatch promotion caused by enqueue and chronic resource contention management into the existing RMF reports. The following two additional extensions are provided with this support:

- The WLMGL reports showing using and delay percentages are extended by contention using and delay values.
- An indicator for jobs with promoted dispatch priority (job-held resources needed by other work) is added to the SMF79 subtype 1 record.

The RMF Postprocessor WLMGL reports are extended with information about the times for temporary dispatching priority promotions caused by basic enqueue and chronic resource management. The data is provided in SMF 72 subtype 3 in the Workload activity section.

The RMF Postprocessor WLMGL service class reports are reformatted and contention delay and using values are added.

The SMF 79 subtype 1 and 2 records (MII Address Space Resource Data and Address Space State Data) are extended by a flag indicating that the address space is temporarily running with promoted dispatching priority.

Using the RMF WLMGL Activity Report

With this report, you can identify service classes running work units at a promoted dispatching priority. A new PROMOTED block, shown in Figure 8-11 on page 210, is added to visualize the times the workload is temporarily promoted because of blocked workload, enqueue management and chronic resource contention management.

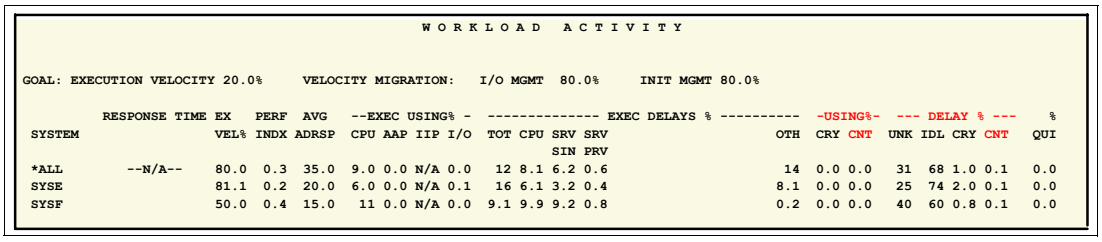
```
                                W O R K L O A D   A C T I V I T Y

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---  SERVICE TIME  ---APPL %---  --PROMOTED--  -----STORAGE-----
  AVG       0.00  ACTUAL                0  SSCHRT   0.0  IOC         0  CPU    0.077  CP      0.04  BLK     0.000  AVG       1814.29
  MPL       0.00  EXECUTION             0  RESP     0.0  CPU     21160  SRB    0.001  AAPCP   0.00  ENQ     0.012  TOTAL        0.95
  ENDED       54  QUEUED                0  CONN     0.0  MSO    223247  RCT    0.001  IIPCP   0.00  CRM     0.350  SHARED       0.00
  END/S     0.26  R/S AFFIN             0  DISC     0.0  SRB       212  IIT    0.000
  #SWAPS      54  INELIGIBLE            0  Q+PEND   0.0  TOT    244619  HST    0.000  AAP     0.00                 --PAGE-IN RATES--
  EXCTD        0  CONVERSION            0  IOSQ     0.0  /SEC     1183  AAP    0.000  IIP     0.00                 SINGLE       0.0
  AVG ENC   0.00  STD DEV               0                              IIP    0.000                               BLOCK        0.0
  REM ENC   0.00                                          ABSRPTN  2265K                                        SHARED       0.0
  MS ENC    0.00                                          TRX SERV 2038K                                        HSP          0.0
```

Figure 8-11 Workload Activity report with new PROMOTED block

WLMGL Workload Activity report

The Service Class/Report Class Period report is reformatted to integrate the contention USING% and contention DELAY% values. The integration of contention delay% and using% values in the WLMGL workload activity service class/report class period report requires a layout modification because of missing space. All using and delay percentage values are now displayed in 3-character fields.



```
                                    W O R K L O A D   A C T I V I T Y


GOAL: EXECUTION VELOCITY 20.0%     VELOCITY MIGRATION:   I/O MGMT  80.0%     INIT MGMT 80.0%

          RESPONSE TIME EX   PERF  AVG   --EXEC USING% -  -------------- EXEC DELAYS % ----------  -USING%-  --- DELAY % ---    %
 SYSTEM                 VEL% INDX ADRSP  CPU AAP IIP I/O  TOT CPU SRV SRV                      OTH  CRY CNT  UNK IDL CRY CNT   QUI
                                                                    SIN PRV
 *ALL        --N/A--    80.0  0.3  35.0  9.0 0.0 N/A 0.0   12 8.1 6.2 0.6                     14  0.0 0.0   31  68 1.0 0.1   0.0
 SYSE                   81.1  0.2  20.0  6.0 0.0 N/A 0.1   16 6.1 3.2 0.4                    8.1  0.0 0.0   25  74 2.0 0.1   0.0
 SYSF                   50.0  0.4  15.0   11 0.0 N/A 0.0  9.1 9.9 9.2 0.8                    0.2  0.0 0.0   40  60 0.8 0.1   0.0
```

Figure 8-12 Workload Activity report with USING% and DELAY%

For additional information, see the following publications:

- *z/OS Resource Management Facility (RMF) Performance Management Guide*, SC33-7992
- *z/OS Resource Management Facility (RMF) Report Analysis*, SC33-7991
- *z/OS Resource Management Facility (RMF) User's Guide*, SC33-7990

8.2 WLM pageable storage management

Pageable storage shortages are one of the most critical to resolve, because real storage is a discrete resource that is limited. An unresolved real storage shortage can result in a system outage. One of the biggest unresolved problems is the handling of address spaces that allocate and fix too much storage.

z/OS V1R10 introduces a new implementation called pageable storage management to provide better management of a pageable storage shortage. The system now monitors the fixed storage consumption of address spaces every two seconds. This information is used to find the address spaces with the highest fixed storage increase. Additional messages and an enhanced ENF 55 signal provide earlier and more information about a pageable storage shortage. The non-dispatchable functionality allows the system to set address spaces, which permanently increases the amount of fixed storage non-dispatchable. This is only done for non-system address spaces.

WTOR processing allows operators or an automation task to easily cancel the address space with the highest fixed storage allocation.

8.2.1 New IEAOPTxx keywords

Pageable storage management is controlled by three new IEAOPTxx parmlib member parameters, as follows:

IRA405I(n%) Specifies the percentage of the fixed storage that causes the system to issue message IRA405I. Specify a value for n to indicate the storage area, where n can be:

0 - Real storage area below 16M. Value Range: 0 to 100. **Default Value: 70**

1 - Real storage area between 16M and 2G. Value Range: 0 to 100. **Default Value: 50**

2 - Total real storage. Value Range: 0 to 100. **Default Value: 50**

STORAGENSWDP Specifies whether the system should set non-swappable address spaces non-dispatchable, to resolve a storage shortage. The syntax is: STORAGENSWDP=option where option is either NO or YES.

YES - Specifies that the system should also select non-swappable address spaces to resolve the storage shortage. The system then sets non-swappable address spaces non-dispachable and issues message IRA210E or IRA410E.

Note: The system does not set address spaces in service class SYSTEM non-dispatchable. **Default Value: YES**

STORAGEWTOR Specifies how the system handles address spaces in a critical storage shortage. The syntax is: STORAGEWTOR=option where option is NO, AUTO, or YES.

YES - Specifies that the system presents a list of address spaces on the console. The operator can reply to a WTOR (IRA221D or IRA421D) request and select which address space to cancel.

AUTO - AUTO is similar to STORAGEWTOR=YES, except that the IRA210I or IRA420I message presents up to 20 address spaces at once. This option is useful when an automation product needs to answer the WTOR request. **Default Value: YES**

8.2.2 Using the IEAOPTxx parameters

The three IEAOPTxx parmlib member keywords produce messages and an ENF 55 signal depending on the selected options.

IRA405I(nn%)

When 50% of the real storage is fixed, the IRA405I(nn%) IEAOPTxx keyword default, a new message, IRA405I, is issued, which indicates that a high amount of frames are fixed in the system. If the system stays above the warning level, the message gets reissued every two hours.

```
IRA405I return code, nn% OF THE REAL STORAGE FRAMES ARE FIXED
```

Where:

return-code The return-code indicates the type of the detected shortage. The order in the list below indicates how severe the shortage is. The most severe type is on the top of the following list. The possible values for return-code are:

03 - Pageable frames below

16 - Megabyte shortage

04 - Pageable frames between 16 megabytes and 2 gigabytes shortage

02 - Pageable frames in real storage shortage

01 - Pageable to auxiliary (PTA) frames (DREF + fixed pages) in processor storage

nn% The percentage of fixed frames in the shortage area.

A new ENF 55 signal qualifier is added, to inform listening applications about this level.

Note: A sample program, IRAEN55S, is available in SYS1.SAMPLIB that gives application programmers additional information about how to listen for the ENF 55 signals.

Large pageable shortage 80%

When 80% of the real storage is fixed, messages IRA400E and IRA404I are issued. Also issued is the ENF 55 signal which is enhanced for this support and can contain a list of the top 20 contributors to the pageable storage shortage.

```
IRA400E return-code, PAGEABLE STORAGE SHORTAGE
IRA404I uuuuuuuu ASID aaaa OWNS xxxxxxxxxx PAGES, yyyyyyyyyy FIXED, zzzzzzzzzz
FIXED IN SHORTAGE AREA
```

IRA400E explanation: The system detected a shortage of pageable central storage frames.

Return-code: The return-code indicates the type of the detected shortage. The order in the list below indicates how severe the shortage is. The most severe type is at the top of the list. The possible values for return-code are as follows:

- 03 - Pageable frames below 16 megabytes shortage
- 04 - Pageable frames between 16 megabytes and 2 gigabytes shortage
- 02 - Pageable frames in real storage shortage
- 01 - Pageable to auxiliary (PTA)

IRA404I explanation: After message IRA400E has been issued, the message IRA404I lists the five largest users of pageable central storage frames in the shortage area.

Message text:

- uuuuuuuu - One of the five largest users of pageable storage frames in the shortage area.
- aaaa - Address space ID of the user.
- xxxxxxxxxx - Number of frames the user owns
- yyyyyyyyyy - Number of fixed frames the user owns
- zzzzzzzzzz - Number of fixed frames in the shortage area the user owns

Note: If IRA400E detected a shortage of type 1, zzzzzzzzzz can be bigger than yyyyyyyyyy because of the included DREF pages.

STORAGENSWDP

If the address spaces with the highest increase are swappable, message IRA410E is issued and the address spaces are logically swapped. The IRA410E message is issued at the time of a pageable storage shortage. The system identified a non-swappable address space with the largest fixed frame increase. The address space is set non-dispatchable so that the address space cannot further increase the amount of fixed frames.

If the address spaces with the high slot increase of fixed frames are non-swappable, based on the IEAOPTxx specification STORAGENSWDP, issue the IRA210E message and no longer dispatch the address spaces.

```
IRA210E uuuuuuuu ASID nnnn SET NON DISPATCHABLE
```

uuuuuuuu Non-swappable address space name of the space with a high slot increase

aaaa Address space ID of the user

IRA210E explanation: When an auxiliary shortage occurred, the system identified a non-swappable address space with the largest slot increase. The address space is set non-dispatchable so that the address space cannot further increase the amount of slots.

System action: The system sets the address space non-dispatchable and no longer processes the address spaces until the shortage is relieved. When the address space gets dispatchable, the system writes message IRA211I.

```
IRA211I uuuuuuuu ASID aaaa SET DISPATCHABLE
```

For address spaces that are non-swappable with a large fixed frame increase, issue IRA410E:

```
IRA410E JOB uuuuuuuu ASID aaaa SET NON DISPATCHABLE
```

IRA410E explanation: At the time of a pageable storage shortage, the system identified a non-swappable address space with the largest fixed frame increase. The address space is set non dispatchable so that the address space cannot further increase the amount of fixed frames.

uuuuuuu Non-swappable address space name of the space with a high fixed frame increase

aaaa Address space ID of the user

The system sets the address space non-dispatchable and no longer processes the address spaces, until the shortage is relieved. When the address space gets dispatchable, the system issues the IRA411I message.

```
IRA411I uuuuuuuu ASID nnnn SET DISPATCHABLE
```

Large pageable shortage 90%

At 90% of the real storage fixed, message IRA401E gets issued if the address spaces with the highest amount of fixed storage are swappable.

```
IRA401E return-code,CRITICAL PAGEABLE STORAGE SHORTAGE
```

IRA401E explanation: The system detected a critical shortage of pageable processor storage frames.

return-code: The return-code indicates the type of the detected shortage. The order in the list below indicates how severe the shortage is. The most severe type is at the top of the list. The possible values for return-code are as follows:

- 03 - Pageable frames below 16 megabytes shortage
- 04 - Pageable frames between 16 megabytes and 2 gigabytes shortage
- 02 - Pageable frames in real storage shortage
- 01 - Pageable to auxiliary (PTA) frames (DREF + fixed pages) in processor storage

Issue IRA410E, shown above, and logically swap the address spaces. If the address spaces with the highest amount of fixed storage are non-swappable, STORAGENSWDP keyword, issue IRA410E and no longer dispatch the address spaces. If the system is more than 15 seconds in a pageable storage shortage, issue message IRA420I and WTOR IRA421D.

STORAGEWTOR

The IEAOPTxx keyword STORAGEWTOR specifies that a list of address spaces is to be displayed. Tivoli System Automation can provide an automation for this new WTOR.

At the time of a critical pageable storage shortage, the system identifies 20 address spaces with the largest fixed frame allocation in the system. These address spaces get displayed and the operator is able to cancel one of them using the outstanding WTOR message IRA421D.

```
IRA420I CRITICAL STORAGE SHORTAGE
```

```
IRA420I ! ## ! USER     ! ASID ! PAGES      ! O/W FIXED  !
IRA420I +----+----------+------+-----------+------------+
IRA420I ! 01 ! I9A2GH11 ! 0081 ! 0000065939 ! 0000065851 !
IRA420I ! 02 ! I9A2GH10 ! 0087 ! 0000065936 ! 0000065517 !
IRA420I ! 03 ! I9A2GH13 ! 0088 ! 0000066002 ! 0000062839 !
IRA420I ! 04 ! XCFAS    ! 0006s! 0000020870 ! 0000001436 !
IRA420I ! 05 ! TRACE    ! 0004s! 0000008262 ! 0000000706 !

*73 IRA421D REPLY M FOR MORE, E TO END, ## TO CANCEL A USER
```

In the message text: ## - Number of the line for the cancel of one of the five address spaces. Or reply with `M` to display the next five address spaces that you can cancel afterward.

8.3 WLM resource delays

Subsystem work managers, such as CICS and WebSphere Application Server use the execution delay monitoring services to tell workload management about their view of the current state of a work request, such as:

```
ready state, idle state, or waiting state
```

The information is kept in performance blocks (PBs), also called monitoring environments.

Using execution delay monitoring services, workload management knows how well work is executing, and where any delays are occurring. The services also provide execution delay information, so that installations can determine where work is being delayed. With this information, the work manager's configuration can be adjusted.

WLM state reporting

WLM provides performance blocks for use in work request state reporting. The state reporting can be done using the IWM4MCHS WLM service. When STATE=WAITING is specified, RESOURCE is a required parameter that indicates the resource that the work manager is waiting for on behalf of the work request described by the performance block.

RESOURCE=TYPEx is a generic name for some internal resource of the work manager for which the work request is waiting. The current implementation only allows TYPEx to be TYPE1 to TYPE5, which is not enough for certain subsystems.

With z/OS v1R10, additional generic names, TYPE6 to TYPE15, can now be specified. This allows subsystems to define descriptions for each TYPEx. These TYPEx names are displayed in RMF.

8.3.1 Subsystem work manager enhancements

A subsystem work manager can now specify up to 15 different generic internal resource types when reporting that work is delayed using the IWM4MCHS service:

```
IWM4MCHS … STATE=WAITING, RESOURCE=TYPEx
```

A short verbal description for each of the generic internal resource types can be specified and can be up to 16 characters using a new WLM IWM4MGDD service. RMF displays these verbal descriptions. This can help to better understand where to change the customization of the work manager in order to reduce the reported delays.

New IWM4MGDD service

With this new service, a subsystem can define descriptions for its generic delay states. The term *generic delay states* in this context is related to service IWM4MCHS. When STATE=WAITING is specified and the resource that is specified is RESOURCE=TYPEx, where x is some number between 1 and 15, with this service, descriptions can be defined that might be more intuitive to a user than the generic terms.

If defined, these descriptions are accessible to performance monitors in the IWMWRCAA data area as a result of a call to the IWMRCOLL service.

Note: For a subsystem that allows multiple instances (address spaces) to be active on the same z/OS system, only one set of descriptions can be in effect. If more than one instance of such a subsystem defines a set of definitions, the last one defined will be effective.

For a subsystem that allows multiple instances to run, possibly at different release levels on the same system, it might be helpful if each instance first checks whether there are already some descriptions defined for that subsystem, before the instance decides whether to define its own descriptions.

The set of descriptions defined by a subsystem does not need to include all 15 TYPEx. A subset can be specified as follows:

```
IWM4MGDD  REQTYPE=DEFINE,DESCRIPTIONS=descriptions --- to Define the set of
descriptions
IWM4MGDD  REQTYPE=RETRIEVE,DESCRIPTIONS=descriptions --- to Retrieve the (last
defined) set of descriptions

descriptions is the address of a data area mapped by macro IWMWGDD
```

8.3.2 RMF reporting for the WLM PB delays support

The RMF Postprocessor WLMGL reports are extended with descriptions for the generic performance block (PB) states, TYPEx. The data is provided in Workload Activity SMF record type 72-3, shown in Figure 8-13 on page 217. The generic delay description can offer additional information about the delay reason.

Ten additional performance block (PB) states with descriptive names are reported in order to better track what the transactions are waiting for were required . The new WLM service IWM4MGDD, Define Descriptions for Generic Delay States, introduced in z/OS V1R10 can be used to allow exploiters, for example WebSphere, to assign a meaningful name to performance blocks for an application. These PB state names are provided by RMF with z/OS V1R10. The usability of the WLMGL report is improved because a meaningful name description assigned by an application is expected to be easier to interpret than the generic names such as TYPE1 and TYPE2.

SMF record type 72 subtype 3 – Workload Activity				
Offset	**Name**	**Length**	**Format**	**Description**
Individual header extension for subtype 3 :				
.....				
82 52	SMF72WRN	2	Binary	Number of work/resource manager state sections
84 54	SMF72DNS	4	Binary	Offset to resource delay names sections.
88 58	SMF72DNL	2	Binary	Length of resource delay names section..
90 5A	SMF72DNN	2	Binary	Number of resource delay names sections .

Figure 8-13 Workload Activity SMF record type 72-3

A new section, Resource Delay Names, is added to the SMF 72 record subtype 3.

Work Manager/Resource Manager State Section				
Offset	**Name**	**Length**	**Format**	**Description**
156 9C	R723RBPM	4	Binary	Number of state samples representing buffer pool misses the result in I/O.
160 A0	R723RDNX	2	Binary	Index into resource delay names table
162 A2	R723RDNN	2	Binary	Number of entries in resource delay names table.
164 A4		8		Reserved

Resource Delay Names Section				
Offset	**Name**	**Length**	**Format**	**Description**
0 0	R723DNST	4	EBCDIC	Subsystem type, as used in the classification rules specified in the WLM administrative application.
4 4	R723DNNU	2	Binary	Number of the resource delay (between '01' and '15' - related to R723RWxx)
6 6	R723DNDE	16	EBCDIC	Resource delay name
22 16	R723RRS4	2	Binary	Reserved

Figure 8-14 SMF record type 72-3

The Resource Delay Names section contains the defined descriptions, subsystem by subsystem, so that the index and number available in the SMF record type 72 subtype 3 work manager and Resource Manager state section can be used to address the appropriate delay reason descriptions. For each subsystem, delay type 1-15 is possible; undefined delay types are omitted.

9

IBM Health Checker for z/OS

The objective of IBM Health Checker for z/OS is to identify potential problems before they impact your availability or, in worst cases, cause outages. It checks the current active z/OS and sysplex settings and definitions for a system and compares the values to those suggested by IBM or defined by you. It is not meant to be a diagnostic or monitoring tool, but rather a continuously running preventative that finds potential problems. IBM Health Checker for z/OS produces output in the form of detailed messages to let you know of both potential problems and suggested actions to take. Note that these messages do not mean that IBM Health Checker for z/OS has found problems that you need to report to IBM! IBM Health Checker for z/OS output messages simply inform you of potential problems so that you can take action on your installation.

This chapter describes the changes made in z/OS V1R10, as follows:

- Using Health Checker infrastructure for migration purposes
- Improvements to the infrastructure
  - HZSPRINT
  - Persistent data
  - Virtual storage constraint relief
- New checks
  - RACF (2 new checks, 1 updated check)
  - XCF/XES (9 new checks, 5 updated checks)
  - USS (2 new checks)
  - SDSF (1 new check)
  - Language Environment (1 new check)

9.1 Using IBM Health Checker for z/OS for migration purposes

Starting in z/OS V1R10, the Health Checker infrastructure is exploited for migration purposes. Health checks that are helpful for determining migration action applicability are provided. These checks, called “migration health checks”, should be used prior to your migration to z/OS V1R10.

Accessing the health checks

To be able to run the migration checks on your existing system before you migrate to a new release, the checks are provided in the service stream as PTFs. The following are true:

- You must install the PTFs on your current system.
- Like all IBM Health Checker for z/OS checks, migration checks can be found using the functional PSP bucket HCHECKER.
- Alternatively, you can see all IBM Health Checker for z/OS checks at:

  `http://www.ibm.com/systems/z/os/zos/hchecker/check_table.html`

Install on the current system

An option is to install the PTFs during a regular service window so that an IPL is scheduled afterwards. Checks are often added by a function when it is started or restarted, so you might find that installing the PTFs before a scheduled IPL might be a better option. Additional migration checks can be added at different times, so having all the latest ones installed prior to making migration plans is recommended.

Migration checks naming convention

Because the naming convention for migration checks indicates which release introduced the corresponding migration actions, activate just the checks appropriate for the migration path. Using SDSF (or another method for viewing checks, such as filters), you can view ahead of time which migration checks you have available on your system. For example:

- If you are migrating from z/OS V1R8 to z/OS V1R10, you need to activate the migration checks for changes that occurred in both z/OS V1R9 and z/OS V1R10.
- If you are migrating from z/OS V1R9 to z/OS V1R10, you only need to activate the migration checks for changes that occurred in z/OS V1R10.

The new migration checks are very similar to the other checks provided by IBM Health Checker for z/OS. The only differences are:

- The names have a convention, ZOSMIGVvvRrr_component_program_name (or, for ICSF, ICSFMIGnnnn_component_program_name),

  where VvvRrr is the version and release that introduced the migration action.

  Notice the MIG characters followed immediately by the release identifier. This convention tells you that the check helps with migration and it tells you the release in which the migration action was introduced.
- By default, migration checks are inactive. This is because you might not want to know about migration actions during non-migration periods.

9.1.1 Activating the migration checks

There are many ways to make a check active, as well as many ways of using wildcards to include specific checks. Here are some examples of using the MODIFY command to make checks active:

```
F HZSPROC,ACTIVATE,CHECK=(IBM*,*MIGV1*)
F HZSPROC,ACTIVATE,CHECK=(IBM*,ICSFMIG*)
F HZSPROC,ACTIVATE,CHECK=(IBM*,ZOSMIGV1R10)
```

Deactivate checks

To deactivate the migration checks if you desire when it becomes no longer necessary to have the migration checks active, you can deactivate them similar to the way you activated them. For example:

```
F HZSPROC,DEACTIVATE,CHECK=(IBM*,*MIGV1*)
F HZSPROC,DEACTIVATE,CHECK=(IBM*,ICSFMIG*)
F HZSPROC,DEACTIVATE,CHECK=(IBM*,ZOSMIGV1R10)
```

9.1.2 z/OS V1R10 migration health checks

The checks are provided in the service stream as APARs and can be installed on current systems. After applying the maintenance for migration health checks, to activate a check, issue the following command and you receive a message that the check is active:

```
F HZSPROC,ACTIVATE,CHECK=(IBM*,ZOSMIGV1R*)
```

Following are the APARs:

- AA24327 - (z/OS R8 RACF, HRF7730)
  - Check "ZOSMIGV1R9_RACF_PASSWRD_ENVELOPE"
- AK59629 - (z/OS R9 SDSF, HQX7740)
  - Check "SDSF_ISFPARMS_IN_USE"
- AK62487 - (z/OS R9 LE, HLE7740)
  - Check "CEE_USING_LE_PARMLIB"
- BK59629 - (z/OS R8 SDSF, HQX7730)
  - Check "SDSF_ISFPARMS_IN_USE"
- BK62487 - (z/OS R8 LE, HLE7730)
  - Check "CEE_USING_LE_PARMLIB"
- GA24221 - (ICSF HCR7731, in z/OS R8), check
  - "ICSFMIG7731_ICSF_RETAINED_RSAKEY" and check
  - "ICSFMIG7731_ICSF_PKDS_TO_4096BIT"
- HA24221 - (ICSF HCR7740, in z/OS R9), check
  - "IICSFMIG7731_ICSF_RETAINED_RSAKEY" and check
  - "ICSFMIG7731_ICSF_PKDS_TO_4096BIT"
- IA24221 - (ICSF HCR7750, in z/OS R10), check
  - "ICSFMIG7731_ICSF_RETAINED_RSAKEY”

Table 9-1 describes the migration health checks made available with z/OS V1R10.

Table 9-1 New migration health checks with z/OS V1R10

Migration check - Applies to: - Migration action:	Description	Reason for check
ZOSMIGV1R9_RACF_PASSWRD_ENVELOPE z/OS V1R8 **Migration action:** Use more-specific profile names to control LDAP change logging of RACF updates.	This check is to identify those installations which are not running password enveloping and have defined a generic profile that could cause RACF to start enveloping passwords once they move to z/OS V1R9 or later.	Check lets you know when password or password phrase enveloping is not being done on z/OS V1R8 and will be enabled for z/OS V1R9 or later. You can then determine if this is what you want and if you are properly configured for enveloping.
ZOSMIGV1R10_CS_BIND4 z/OS V1R8 z/OS V1R9 z/OS V1R10 **Migration action:** For this Communication Server check, migrate from BIND DNS 4.9.3.	Checks whether the Berkeley Internet Name Domain 4.9.3 (BIND 4.9.3) DNS server is in use on this system. IBM has indicated in statements of direction that the BIND 4.9.3 DNS server will not be available in future IBM z/OS Communications Server releases after z/OS V1R10.	Since the BIND 4.9.3 DNS server will no longer be supported in future releases after z/OS V1R10, IBM suggests that customers who currently use or plan to use BIND 4.9.3 DNS server implement the BIND 9.2.0 DNS server as a replacement. Customers who use the load balancing Connection Optimization (DNS/WLM) feature of BIND 4.9.3 DNS server should investigate Sysplex Distributor, the Load Balancing Advisor (LBA), Automated Domain name Registration (ADNR), or other load balancing solutions.
ZOSMIGV1R10_CS_BINL z/OS V1R8 z/OS V1R9 z/OS V1R10 **Migration action:** For this Communication Server check, migrate from the Boot Information Negotiation Layer function.	Checks whether the Boot Information Negotiation Layer (BINL) server function is in use on this system. IBM has indicated in statements of direction that the BINL server will not be available in future IBM z/OS Communications Server releases after z/OS V1R10.	Since the BINL server will no longer be supported in releases after z/OS V1R10, IBM suggests that customers who currently use or plan to use BINL server should investigate the use of IBM Director and Remote Deployment Manager (Tivoli Provisioning Manager for OS Deployment Director Extension) for network-based operating system installation services.
ZOSMIGV1R10_CS_DHCP z/OS V1R8 z/OS V1R9 z/OS V1R10 **Migration action:** For this Communication Server check, migrate from the Dynamic Host Configuration Protocol server.	Checks whether the Dynamic Host Configuration Protocol (DHCP) server function is in use on this system. IBM has indicated in statements of direction that the DHCP server will not be available in future IBM z/OS Communications Server releases after z/OS V1R10.	Since DHCP server will no longer be supported in releases after z/OS V1R10, IBM suggests that customers who currently use or plan to use DHCP server should investigate using a DHCP server on Linux for System z.

Migration check - Applies to: - Migration action:	Description	Reason for check
ZOSMIGV1R10_CS_NDB z/OS V1R8 z/OS V1R9 z/OS V1R10 **Migration action:** Migrate from Network Database function	Checks whether the Network Database System (NDB) server function is in use on this system. IBM has indicated in statements of direction that the NDB server will not be available in future IBM z/OS Communications Server releases after z/OS V1R10.	Since NDB server will no longer be supported in future releases after z/OS V1R10, IBM suggests that customers who currently use or plan to use the NDB server should investigate the Distributed Data Facility (DDF) provided by z/OS DB2, and the DB2 Run-Time Client. DDF allows client applications running in an environment that supports DRDA to access data at DB2 servers.
ICSFMIG7731_ICSF_PKDS_TO_4096BIT z/OS V1R8 (or ICSF FMID HCR7731) and z/OS V1R9 (or ICSF FMID HCR7740) **Migration action:** For this ICSF check, increase the size of your PKDS.	Verifies that the PKDS size in an ICSF pre-HCR7750 environment is sufficiently allocated to support 4096-bit RSA keys.	ICSF FMID HCR7750 introduces support for 4096-bit RSA keys, which requires a larger PKDS than prior ICSF releases needed. If a customer at a pre-HCR7750 FMID ICSF level migrates to HCR7750 without first reallocating the PKDS for 4096-bit key support, ICSF at HCR7750 will fail to start. This ICSF migration check will detect the case where the currently active PKDS is not sufficiently allocated for the HCR7750 environment and inform the customer that a PKDS reallocation action is necessary.
ICSFMIG7731_ICSF_RETAINED_RSAKEY z/OS V1R9 z/OS V1R10 **Migration action:** Stop using RSA private keys on a PCICC or PCIXCC/CEX2C cryptographic card.	PCIXCC/CEX2C cryptographic card. You should run the check periodically, when the events occur that affect check results. For example, run the check dynamically when: ► The ICSF product release level is being upgraded to any new ICSF release level. ► The z/OS product release level is being upgraded and ICSF is an exploited feature for that z/OS image.	A PCICC or PCIXCC/CEX2C card may possess the only copy of a retained RSA private key. Customers that run applications and middleware that utilize the retained key functionality of these cards are exposed to the loss of keys upon hardware failure, which may result from a problem as simple as an exhausted or malfunctioning card battery. Lost retained keys have the further implication of lost data, for retained key management keys, and an inability to verify signatures, for retained signature keys. Starting with the Cryptographic Support for z/OS V1R7-V1R9 and z/OS.e V1R7-V1R8 Web deliverable (ICSF FMID HCR7750), you no longer have the ability to store new private RSA keys intended for key management usage in a cryptographic coprocessor. Existing applications will continue to be able to use the retained keys and to delete them from the cryptographic coprocessor cards.
CEE_USING_LE_PARMLIB z/OS V1R8 z/OS V1R9 z/OS V1R10	Verifies use of Language Environment parmlib CEEPRMxx.	Default Language Environment run-time options should be set within a CEEPRMxx parmlib member. If USERMODs are in use, they should be converted to use parmlib.
SDSF_ISFPARMS_IN_USE z/OS V1R8 z/OS V1R9 z/OS V1R10	Checks that SDSF dynamic statements in ISFPRMxx are being used for configuration options to avoid reassembly of ISFPARMS. Checks that if ISFPARMS is being used, only default values have been specified.	ISFPARMS is used for specifying global configuration options, panel formats, and security for SDSF functions. There are two alternatives for ISFPARMS: assembler macros that you define, assemble, and link into the SDSF load library, and statements which are updated using a text editor and reside in parmlib.

9.2 Improvements to the infrastructure

Several internal changes have been implemented in z/OS V1R10 for Health Checker to the following functions:

- HZSPRINT utility
- Persistent data services
- Virtual storage constraint relief (VSCR)

9.2.1 HZSPRINT utility

The HZSPRINT utility lets you see check output in the message buffer or the IBM Health Checker for z/OS log streams. HZSPRINT writes the current message buffer for the target checks to SYSOUT.

The current HZSPRINT service only returns information about the current iteration of a check, or to the information saved in a specified logstream. With z/OS V1R10, HZSPRINT can start with the current iteration of a check, and obtain multiple iterations of a check within a specified time period. The HZSQUERY service is enhanced to obtain the n-1 iteration of a check from the HZS log stream. This allows easy access to the message buffers of multiple iterations of a check.

New keyword on HZSPRINT

The TIMERANGE keyword allows the HZSPRINT utility to window the results to a specified check start time. TIMERANGE lets you limit the data in SYSOUT from the log stream to the data from the time between the starting and stopping time range.

Specify the 12-char-start and 12-char-stop as YYYYMMDDHHMM. All 12 characters must be valid decimal digits and must represent a valid year, month (01-12), day (01-31 depending on the month), hour (00-23), and minute (00-59) specification.

You can specify the TIMERANGE parameter only with LOGSTREAM. The system ignores the parameter when processing non-LOGSTREAM data.

```
TIMERANGE(yyy1m1d1h1m1,yyy2m2d2h2m2)

Where:
yyy1: the 4-digit year of the start time
m1:   the 2-digit month of the start time
d1:   the 2-digit day of the start time
h1:   the 2-digit hour (0-23) of the start time (second)
m1:   the 2-digit minute (0-53) of the start time
yyy2: the 4-digit year of the stop time
m2:   the 2-digit month of the stop time
d2:   the 2-digit day of the stop time
h2:   the 2-digit hour (0-23) of the stop time (second)
m2:   the 2-digit minute (0-53) of the stop time
```

In the HZSPRINT JCL stream, an example of a specification is:

```
PARM=('CHECK(check_owner,check_name)','EXCEPTIONS',
'TIMERANGE(yyy1m1d1h1m1, yyy2m2d2h2m2)')
```

If no LOGSTREAM is specified, but check iterations have been written to the logstream, HZSPRINT processing will go backwards from the current iteration through the save check iterations.

9.2.2 Persistent data services

Persistent data is a data structure that preserves its old versions, that is, previous versions may be queried in addition to the latest version.

New services for access to persistent data

The HZSPREAD read service for persistent data and the HZSPWRIT write service for persistent data are implemented to save and retrieve data between IPLs and are now available to non-BCP checks. Persistent data continues to be saved in the HZSPDATA DD. HZSPDATA may have to be resized as persistent data users increase over time.

Persistent data allows checks to save and compare values between IPLs. These services were only available previously to BCP checks. The HZSPWRIT macro is an interface used by check routines to write persistent data into the IBM Health Checker for the z/OS Persistent data set, which is allocated using the HZSPDATA ddname in the startup procedure.

Use HZSPWRIT only within the Init, Check, or Cleanup function for a local check, or within the InitRun or Run function for a remote check. The data is associated with the writing check, and can be retrieved by the HZSPREAD macro, specifying the check owner and check name. The data remains even if the writing check is deleted. If the check iteration completes with an abend (or a remote check iteration is designated unsuccessful) or an invocation of HZSPWRIT is not successful, then the persistent data for that iteration is not retained. Note that “unsuccessful” has no correlation with whether or not the check detected exceptions.

The HZSPREAD macro is an interface used by check routines to read data that has been preserved in the IBM Health Checker for z/OS Persistent data set. Two groups of data are preserved for an IPL, the first and the most recent.

9.2.3 Virtual storage constraint relief

Health Checker control blocks previous to z/OS V1R10 are below the bar. With this release, the internal control blocks are moved to 64 bit storage. This increases the potential number of registered checks for Health Checker. Now, Health Checker services can be called by Amode 64-bit callers.

9.3 Current health checks

All checks are local checks since they run in the IBM Health Checker for z/OS address space, unless otherwise noted. The following are the existing z/OS components that provide health checks:

- ASM checks (IBMASM)
- Communications Server checks (IBMCS)
- Consoles checks (IBMCNZ)
- Contents supervision checks (IBMCSV)
- Global Resource Serialization checks (IBMGRS)
- HSM checks (IBMHSM)

- ICSF checks (IBMICSF)
- Language Environment checks (IBMCEE)
- PDSE checks (IBMPDSE)
- RACF checks (IBMRACF)
- RRS checks (IBMRRS)
- RSM checks (IBMRSM)
- SDSF checks (IBMSDSF)
- SDUMP checks (IBMSDUMP)
- Supervisor checks (IBMSUP)
- System logger checks (IBMIXGLOGR)
- TSO/E (IBMTSOE)
- z/OS UNIX System Services checks (IBMUSS)
- VSAM checks (IBMVSAM)
- VSAM RLS checks (IBMVSAMRLS)
- VSM checks (IBMVSM)
- Cross System Coupling Facility (XCF) checks (IBMXCF)

9.3.1 New health checks with z/OS V1R10

The following are new checks with z/OS V1R10:

- RACF (2 new checks, 1 updated check)
- XCF/XES (9 new checks, 5 updated checks)
- USS (2 new checks)
- SDSF (1 new check)
- Language Environment (1 new check)

9.4 New and changed RACF health checks

There are three checks that are either new or enhanced:

- The RACF_ICHAUTAB_NONLPA check examines the RACF Authorized Caller Table (ICHAUTAB) and reports if there are any non-LPA entries in it. The output format is similar to the report format for the ICHAUTAB Report in RACF_SENSITIVE_RESOURCES, with the exception that LPA-resident modules are not listed.
- This is a new general resource Health check, as shown in 9.4.2, “Write your own RACF resource checks” on page 227.
- The RACF_SENSITIVE_RESOURCES check examines the security characteristics of several system-critical data sets and general resources other than data sets. The output of this check is a list of exceptions flagged. This check has an enhancement in z/OS V1R10, and the check is as follows:

```
IBMRACF,RACF_SENSITIVE_RESOURCES
```

9.4.1 RACF_ICHAUTAB_NONLPA check

The RACF_ICHAUTAB_NONLPA check examines the RACF Authorized Caller Table (ICHAUTAB) and reports if there are any non-LPA entries in it. The output format is similar to the report format for the ICHAUTAB Report in RACF_SENSITIVE_RESOURCES, with the exception that LPA-resident modules are not listed. IBM recommends that installations have no entries in the ICHAUTAB table.

This check raises an exception if you have a non-LPA ICHAUTAB module defined to your environment. The severity of the check is medium and the interval it runs at is once a day.

9.4.2 Write your own RACF resource checks

A new RACF RACFHC class is defined to allow an installation to specify a resource to be checked by Health Checker. This implementation requires the defining of RACF profiles in this new class and a PARMLIB specification to add the health check. The profile contains a list of resources that you want to check along with the maximum allowable general user access you want for each resource.

The ADDMEM field

The ADDMEM field used with the RACFHC class profile defines the resources that you want checked. The format of each member list entry in the profile is as follows:

```
addmem(className/resourceName/volume/maxUacc)
```

className The class of the resource which is to be checked. Valid values are DATASET and any RACF general resource class which is defined on the system.

resourceName The name of the resource which is to be checked.

volume If the className is DATASET then this is a volume upon which the data set resides. This parameter is optional. If it is not specified, then the catalog is searched to find the volume serial for the data set. If the className is not DATASET, do not specify a volume. If you specify a volume for a className other than DATASET, you will receive an error message.

maxUacc The maximum allowed general user access to the resource that is the access level which if exceeded results in an exception. Valid values are NONE, READ, UPDATE, and CONTROL.

Special values you can use in ADDMEM

You can specify any number of resource names up to the maximum amount of data which can be placed into the member list portion of a profile using the ADDMEM operand. Only the following types of data sets are allowed to be specified as resources:

```
Sequential, partitioned, library, or VSAM data sets.
```

To make defining your profile easier, you can also use the special values in ADDMEM shown in Figure 9-1 on page 228.

Value	Description
IRR_APFLIST	Examines all of the entries in the current APF list.
IRR_LINKLIST	Examines all of the entries in the current link list
IRR_PARMLIB	Examines all of the entries in the current PARMLIB
IRR_RACFDB	Examines the current primary and backup RACF databases
IRR_SYSREXX	Examines SYS1.SAXREXEC
IRR_ICHAUTAB	Examines the entries in ICHAUTAB

Figure 9-1 Special ADDMEM values

Sample profile definition for a predefined set of resources

The following RACF command defines a resource class named MY_RESOURCE_LIST:

```
RDEFINE RACFHC MY_RESOURCE_LIST
ADDMEM(DATASET/SYS1.LINKLIST//NONE
IRR_LINKLIST
IRR_RACFDB)
```

Note: If you specify one of these special ADDMEM values, you cannot specify any other value, such as className, resourceName, volume, or maxUacc on that entry.

Example to create a user health check

The following sections describe how to create a health check that checks a specific user's access to a data set. This new RACF RACFHC resource class is implemented for users who wish to check the security of their own resources.

This example uses the following specifications:

```
RACF class:              RACFHC
Check name:              MY_ROGERS_HC
Resourcename             MY_RESOURCE_LIST
DATASET class data set   ROGERS.HCHECK1.JCL
User name                HARRY
Parmlib member           HZSPRMPR
Check OWNER              ROGERS
```

Define the profile to RACF

The RACFHC class contains profiles that have the resources that you want to check. Following is a RDEFINE command to add a profile and that is protecting a DATASET class data set ROGERS.HCHECK1.JCL:

```
RDEFINE RACFHC MY_RESOURCE_LIST
ADDMEM(DATASET/ROGERS.HCHECK1.JCL//NONE
RACFHC/MY_RESOURCE_LIST//NONE)
```

Data set ROGERS.HCHECK1.JCL is to be checked for authorization access by RACF for a user ID whose authority to the resources is listed in the profile the check will examine. This user ID is specified in the health checker routine that is going to be defined in the HZSPRMPR parmlib member, shown in "PARMLIB entry definitions" on page 229.

PARMLIB entry definitions

You can create your own RACF installation-defined resource checks to see if your resources have the security characteristics you want.

For each check you wish to create, choose a name for your RACF installation-defined resource check, and using this name, define the check to IBM Health Checker for z/OS in a HZSPRMxx parmlib member.

Figure 9-2 shows an example of adding a RACF installation-defined resource check in the HZSPRMPR parmlib member. In the definition, the check owner is ROGERS. The check name is MY_ROGERS_HC.

The PARM keyword identifies the user ID, (HARRY), to be checked for access to the data set. The RESOURCELIST defines the resource name specified in the RACFHC profile in “Define the profile to RACF” on page 228.

```
PARM('USER(HARRY) RESOURCELIST(MY_RESOURCE_LIST)')
```

Next, activate the PARMLIB entry that contains profiles that list the resources to check for each installation-defined health check by IBM Health Checker for z/OS.

```
ADD    ,CHECK(ROGERS,MY_ROGERS_HC)
       ,CHECKROUTINE(IRRHCR00)
       ,MESSAGETABLE(IRRHCM00)
       ,ENTRYCODE(100)
       ,PARM('USER(HARRY) RESOURCELIST(MY_RESOURCE_LIST)')
       ,DATE(20080530)
       ,REASON('A SAMPLE CHECK TO DEMONSTRATE A USERCHK')
       ,SEVERITY(HIGH)
       ,INTERVAL(00:10)
       ,GLOBAL
       ,ACTIVE
```

Figure 9-2 Check routine to add to parmlib member

Attention: You must specify the following values for your check:

```
CHECKROUTINE(IRRHCR00)
MESSAGETABLE(IRRHCM00)
ENTRYCODE(100)
```

Add the new health check

To add the health check which was added to member HZSPRMPR in SYS1.PARMLIB, use the following operator command:

```
F HZSPROC,ADD,PARMLIB=PR
```

Figure 9-3 on page 230 shows an SDSF display of the health checks. The user-defined check is shown and pointed to by the arrow.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HEALTH CHECKER DISPLAY  SC70                      LINE 41-60 (128)
COMMAND INPUT ===>                                        SCROLL ===> HALF
ACTION=//-Block,=-Repeat,+-Extend,A-Activate,D-Display,DL-DisplayLong,
ACTION=DP-DisplayPolicies,DPO-DisplayOutdatedPolicies,DS-DisplayStatus,
ACTION=E-Refresh,H-Deactivate,P-Delete,PF-DeleteForce,R-Run,S-Browse,
ACTION=SB-ISPFBrowse,SE-ISPFEdit,U-RemoveCat,X-Print,XC-PrintClose,XD-PrintDS,
ACTION=XDC-PrintDSClose,XF-PrintFile,XFC-PrintFileClose,XS-PrintSysout,
ACTION=XSC-PrintSysoutClose
NP   NAME                            CheckOwner       State              Status
     IEA_ASIDS                       IBMSUP           ACTIVE(ENABLED)    SUCCES
     IEA_LXS                         IBMSUP           ACTIVE(ENABLED)    SUCCES
     IXGLOGR_ENTRYTHRESHOLD          IBMIXGLOGR       INACTIVE(ENABLED)  INACTI
     IXGLOGR_STAGINGDSFULL           IBMIXGLOGR       ACTIVE(ENABLED)    SUCCES
     IXGLOGR_STRUCTUREFULL           IBMIXGLOGR       ACTIVE(ENABLED)    SUCCES
S    MY_ROGERS_HC                    ROGERS           ACTIVE(ENABLED)    EXCEPT
     PDSE_SMSPDSE1                   IBMPDSE          ACTIVE(ENABLED)    SUCCES
     RACF_FACILITY_ACTIVE            IBMRACF          ACTIVE(ENABLED)    SUCCES
     RACF_GRS_RNL                    IBMRACF          ACTIVE(ENABLED)    SUCCES
     RACF_IBMUSER_REVOKED            IBMRACF          ACTIVE(ENABLED)    EXCEPT
```

Figure 9-3 SDSF display of the health checks

Example of how the health check works

The following sections describe how the access check works for the user health check that is added in the parmlib member shown in “PARMLIB entry definitions” on page 229.

- Data set ROGERS.HCHECK1.JCL is to be protected from access by user ID HARRY.
- Currently user ID HARRY has READ access to the data set.
- When the check is added, it then runs when the interval expires. Figure 9-4 on page 231 shows the output from the RACF installation-defined resource check when you select, (S), the health check, as shown in Figure 9-3 on page 230. Because the UACC is defined as NONE in the RDEFINE command, the health check routine displays this exception, (**E**), in the output in Figure 9-4 on page 231.

  This indicates that user ID HARRY has access to the data set when a profile has been created to prevent this access.

```
CHECK(ROGERS,MY_ROGERS_HC)
START TIME: 05/30/2008 13:29:27.577326
CHECK DATE: 20080530  CHECK SEVERITY: HIGH
CHECK PARM: USER(HARRY) RESOURCELIST(MY_RESOURCE_LIST)

                 Resource List from MY_RESOURCE_LIST

S Resource Name                    Class    Vol    UACC Warn ID*  User
- ------------------------------- -------- ------ ---- ---- ---- ----
  MY_RESOURCE_LIST                 RACFHC          None No   ****
E ROGERS.HCHECK1.JCL               DATASET  SBOXA7 None No   **** >None
* High Severity Exception *

IRRH237E The MY_ROGERS_HC check has found one or more potential errors
in the security controls for the installation-defined resources specified
in this check.

  Explanation:  The RACF security configuration check has found one or
    more potential errors with the protection mechanisms for the
    resources specified for this check.

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

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

Figure 9-4 Installation Health Check output

- To remove user HARRY from access to the data set, remove his user ID from the access list with a RACF command.
- Once the access is removed, run the health check. The result is shown in Figure 9-5. Notice that the (**E**) shown in Figure 9-4 is removed.

```
CHECK(ROGERS,MY_ROGERS_HC)
START TIME: 05/30/2008 15:54:45.595447
CHECK DATE: 20080530  CHECK SEVERITY: HIGH
CHECK PARM: USER(HARRY) RESOURCELIST(MY_RESOURCE_LIST)


                 Resource List from MY_RESOURCE_LIST

S Resource Name                    Class    Vol    UACC Warn ID*  User
- ------------------------------- -------- ------ ---- ---- ---- ----
  MY_RESOURCE_LIST                 RACFHC          None No   ****
  ROGERS.HCHECK1.JCL               DATASET  SBOXA7 None No   ****

IRRH238I The MY_ROGERS_HC check has not found any errors in the security
controls for the installation-defined resources specified in this check.

END TIME: 05/30/2008 15:54:45.608869  STATUS: SUCCESSFUL
  administrator and the system auditor.
```

Figure 9-5 User health check after modifying the user's access to the data set

9.4.3 RACF_SENSITIVE_RESOURCES check

The enhancement to this existing health check is to issue a new message, IRRH204E, shown in Figure 9-6.

```
IRRH204E The RACF_SENSITIVE_RESOURCES check has found one or more potential
errors in the security controls on this system.
```

Explanation: The RACF security configuration check has found one or more potential errors with the system protection mechanisms.

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

Operator response: Report this problem to the system security administrator and the system auditor.

Figure 9-6 New message added to the health check

9.5 New and updated XCF Health Checks

There are nine checks that are new with z/OS V1R10, as follows:

- **Check(IBMXCF,XCF_CDS_SPOF)**

  This check raises an exception when a primary and alternate couple data set has single point of failure. It checks all of the couple data set types that are in use on the system.

  Severity (High) - Interval(1:00)

  If the check is run with Verbose=YES, a couple data set report is generated even when no single point of failure is found.

- **Check(IBMXCF,XCF_CF_Allocation_permitted)**

  This check raises an exception when a CF can not be used for structure allocation.

  Severity (Medium) - Interval(4:00) - Global

- **Check(IBMXCF,XCF_CF_Str_Availability)**

  This check raises an exception when the current definition of a structure does not contain at least 2 CFs that are online and in separate CECs in its preference list.

  Severity (Medium) - Interval(8:00) - Global

  If the check is run with Verbose=YES, the check will report on all defined structures.

- **Check(IBMXCF,XCF_CF_Str_Duplex)**

  This check raises an exception when an allocated structure is defined as DUPLEX(ENABLED) or DUPLEX(ALLOWED), but is not duplexed.

  Severity (Medium) - Interval(1:00) - Global

  If the check is run with Verbose=YES, the check will also report on structures that are duplexed.

- **Check(IBMXCF,XCF_CF_Str_NonVolatile)**

  This check raises an exception when a allocated structure is in a volatile CF, but a connector requested a non-volatile structure.

  Severity (Medium) - Interval(8:00) - Global

  If the check is run with Verbose=YES, the check will report on all allocated structures.

- **Check(IBMXCF,XCF_CF_Sysplex_Connectivity)**

  This check raises an exception when the number of CFs connected to all of the systems in the sysplex falls below the required minimum number of CFs.

  PARM: MINCFS(2) - Severity (Medium) - Interval(1:00) - Global

  If the check is run with Verbose=YES, the check will report on all coupling facilities defined in the CFRM active policy.

- **Check(IBMXCF,XCF_SFM_ConnFail)**

  This check raises an exception when the Sysplex Failure Management (SFM) policy does not match the recommended action for loss of signal connectivity.

  PARM: CONNFAIL(YES | NO) - Severity (Medium) - Interval(4:00) - Global

- **Check(IBMXCF,XCF_SFM_SSumLimit)**

  This check raises an exception when the Sysplex Failure Management (SFM) policy does not specify a SSUMLIMIT attribute that matches the recommended value.

  PARM: SSUMLIMIT(60) - Severity (Medium) - Interval(4:00)

- **Check(IBMXCF,XCF_SFM_Sum_Action)**

  This check raises an exception when the Sysplex Failure Management (SFM) policy does not specify the recommended indeterminate status actions.

  PARM: SUMACTION(ISOLATE) SUMINTERVAL(0) - Severity (Medium) - Interval(4:00)

There are five checks that are updated with z/OS V1R10, as follows:

- **Check(IBMXCF,XCF_CDS_Separation)**

  This check raises an exception if any of the primary CFRM, SYSPLEX, or LOGR couple data sets are on the same volume.

  A new parameter has been added to this check to make the LOGR couple data set optional in the list of primary couple data sets that should reside on unique volumes.

  PARM: LOGR(NO | YES)

  APAR OA22931 allows this check to be available with z/OS V1R8 and z/OS V1R9

- **Check(IBMXCF,XCF_CF_Str_PrefList)**

  This check raises an exception when a structure is not allocated in the first CF in its preference list. The messages have been enhanced to explain why a structure may have been allocated differently than the order specified in the preference list, and why this may or may not need correcting.

  Note: This check has also been changed from a single system (local) check to a sysplex-wide (global) check.

- **Check(IBMXCF,XCF_SFM_Active)**

  This check raises an exception when the Sysplex Failure Management (SFM) does not match the specified value.

  Note: This check has also been changed from a single system (local) check to a sysplex-wide (global) check.

- **Check(IBMXCF,XCF_Sig_Path_Separation)**

  This check raises an exception when a single point of failure is found in the signaling paths to all systems which are connected. Additional support has been added to this check to verify that XCF signaling paths using Coupling Facility structures, if used, provide at least two signaling structures, and that they reside in different Coupling Facilities in different physical CECs.

- **Check(IBMXCF,XCF_Sig_Str_Size)**

  This check raises an exception when the XCF signaling list structure is too small. A parameter has been added to this check to define the number of systems against which the signaling structure should be verified against. The parameter is as follows:

  PARM: SYSTEMS(ACTIVE | MAXSYSTEM)

9.6 New z/OS UNIX System Services health checks

There are two new USS checks, as follows:

- **Check(IBMUSS, USS_Parmlib_Mounts)**

  This check raises an exception when file systems specified in the BPXPRMxx parmlib members to be mounted during initialization were not mounted successfully.

  Severity - High - Interval(ONETIME)

- **Check(IBMUSS, USS_Client_Mounts)**

  This check raises an exception when a file system could be mounted locally but is not.

  Severity - Medium - Interval(1:00)

9.7 New SDSF health check

There is one new SDSF check, as follows:

- **Check(IBMSDSF, SDSF_ISFPARMS_IN_USE)**

  This check issues an exception if the specified SDSF server is started, and a customized ISFPARMS was used.

  The ISFPARMS customization report then lists differences between the defaults and the installation ISFPARMS.

  SEVERITY(LOW) - PARMS: SERVER(SDSF) - INTERVAL(ONETIME)

  This check is defined when the SDSF server starts. If the server is not being used, use the EXIT statement in PROGxx parmlib member. Also, see ISF.SISFJCL(ISFSPROG). If Verbose(YES) is specified, the check will list every keyword in ISFPARMS regardless of whether it has been customized.

  APAR PK59629 makes this check available with z/OS V1R8 (HQX7730) and z/OS V1R9 (HQX7740).

9.8 New Language Environment health check

There is one new Language Environment check, as follows”

- **Check(IBMCEE,CEE_Using_LE_Parmlib)**

  This check issues an exception if the CEEPRMxx parmlib member was not used to set the default Language Environment run-time options.

  SEVERITY(LOW) - INTERVAL(ONETIME)

Note: For additional information on the new and updated health checks, see *IBM Health Checker for z/OS User's Guide*, SA22-7994.

10

JES3 enhancements

This chapter deals with the following:

- Understanding the authorization changes in JES3 SSI calls.
- Understanding how to use spool browse in JES3.
- Describe the differences between JES2 and JES3 spool browse.
- Describe and make use of extended status changes in JES3.
- SDSF support for JES3.

10.1 JES spool browse

Spool data set browse (SDSB) is a function application program that can be invoked to process spool data sets. JES provides an interface that application programs can use for this purpose. SDSB can be used to access data sets that are still open in running address spaces. Any full data buffers that have been written to spool can be read and, optionally, data that has not been written (unwritten buffer support) can also be accessed (provided the application is running on the same system where the target data set is open).

Programs can use SDSB to allocate spool data sets. The data set being requested is specified by passing the JES data set name along with a spool browse token to MVS dynamic allocation.

10.1.1 JES3 spool browse

Spool browse, sometimes referred to as Spool Data Set Browse (SDSB) is implemented as a new function with z/OS V1R10. The need for spool browse is to avoid the single user lock that the SYSOUT Application Programming Interface (SAPI) imposes on applications.

Under SAPI, when an application makes a PUT/GET call to access a data set, as shown in Figure 10-1, it causes JES3 to schedule that data set to the requesting application until the application releases the data set. As a result, no other application can access this data set during this time. In addition, SAPI is limited to either output belonging to jobs that have finished running, or output that has been spun off and closed by a job that is still running. SAPI cannot access any other output for an active job, even in completed steps.

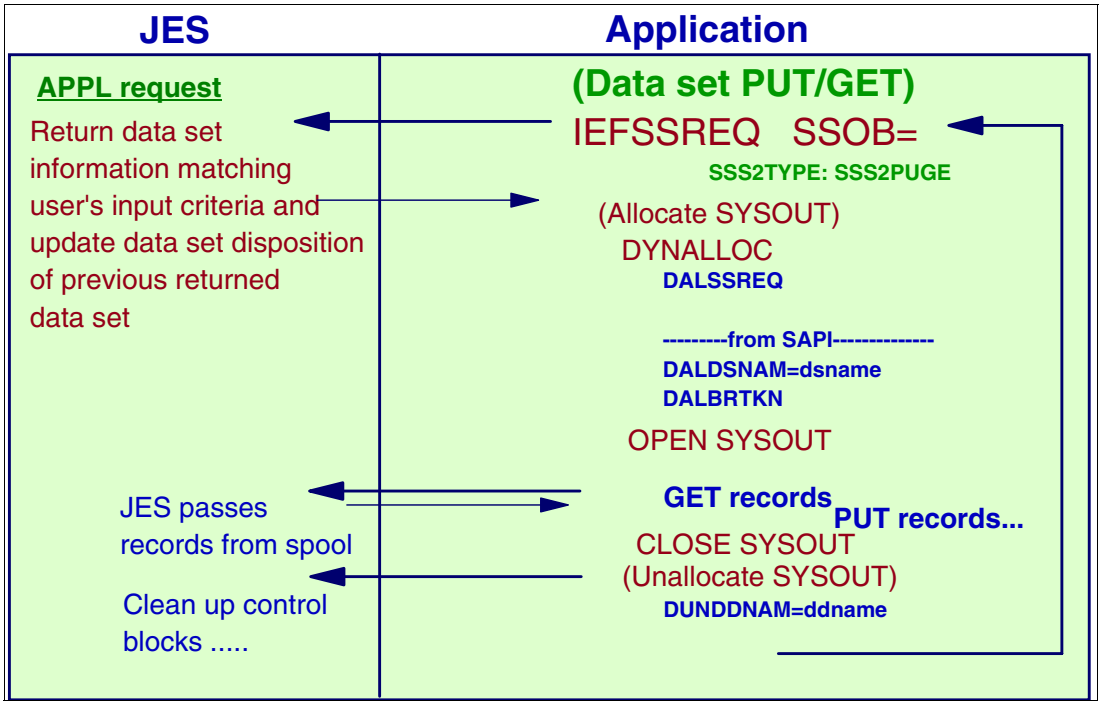


Figure 10-1 SYSOUT Application Programming Interface (SAPI) requests

Spool browse access

The new JES3 spool browse does not have these limitations of the SAPI implementation. Using the spool browse interface, an application can use SVC 99 to allocate a SYSOUT data

set without going through SAPI to obtain access to it. The benefits are more transparency between the JESs, and the ability to access output for a job while it is still running.

Using the spool browse interface any application can access a spool data set that the user of the application is authorized to without requiring the application to go through SAPI to lock the data set. The application can look at output for a job that has ended or for a job that is still active. Output is available before the job ends or closes and spins off data.

Spool browse is invoked using the following steps:

- Identify a data set (such as by using extended status or by specifying a predetermined name).
- Use dynamic allocation to build a request block and text units containing the data set name determine in the first step, and allocate the data set.
- Open the data set, read records from it.
- When done reading records, close the data set and unallocate the DD associated with the data set.

Data sets to browse

In order to browse a spool data set you need to know enough information about its name to specify it to dynamic allocation. If you know the full five-part data set name you can use that. If the data set is one of the JES logging data sets—JESMSGLG, JESJCL, or JESYSMSG—you can take a short cut and leave out the data set number qualifier. You can also specify the special name JCL or its synonym, JESJCLIN, which puts instream (SYSIN) data together with the JESJCL data set to retrieve the complete original JCL stream.

The data set takes the form userid.jobname.jobid.Dnnnnnnn.dsname. dsname comes from the JCL as follows:

```
//DDNAME DD SYSOUT=c,DSNAME=&dsname
```

If the data set name is not specified in the JCL, a “?” is used.

Special data sets can be used as follows:

```
userid.jobname.jobid.JCL (alternatively, JESJCLIN)
userid.jobname.jobid.JESMSGLG
userid.jobname.jobid.JESJCL
userid.jobname.jobid.JESYSMSG
```

Note: JES3 does not concatenate spun-off JESMSGLG or JESYSMSG.

Specifying data set names with wild cards

If you do not know the entire data set but you know some information about it, you can use wild cards. For the data set number you can substitute a single asterisk as a place holder. For the data set name you can substitute a pattern consisting of an asterisk to signify zero or more characters in the given position, or a question mark to signify exactly one character in the given position. You cannot use a pattern for userid, jobname, and jobid.

When you use a pattern for a data set name, the first data set in the job that matches the pattern is always used. Some examples of the use of wild cards are:

- An * can be used as a place holder for Dnnnnnnn.
- A data set name can be specified as a pattern, as follows:

```
userid.jobname.jobid.*.pattern (pattern for &DSNAME)
```

Wildcard characters are as follows:

- ? for exactly one character
- * for zero or more characters

In the data set name, for the userid, jobname, and jobid, you cannot specify wild cards. Only the first data set in the job matching the pattern will be considered the true data set for allocation.

For example, JOB1.NET.J0000056.D000000A.CHECKS will match these patterns:

```
JOB1.NET.J0000056.*.C*S
JOB1.NET.J0000056.*.CH?CKS
```

10.2 JES3 V1R10 SSI enhancements

One of the biggest inhibitors to using the Subsystem Interface has been the fact that SSI calls require running in supervisor state, which in turn requires APF authorization. This means an application coder must go to a system programmer to install the application in an APF library. User-written APF programs are a security risk. They could accidentally or maliciously compromise system integrity or step on something that is protected from normal programs, possibly causing an outage. Even if a user program is risk free, the system programmer must still ascertain that fact when installing an initial version or later versions of that program. An application therefore cannot just be coded and then executed. It is necessary to go through the system programmer staff which could cause it take longer to test and debug the application.

The SSIs 79, 80, 75, and 11 are enhanced to no longer require APF authorization. Starting in z/OS V1R10 JES3 will support these SSIs in an unauthorized environment.

In addition, the Subsystem Interface text unit used on dynamic allocation for a SYSOUT data set, DALSSREQ, requires APF authorization. A new text unit, DALUASSR, is provided for the unauthorized environment. This change works with the new JES3 spool browse.

Note: For the APF alternative, the application can use SSI 79, 80, 75, or 11 the way it has done before. A MODESET call is no longer needed. Existing MODESETs can be removed. A load module does not have to reside in an APF library. It can reside in a user's private library.

10.2.1 Using extended status to request data sets (SSI 80)

The extended status function call (SSI function code 80) allows a user-supplied program to obtain detailed status information about jobs and SYSOUT in the JES queue. Both JES2 and JES3 subsystems support job status information.

Extended status is more than just an extension to the original STATUS SSI. It is intended as a programming interface into both JES2 and JES3 to obtain not only status information, but general information about jobs and SYSOUT. It can be used to obtain information to present on a front end such as SDSF, or to programmatically check the status of jobs. It can also be used as a screener for an SAPI print application to select work to process next.

10.2.2 Issue an SSI 80 request

With z/OS V1R10, this request can be issued from an unauthorized requester. The IEFSSREQ macro is used to issue the request pointing at an SSOB and SSIB. The SSOB extension is mapped by the IAZSSST macro in which you specify the SSST fields shown in Figure 10-2.

The extended status interface is designed to be a general purpose interface to obtain information from JES. Callers use the STATTYPE field to indicate the type of data they require. This SSI call returns job information and SYSOUT status for information that is stored on the JES spool.

You can use extended status to ask JES3 to provide a list of data sets from which you can select the one you want. Before z/OS V1R10, the JES3 extended status function showed only data sets for jobs that had ended, or closed spun-off data sets written by active jobs.

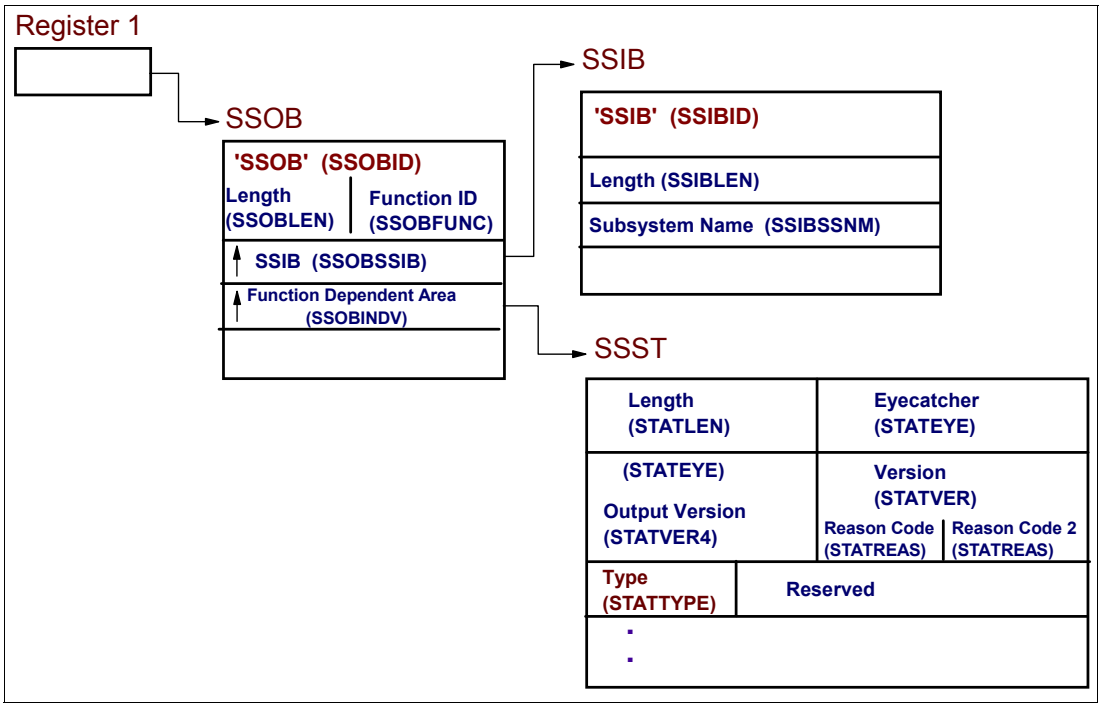


Figure 10-2 SSI function code 80 call using IEFSSREQ

The STATTYPE field

The extended status interface is designed to be a general purpose interface to obtain information from JES. Callers use the STATTYPE field to indicate the type of data they require.

Table 10-1 STATTYPE function to be performed on a request

Field Value	Description
STATTERS	Requests obtaining terse job level data. The data returned on this call does not require large amounts of system overhead.
STATVRBO	Requests obtaining verbose job level data. The data returned on this call includes that returned for the terse job level call. STATVER must be set to at least STATV040 for this request to be valid. The request requires system overhead for I/O to obtain the data.

Field Value	Description
STATMEM	Return memory from a previous request. After one or more requests for data, the memory obtained must be returned using this function.
STATOUTT	Requests obtaining terse SYSOUT level data (including job level information). Data returned on this call does not require large amounts of system overhead. STATVER must be set to at least STATV030 for this request to be valid.
STATOUTV	Requests obtaining verbose SYSOUT level data. The data returned on this call includes that returned for the terse SYSOUT level call. STATVER must be set to at least STATV040 for this request to be valid. The request requires system overhead for I/O to obtain the data. Terse and verbose output for running jobs is also returned.
STATDLST	Requests data set list for a job. This request obtains verbose type information for all data sets associated with a job. It includes information on SYSIN and other internal data sets. It is only valid for STATV060 and above callers.

The STATOUTV flag

To obtain verbose SYSOUT level data, the extended status request is invoked by setting the STATOUTV flag in field STATTYPE of macro IAZSSST and also setting one of the following input fields:

- Set STATTRSA to zero and ensure that the job ID filters specified by STATSJBI refer to the same job ID in STATJBIL and STATJBIH (or that STATJBIH is set to zero). Both terse and verbose data are returned. Verbose data is also returned for all valid SYSOUT data sets (chained into the STATJQ). If the job is still executing, STATVOs for data sets that are still open may also be returned. Lastly, terse SYSOUT data is returned.
- Set STATTRSA to zero and ensure that STATSCTK has STATCTKN set to the SYSOUT token of the SYSOUT group for which you want verbose data. Both terse and verbose job and SYSOUT data are returned (only for the data sets represented by the token passed).
- Set STATTRSA to a STATJQ (obtained previously with no intervening memory management call). Similar to the case in which STATSJBI is set, verbose job data will be chained into the STATJQ, STATVOs are obtained for all valid SYSOUT data sets, and STATSEs are obtained for all SYSOUT groups for the job.
- Set STATTRSA to a STATSE (obtained previously with no intervening memory management call). Similar to the case in which STATSCTK is set, verbose job data is obtained for the job, and all the STATVOs related to the STATSE.

z/OS V1R10 extended status

Starting in z/OS V1R10, extended status shows data sets that are currently open, or that were created in steps that have previously ended, even if the data sets are not closed and spun off. Limited information on these data sets is provided; for example information such as (FORMS and CHARS) that come from combinations and overrides of SYSOUT, OUTPUT, //*FORMAT, and initialization statements does not get built until the job ends or spins off the data set.

Using output verbose status calls

A verbose SYSOUT request returns information relevant to particular SYSOUT data sets in a job.

Note: Using the extended status SSI, the verbose output function with z/OS V1R10 now shows all data sets for steps that have run, or are running now. The data set name is in the STVSDSN field.

Therefore, the data set name you can supply to the dynamic allocation is available in the verbose output field STVSDSN. In addition to verbose output there is a new function code, STATDLST.

10.2.3 Sample job to illustrate data set names

The sample JCL shown in Figure 10-3 illustrates how extended status returns data for this job, which runs two steps that produce output and then waits in the third step for an operator reply.

```
//JOB1 JOB MSGCLASS=T
//GENER1   EXEC     PGM=IEBGENER
//SYSPRINT DD       DUMMY
//SYSIN    DD       DUMMY
//SYSUT2   DD       SYSOUT=*
//SYSUT1   DD       *
This is the first instream data set.
/*
//GENER2   EXEC     PGM=IEBGENER
//SYSPRINT DD       DUMMY
//SYSIN    DD       DUMMY
//SYSUT2   DD       SYSOUT=*
//SYSUT1   DD       *
This is the second instream data set.
/*
//WTORSTEP   EXEC   PGM=WTOR
//STEPLIB    DD     DSN=ROGERS.UTIL.LOAD,DISP=SHR
```

Figure 10-3 Sample JCL job

Select a data set using the STATOUTV function with z/OS V1R10

The data sets returned by the STATOUTV function are the three standard message data sets and the two SYSUT2 data sets from the IEBGENER steps. Remember that STATOUTV must limit the selection to a single job. Starting in z/OS V1R10, JES3 supports using the STATTRSA pointer field to expand a terse job or output element returned by a previous extended status terse call into verbose sections, as shown in Figure 10-4 on page 244. STATTRSA is new in JES3 in z/OS V1R10.

Note: In order to expand a terse job or output element and build verbose data off it, the system that the application is running on must be at z/OS V1R10 JES3 or higher. Unlike most subsystem interface functions in JES3, the global can be z/OS V1R8, or V1R9, and does not need to be at z/OS V1R10.

```
STVSDSN                        STVSPRCD    STVSSTPD   STVSDDND
ROGERS.JOB1.JOB00456.D0000002.JESMSGLG                 JESMSGLG
ROGERS.JOB1.JOB00456.D0000003.JESJCL                   JESJCL
ROGERS.JOB1.JOB00456.D0000004.JESYSMSG                 JESYSMSG
ROGERS.JOB1.JOB00456.D000000B.?             GENER1     SYSUT2
ROGERS.JOB1.JOB00456.D000000C.?             GENER2     SYSUT2
```

Figure 10-4 Data set names and fields returned using the STATOUTV function

Select a data set using the STATDLST function with z/OS V1R10

A new extended status function is STATDLST, which builds a data set list. It is similar to STATOUTV and returns similar information except that it also builds information for internal and instream (SYSIN) data sets. A list of all data sets associated with a single job can be requested by setting STATTYPE to STATDLST. Since this is considered a verbose type call (I/O is required to obtain the needed information), only information about a single job can be requested (STATTRSA is supported).

Note: A difference between JES2 and JES3 is that in JES2, one terse verbose element and one verbose element is returned for each data set. In JES3, terse elements can have a group of verbose elements (same as STATOUTV).

The rules on using STATDLST are the same as for STATOUTV. The request must be limited to a single job. This can be specified as a job range of one job, a client token, or a STATTRSA expansion.

Note: In order to use STATDLST, the global and the system on which the application is running must be z/OS V1R10 or higher.

Using the same JCL as in the earlier STATOUTV example shown in Figure 10-3 on page 243, Figure 10-5 shows what the STATDLST function returns. The internal data sets created when the job is submitted are returned, and the input SYSUT1 data sets for the two steps are returned.

```
STVSDSN                        STVSPRCD     STVSSTPD     STVSDDND
ROGERS.JOB1.JOB00456.D0000001.JESJCLIN                   JESJCLIN
ROGERS.JOB1.JOB00456.D0000002.JESMSGLG                   JESMSGLG
ROGERS.JOB1.JOB00456.D0000003.JESJCL                     JESJCL
ROGERS.JOB1.JOB00456.D0000004.JESYSMSG                   JESYSMSG
ROGERS.JOB1.JOB00456.D0000005.J3SCINFO                   J3SCINFO
ROGERS.JOB1.JOB00456.D0000006.J3JBINFO                   J3JBINFO
ROGERS.JOB1.JOB00456.D0000007.JCBLOCK                    JCBLOCK
ROGERS.JOB1.JOB00456.D0000008.JOURNAL                    JOURNAL
ROGERS.JOB1.JOB00456.D0000009.?              GENER1      SYSUT1
ROGERS.JOB1.JOB00456.D000000A.?              GENER2      SYSUT1
ROGERS.JOB1.JOB00456.D000000B.?              GENER1      SYSUT2
ROGERS.JOB1.JOB00456.D000000C.?              GENER2      SYSUT2
```

Figure 10-5 Data set names and fields returned using the STATDLST function

Allocate the selected data set

Once a data set is selected, the next step is to allocate it using SVC 99. As with any other dynamic allocation you do this by building an S99RB request block and a text unit list.

The required allocation text units consist of:

- DALDSNAM (data set name) containing the data set name or pattern
- A choice of DALDDNAM (to specify your own DD name) or DALRTDDN (to ask dynamic allocation to return a unique system generated DD name)
- The subsystem request text unit; for requests that are running with APF authorization use the text unit DALSSREQ.

Where:

DALDSNAM Used with the returned name from the SSSODSN field.

If you cannot specify the exact data set name, use STVSDSN, which is returned by extended status.

DALSTATS Specify DALSTATS of SHR, which is x'08'.

DALBRTKN The browse token. Before issuing the dynamic allocation request, the application must build the browse token and pass it with the DALBRTKN text unit. The format of the browse token is mapped by macro IAZBTOKP. The browse token is built in text unit format with seven subparameters.

DALSSREQ Indicates a request that JES needs to handle. The parameter value in this text unit is the name of the subsystem that processed the IEFSSREQ macro. DALSSREQ or DALUASSR (subsystem name) is the unauthorized version of DALSSREQ. They are mutually exclusive.

DALRTDDN Indicates that the DDNAME associated with the allocation be returned to the caller of DYNALLOC. Your external writer then places this DDNAME in the DCB macro that needs to open the SYSOUT data set as input for your reads. Prime the parameter in this text unit with blank (X'40') characters before issuing the DYNALLOC macro. This text unit is returned from DYNALLOC with the correct DDNAME.

Browse token

You must construct a browse token consisting of seven subparameters each of which contains a distinct length and value. All seven subparameters must be specified or allocation will reject the call and not pass it to JES. If a subparameter is not known or not used, code a placeholder of zeroes and an appropriate keyword length.

BTOKPL1 Length of the browse token identifier (LENGTH(BTOKID)).

BTOKID - Browse token ID (BTOK). The IAZBTOKP macro defines constant BTOKCID to be used to set this field.

BTOKPL2 Length of the token version field (LENGTH(BTOKVER)).

BTOKVER - The 2-byte version number of the token parameter list. Byte 1 (or TOKTYPE) indicates the call type. If it is set to BTOKBRWS, then this is a normal browse request. If it is set to BTOKSTKN, then this is a SPOOL token based browse request. JES3 supports only BTOKSTKN. Byte 2 (or BTOKVERS) is the parm list version and should be set to BTOKVRNM.

BTOKPL3 Length of the IOT field LENGTH(BTOKIOTP).

BTOKIOTP - Data pointer whose content is based on the first byte of BTOKVER.

If BTOKVER is set to BTOKBRWS, this is the optional MTTR of the IOT containing the PDDB of the file to be allocated (obtained from JOEIOTTR or IOTTRACK, for example). JES3 does not support this form of BTOKIOTP.

If BTOKVER is set to BTOKSTKN, this must point to either a client token (returned from dynamic allocation using key DALRTCTK) or a data set token (returned by the SAPI SSI in field SSS2DSTR or the Extended Status SSI in field STVSCTKN).

BTOKPL4 Length of the job key field (LENGTH(BTOKJKEY)).

BTOKJKEY - Optional job key of the file to be allocated (for example, the job key obtained from JCTJBKEY, JQEJBKEY, or SJBJKEY). This field is not used if BTOKVER is set to BTOKSTKN. This field is required if BTOKTYPE is set to BTOKBRWS and BTOKIOTP is non-zero. JES3 does not support this parameter and this field is set to zero.

BTOKPL5 Length of the ASID field (LENGTH(BTOKASID)).

BTOKASID - The 2-byte ASID of the data set owning job if the job is active on the same system and active buffers are needed. If active buffers are not needed, then pass 0. If the ASID is not known, then pass X'FFFF' and JES will determine the correct ASID.

BTOKPL6 Length of the RECVR field (LENGTH(BTOKRCID)).

BTOKRCID - Eight-byte userid to be used as the RECVR on the SAF call or zeros if the RECVR is not being used. JES uses this field to check authority to the browse request. When RECVR is used, the value must be left justified and padded with blanks. When this parameter is specified, the logstr field should also be used so that usage of recvr can be logged. However, neither JES nor SAF enforces this convention.

BTOKPL7 Length of the logstr field (LENGTH(BTOKLOGS)).

BTOKLSDL - Length of the logstr (specified in field BTOKLSDA) to be used on the SAF call used by JES to check authority to the browse request, or zero if the logstr is not being used. The logstr length must be a value from 0 to 254. BTOKLSDA Text of the logstr if BTOKLSDL is non-zero, or zeros if the logstr is not being used. The maximum length text is 254 characters.

Note: When you use the compatibility interface to read the data set, you could also use text units specifying the record format, record length, and blocksize.

Open the allocated data set

After the data set is allocated, the processing is then the same as you would do with SAPI. The data set can be read using one of two methods:

1. Using the compatibility interface (DCB, GET)

   Using the compatibility interface, fill in the DCB with the logical record length and record format of the data set. They can be obtained from the extended status fields STVSMLRL and STVSRECF, respectively.

2. Using the ACB/RPL interface

   Use the compatibility interface when synchronous sequential access is required. Use the ACB/RPL interface when random access to the spool file is required, or when asynchronous processing is required.

After doing the OPEN you can GET records as you would with SAPI.

When finished with the data set, CLOSE the DD and call SVC 99 again to free (unallocate) the DD.

10.3 SDSF support for JES3

SDSF now operates in a JES3 environment. The job-related panels (DA, I and ST) show data for JES3 jobs. Other SDSF panels that do not depend on JES are also available in the JES3 environment, as follows:

- Enhancements to the support for the REXX programming language. SDSF simplifies specifying the field list to use when accessing a panel, allows you to obtain all of the column values for a specific row, supports an unlimited number of filters and lets you define the relationship between filters (AND or OR).
- A new FILTER parameter on the QUERY command to display the values for SDSF filters.
- New columns on several SDSF panels.
- A new LL action character on the DA panel.
- Addition of a check for use with IBM Health Checker for z/OS. This check determines whether you are using the assembler macro ISFPARMS rather than parmib member ISFPRMxx for SDSF configuration.
- Elimination of the need for the SDSF sysplex support, which uses the SDSF server and WebSphere MQ, to see the latest data (data not yet written to spool) when browsing a JES2 job that is running on a system other than the one you are logged on to. The SDSF sysplex support is still required to see the latest data when browsing the SYSLOG for a system other than the one you are logged on to, and for sysplex-wide device panels. The sysplex support is not available in a JES3 environment.

10.3.1 JES3 operation in SDSF

SDSF now operates in the z/OS JES3 environment. These job-related panels show data for JES3 jobs:

- Display Active Users (DA)
- Input Queue (I)
- Status (ST)

Note: SDSF may be invoked on either a local or global processor running z/OS V1R10 JES3. When SDSF is invoked on a local processor, the global processor must also be at the z/OS V1R10 JES3 level.

You can use action characters to manage the jobs and to list and browse their data sets. In addition, you can modify the attributes of JES3 jobs by overtyping columns on SDSF panels.

The following SDSF panels that do not depend on JES are also available in the JES3 environment, and are shown in Figure 10-6 on page 248:

- Enclaves (ENC)
- IBM Health Checker for z/OS (CK)
- Operlog (LOG)
- Processes (PS)
- Scheduling environments (SE)
- System requests (SR)
- WLM resources (RES)
- User session log (ULOG)

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
 HQX7750 -----------------  SDSF PRIMARY OPTION MENU  -------------------------
 COMMAND INPUT ===>                                          SCROLL ===> HALF

 DA    Active users                      CK    Health checker
 I     Input queue
 ST    Status of jobs                    ULOG  User session log

 LOG   System log
 SR    System requests
 SE    Scheduling environments
 RES   WLM resources
 ENC   Enclaves
 PS    Processes

 END   Exit SDSF
```

Figure 10-6 SDSF Primary JES3 menu

10.3.2 SDSF check for IBM Health Checker for z/OS

SDSF adds a check named SDSF_ISFPARMS_IN_USE for use with IBM Health Checker for z/OS. This check determines whether you are using the assembler macro ISFPARMS rather than the ISFPRMxx parmlib member for SDSF configuration. IBM recommends that you use parmlib member ISFPRMxx. If you are not using the SDSF server, you must update the PROGxx parmlib member to define the SDSF check to the dynamic exit facility.

11

Global resource serialization

This chapter describes the following changes to global resource serialization (GRS):

- Performance monitoring enhancements
- GRSRNL=EXCLUDE migration to full RNLs
- GRS IPCS enhancements

11.1 Performance monitoring enhancements

The main objective of the global resource serialization monitor tool is to assist in planning the RNLs for global resource serialization implementation. The tool monitors ENQ, DEQ, and RESERVE requests, and collects data about the resources and the requesters.

Currently, GRS latch performance for non-zero key calls, such as RRS, is substantially improved. The GRS monitor's performance was dramatically improved. However, GRS continues to have a health check that warns users not to run the monitor for long periods of time on production systems. The health check was added for the following reasons:

- Prior to z/OS V1R9, having the monitor on caused some ENQs to take different paths through non-monitor ENQ processing. This exposed some errors in user code because some ENQ paths enforced/expected the exploiter to abide by certain rules whereas other paths were more forgiving. In z/OS V1R9 and above, all ENQ paths followed the more forgiving rules.
- Having the monitor on causes performance overhead on every ENQ, DEQ, RESERVE, and ISGENQ in the system.

11.1.1 New monitor performance enhancements

With z/OS V1R10, there is a reduction in the path length and hardware cache misses in critical GRS paths. This includes GRS latch, ENQ/GQSCAN related paths, and I5Z: CMSEQDQ lock contention. Also provided is a monitor filter for collecting only RESERVE activity to ensure that the monitor runs at the right dispatch priority. These changes should make running the monitor more practical from a performance impact perspective. Consider the following recommendations:

- Even though the performance was dramatically improved in R10, it still increases the path length. As such, the health check remains just to make it aware that it is on. It is recommended to use the monitor when needed but not have it on unless an installation feels it is justified. New information in *z/OS MVS Planning: Global Resource Serialization*, SA22-7600 that suggests that if you are concerned about the performance overhead, you should measure it via the ISGEQRSP sample program.
- In previous releases, not having the monitor set to the correct dispatch priority, since it needs to be SYSSTC, can cause system problems where high priority ENQs would be blocked by low priority monitor work.

  The monitor now joins a GRS-dependent enclave which insures that it is run at the same dispatch priority as GRS. However, note that its CPU time is now also added to the GRS address space's overall CPU time.

New filter option

A new filter option, REQTYPE=NCRESERVE, is added for the GRS monitor, as follows:

```
GFLG FILTER=Y/N,MATCH=Y/N
REQTYPE = ALL (REQUESTs that pass other filtering)
REQTYPE = NCRESERVE
```

This new REQTYPE only gathers non-converted reserves and only gathers requests that result in actual hardware reserves and pass other filtering options. The default is REQTYPE=ALL.

The GRS monitor filter table is defined by updating a modified copy of the sample filter member ISGAMF00 contained in SYS1.SAMPLIB, assembling it, and then issuing a monitor command to make it active.

Note: You may want to remove the old data first by restarting the monitor with the new filter table rather than just modifying it. ISGAMF08 is the default filter table.

Monitor example

The following example only collects events that result in a hardware reserve for RESERVEs issued against the SYSZVVDS resource.

```
GFLG FILTER=Y,
REQTYPE=NCRESERVE,MATCH=Y
SVCF /*this statement is required*/
NAME N=SYSZVVDS,T=M,L=8
```

Remember that changing your filter options does not clear what was already collected by the monitor. As such the data can be misleading. You may need to restart the monitor with the new filter table if you do not want the old data in the reports.

With this new option, the REQTYPE=NCRESERVE specifies that only non-converted reserves, ones that result in actual device reserves and whose corresponding ENQ has a QNAME of SSYZVVDS are to be traced and collected. The previously existing MATCH=Y keyword indicates that the NAME filter must match as well, as follows:

```
NAME N=SYSZVVDS,T=M,L=8
```

The T=M indicates that N= is for a major or qname, and N= SYSZVVDS indicates that the QNAME is SYSZVVDS. The L=8 indicates the length of the major name to check for a match.

Note: A names list is not required. When the NAME keyword is not provided, only the FILTER and REQTYPE keywords are used to determine whether an event should be traced.

Monitor exploiters

All GRS latch exploiters benefit. Some of the larger users of the monitor are UNIX System Services, RRS, System Logger, IOS, WLM, and SMB.

Note: The monitor provides benefit for both RING and STAR mode, but not GRS=NONE images.

11.2 GRSRNL=EXCLUDE migration to full RNLs

Currently, GRS provides the ability to change RNLs dynamically. If a system is IPLed in GRSRNL=EXCLUDE mode, then a dynamic RNL change cannot be performed. If it is desired to migrate from GRSRNL=EXCLUDE to full RNLs, this requires a complex-wide sysplex outage re-IPL.

With z/OS V1R10, migrating from a GRSRNL=EXCLUDE environment to full RNLs no longer requires a sysplex-wide IPL for certain environments. A new FORCE option for the SET GRSRNL=xx command processing now can dynamically switch to the specified RNLs and move all previous requests to the appropriate scope (SYSTEM or SYSTEMS) as if the RNLs

were in place when the original ENQ requests were issued. This feature eliminates the IPL of the entire sysplex when making this change.

Note: This new feature can be implemented back to z/OS V1R8 and z/OS V1R9 via APAR OA22578.

Usage restrictions

The usage restrictions are enforced to ensure data integrity. The migration is canceled if any of these requirements are not met and GRSRNL=EXCLUDE remains in effect. Consider the following:

- When in GRS Ring mode, the ISG248I message is issued indicating that the SET GRSRNL command is not accepted in a GRSRNL=EXCLUDE environment.
- When in GRS Star mode, before the SET GRSRNL=xx command migration begins, message ISG880D prompts out, asking the user to confirm the use of the FORCE option, as follows:

```
ISG880D WARNING: GRSRNL=EXCLUDE IS IN USE. REPLYING FORCE WILL RESULT IN THE
USE OF SPECIFIED RNLS. REPLY C TO CANCEL
```

The following restrictions are required for data integrity:

- Can only be issued in a single system GRS STAR sysplex with GRSRNL=EXCLUDE.
- No ISGNQXIT or ISGNQXITFAST exit routines can be active nor requests that were affected by one.
- No ISGNQXITBATCH or ISGNQXITBATCHCND exit routines can be active.
- No local resource requests can exist that must become global where that global resource is already owned. This could result if some requests were issued with RNL=NO.
- No resources with a MASID request can exist where only some of the requesters will become global. This could result if some requests were issued with RNL=NO or if some requests were issued with different scopes.
- No RESERVEs can be held that were converted by the ISGNQXITBATCH or ISGNQXITBATCHCND exit that do not get converted by the new RNLs. This could result if a resource was managed globally by an alternative serialization product before the migration, but afterwards is to be serialized by a device RESERVE.

Note: Any errors in changing the RNL configuration can lead to deadlocks or data integrity errors. When doing migration to the specified RNLs, the only way to move back to GRSRNL=EXCLUDE is a sysplex-wide IPL. The FORCE option cannot be specified from the existing RNLs to another set of RNLs. Long-held resources can delay such a change infinitely, and therefore a cancellation of jobs or even a sysplex-wide outage is required to end the job.

11.3 GRS IPCS enhancements

With z/OS V1R10, coding changes have been introduced to increase performance, as follows:

- Filtering options have also been introduced to reduce the amount of data processed and presented.

- A new GRS panel contains the following information:
  - Filter options
  - Summary/Detailed report types
- QCBTRACE, GRSTRACE, and ENQ attribute improvements, as follows:
- Event TODs show ENQ history for requests, contention, and ownership.
- Altered by RNLs, exits, and 3rd party managed.
- Directed ENQ details for the requesting TCB and target TCB.

  GRS has two major reporting interfaces. Note that GRSTRACE is the same as QCBTRACE. QCBTRACE provides ENQ information based on internal GRS control blocks that are contained in the GRS address space of the system that the dump was taken on. In STAR mode, QCBTRACE does not provide any information about other systems in the sysplex. However, it can provide more detailed ENQ information for ENQs that are obtained on the local system. It knows very little or nothing about A) the dumped system's ENQs that are related to waiters or blockers on other systems, and B) it knows nothing about holders on other systems. However, because QCBTRACE can use GRS internal control blocks, it can provide more information about the ENQs that originated on the dumping system.

GRSDATA provides complex-wide ENQ information that is gathered by a system-wide GQSCAN at dump or execution time (if live system data).

11.3.1 New IPCS GRS panel

You get to the GRSDATA panel via the following IPCS options:

- Select Option 2-Analysis on the primary IPCS panel.
- Then select Option 6-Component.
- Then select GRSDATA as shown in Figure 11-1 on page 254.

  The GRSDATA and GRSTRACE reports can be used to view resources and requesters known to the local system. The GRSDATA report uses SDATA=GRSQ records. The GRSTRACE report uses GRS internal control blocks from the GRS address space and includes diagnostic data and configuration information about GRS. Both reports support several filtering options to limit the amount of data returned. The GRSTRACE report also supports a DETAIL view.

  When GRS is in STAR mode, GRSTRACE can only show requests from the local system. The GRSDATA report can be used to see information that includes global resources from other systems. The amount of data included will depend on the GRSQ setting of the local system.

```
------------------ IPCS MVS DUMP COMPONENT DATA ANALYSIS --------------------
OPTION ===>                                                 SCROLL ===> CSR

To display information, specify "S option name" or enter S to the left
of the option desired.  Enter ? to the left of an option to display
help regarding the component support.

S Name     Abstract
S GRSDATA  ENQ/DEQ resources
  ICMPHDR  TCP/IP ICMP Header Formatter
```

Figure 11-1 IPCS panel to select the new GRSDATA panel

New GRSDATA panel

New keywords can also be used via keywords directly specified on the IP GRSDATA or IP VERBX GRSTRACE command.

Note that 5-contention from the main panel results in a GRS report that is similar to the GRSDATA Summary report when the contention filter is applied (ENQs in contention). However, it has no filter capability. GRSTRACE can provide a lot more detail for ENQs from the locally dumped system.

Figure 11-2 on page 255 is the new GRSDATA IPCS panel that is presented. Directly invoking the GRSDATA or VERBX GRSTRACE commands will not present this panel.

GRSDATA panel filter explanations

At least one requestor in a resource chain must match all of the filtering options in order for a resource to be returned.

SYSNAME Name of the system requesting the resource.

JOBNAME Name of the job requesting the resource.

ASID Address space number (in hex) requesting the resource.

TCB TCB address of the requesting task.

QNAME QNAME (Major Name) of the resource.

RNAME RNAME (Minor Name) of the resource.

SCOPE Select any combination (no selection means any scope).

CONTENTION Select this to only show resources in contention. This is filter is for ENQ contention only. Device contention will not be taken into consideration.

RESERVE Select this to only show resources with non-converted RESERVE requests.

START TIME Select a start time. Only resources with requests that occurred at or after this time will be shown.

STOP TIME Select a stop time. Only resources with requests that occurred at or before this time will be shown.

The time filters above are expected to be in the time format LOCAL, GMT, or UTC selected on the panel. LOCAL is the default. These filters are only applicable for the GRSTRACE report. The following fields accept wild cards:

```
SYSNAME, JOBNAME, QNAME, and RNAME
```

Wild card rules are to use:

* for zero or more characters and ? for exactly one character.

```
------------------------ IPCS - GRSDATA SUBCOMMAND --------------------
SELECT OPTION ===>
 Select a report type. The default is the GRSDATA report type.
   _ GRSDATA        _ GRSTRACE
 Select a level of detail. The default is SUMMARY reporting.
   _ SUMMARY        _ DETAIL (GRSTRACE only)
 Select the time format to use for the GRSTRACE report. The default is LOCAL.
   _ LOCAL          _ GMT          _ UTC
 Select zero or more filtering options. The default is NO filtering.
 Filters that do not apply to a given report will be ignored.
   SYSNAME  ________  JOBNAME  ________  ASID x' ____ '  TCB x' ________ '
   QNAME    ________
   RNAME    ________________________________________________________
   SCOPE:   _ STEP    _ SYSTEM    _ SYSTEMS
   _ CONTENTION       _ RESERVE
   START TIME MM/DD/YY,HH:MM:SS.DDDDDD  STOP TIME MM/DD/YY,HH:MM:SS.DDDDDD

 GRSDATA

  S = START  selected report.
  R = Reset all panel variables.
  END = Exit GRSDATA panel.
```

Figure 11-2 New IPCS GRSDATA subcommand panel

11.4 Query outstanding related ENQs

With z/OS V1R10, to allow applications to query outstanding related ENQs via something other than the ENQ's resource identity (QNAME/RNAME), the following macros ISGENQ and ISGQUERY now support a 32-byte USERDATA field.

The ISGENQ macro combines the serialization abilities of the ENQ, DEQ, and RESERVE macros. ISGENQ supports AMODE 31 and 64 in primary or AR ASC mode. ISGENQ enables you to specify that USERDATA can also be associated with the ENQ. ISGQUERY can retrieve the USERDATA, which allows the USERDATA to be a filter on the query.

The USERDATA keyword

When issuing the ISGENQ macro, USERDATA can be specified as follows:

```
USERDATA=userdata
USERDATA=NO_USERDATA
```

When TEST=NO and REQUEST=OBTAIN are specified, an optional input parameter that contains the userdata to be associated with this request can be specified. Note that GRS has no interest in the contents of USERDATA. Unlike the QNAME, RNAME, and SCOPE parameters, USERDATA has no meaning in the definition of the logically serialized resource identity. For example, exclusive requests with different user data and the same QNAME, RNAME, and SCOPE contend with each other. This request requires a version 2 parameter list. The default is NO_USERDATA.

The USERDATA can be up to 32 bytes of application data specified on ISGENQ REQUEST=OBTAIN that is associated with the ENQ resource. GRS has no interest in the data's contents. Applications can map the user data's contents in any way preferred and this data is not propagated throughout the GRS complex.

ISGQUERY macro

This macro supports the UERSDATA as an ISGQUERY specific or wild card filter. This support for the ISGENQ and ISGQUERY macros is primarily for vendors.

11.5 GRS ENQ and latch services

GRS provides two sets of critical system serialization services. The GRS ENQ services provide the ability to serialize an abstract resource within the scope of a JOB STEP, SYSTEM or multi-system complex (GRS complex). The GRS complex is usually equal to the sysplex. Via the HW reserve function, DASD volumes can be shared between different systems that are not in the same GRS complex or even in the same operating system. For example, between z/VM and z/OS. Enq/Reserve services can be used by authorized or unauthorized users. Almost every component, subsystem, and many applications use ENQ in some shape or form.

The GRS latch services provide a high-speed serialization service for authorized callers only. It uses user-provided storage to manage a lock table that is indexed by a user-defined lock number. GRS latch is also widely used. Very large users are USS, System Logger, RRS, MVS, and many others.

The GRS latch non-contention instruction path is on the order of a moderate number of instructions while ENQ is on the order of thousands for a local (single system) ENQ. GRS latch requires more recovery type coding on behalf of the user, for example, resource manager cleanup at task and address space termination.

New latch request information

With z/OS V1R10, a unit of work address and the "latch request hold elapsed time" is added to the **D GRS,CONTENTION,LATCH** command output. This solves a long term problem determination issue related to which units of work within an ASID are affected and if displayed, contention is for new or old instances of contention. With current releases, this is impossible to determine.

The difference is between long-term contention and new instances of short-term contention. For example, every time you look the same players are in contention but you do not know whether something is moving or not. Some latches can be in frequent contention.

The new information provided is as follows:

- The holding and or waiting units of work TCB or SRB within an ASID
- The amount of time that the latch is in contention

Not having this information makes it impossible to determine whether a latch was in contention continuously between intervals. For example, it has gone in and out of contention, but every time you look the same players are in contention.

Command examples

The GRS command to display latch contention is as follows:

```
D GRS,LATCH,CONTENTION
```

If latch contention exists, the system displays the messages shown in Figure 11-3.

```
ISG343I 23.00.04 GRS LATCH STATUS 886
LATCH SET NAME: MY.FIRST.LATCHSET
 CREATOR JOBNAME: APPINITJ CREATOR ASID: 0011
  LATCH NUMBER: 1
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    MYJOB1     001   EXCLUSIVE   OWN      006E6CF0  Y    00:00:40.003
    DATACHG    0019  EXCLUSIVE   WAIT     006E6B58  Y    00:00:28.001
    DBREC      0019  SHARED      WAIT     006E6CF0  Y    00:00:27.003
  LATCH NUMBER: 2
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    PEEKDAT1   0011  SHARED      OWN      007E6CF0  Y    00:00:32.040
    PEEKDAT2   0019  SHARED      OWN      007F6CF0  Y    00:00:32.040
    CHGDAT     0019  EXCLUSIVE   WAIT     007D6CF0  Y    00:00:07.020
  LATCH SET NAME: SYS1.FIRST.LATCHSET
 CREATOR JOBNAME: INITJOB2 CREATOR ASID: 0019
  LATCH NUMBER: 1
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    MYJOB2     0019  SHARED      OWN      006E6CF0  Y    00:01:59.030
  LATCH NUMBER: 2
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    TRANJOB1   0019  SHARED      OWN      006E7B58  Y    01:05:06.020
    TRANJOB2   0019  EXCLUSIVE   WAIT     006E9B58  Y    00:01:05.003
```

Figure 11-3 D GRS,LATCH,CONTENTION command output

Note: Refer to *z/OS MVS Diagnosis: Reference*, GA22-7588 and component-specific documentation for additional information on what specific latch contention could mean and what steps should be taken for different circumstances.

12

DFSMS enhancements

This chapter describes the enhancements to Data Facility Storage Management Subsystem (DFSMS) for z/OS V1R10.

The following topics are discussed:

- A brief introduction to DFSMS
- New parameters for the SMS command
- STOW IFF macro to prevent unintended replacement of a PDSE member
- SMS data class override
- Changed SMS message to SYSLOG
- Managing retention exceptions for DFSMSrmm during inventory management
- CDS fast replication for utilities EDGHSKP and EDGBKUP
- DELETE disposition support for tape data sets
- TSO command add-on

12.1 DFSMS introduction

DFSMS comprises a suite of related data and storage management products for the z/OS system. The following products are available as part of the DFSMS software suite:

DFSMSdfp Data Facility Product (DFSMSdfp) is a base element of z/OS. DFSMSdfp is automatically included with z/OS. DFSMSdfp performs the essential data, storage, and device management functions of the system. It helps us to store and catalog information on DASD, optical and tape resources.

DFSMSdss Data Set Services (DFSMSdss) is an optional feature of z/OS. It copies and moves data to help manage storage data and space more efficiently. DFSMSdss for example provides a utility to back up and restore data.

DFSMShsm Hierarchical Storage Manager (DFSMShsm) is an optional feature of z/OS. It is a disk storage management and productivity product for managing low activity and inactive data. It provides backup, recovery, migration and space management functions, as well as full-function disaster recovery support. It is mainly used by customers to improve disk use by automatically managing both space and data availability in a storage hierarchy.

DFSMSrmm Removable Media Manager (DFSMSrmm) is an optional feature of z/OS. DFSMSrmm can manage all of your tape volumes and the data sets on those volumes. It protects tape data sets from being accidentally overwritten, manages the movement of tape volumes between libraries and vaults over the life of the tape data sets, and handles expired and scratch tapes, all according to policies you define.

DFSMStvs Transactional VSAM Services (DFSMStvs) is an optional feature of z/OS. DFSMStvs enhances resource lock serialization of VSAM type data sets. It enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently. In addition to that it provides transactional recovery and data set recovery. Without DFSMStvs, batch jobs cannot update data sets while they are in use by CICS.

So basically DFSMS is an operating environment that helps automate and centralize the management of storage based on the policies that your installation defines for availability, performance, space, and security.

The heart of DFSMS is the Storage Management Subsystem. “SMS” also can stand for “system-managed storage”. Using SMS, the storage administrator defines policies that automate the management of storage and hardware devices. These policies describe data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements for the system. DFSMS is an exclusive element of the z/OS operating system and is a software suite that automatically manages data from creation to expiration.

12.2 z/OS V1R10 enhancements for DFSMSdfp

The enhancements for DFSMSdfp and DFSMSrmm are as follows:

- New parameters for the SMS command
- STOW IFF macro to prevent unintended replacement of a PDSE member
- SMS data class override

- Changed SMS message to SYSLOG
- Managing retention exceptions for DFSMSrmm during inventory management
- CDS fast replication for utilities EDGHSKP and EDGBKUP
- DELETE disposition support for tape data sets
- TSO command add-on

12.3 New DISPLAY SMS options for PDSEs

This section describes the enhancements made to DFSMSdfp, as follows:

- DFSMS provides an enhanced DISPLAY SMS command with new options to display point-in-time PDSE cache information in real time (as opposed to SMF records) and the overall effectiveness of PDSE caching.
- A new display for PDSE caching statistics at the data set level. The new command and its displays are designed to help you determine whether changes to cache settings might improve system performance.
- PDSE members can be cached in a hiperspace to provide enhanced performance for those PDSE data sets that are considered important in a critical performance path.

12.3.1 PDSE system commands

PDSE stands for partition data set extended. A PDSE is a partitioned data set that contains a directory of sequentially organized members, each of which can contain a program or data. This is similar to a directory in the Windows or UNIX operating systems. You can use a PDSE instead of a regular PDS. The main advantage of using a PDSE over a PDS is that a PDSE automatically reclaims the space released by a previous member deletion, without the need for a reorganization.

Prior to z/OS V1R8, the number of concurrently opened PDSE members that could exist on a system was limited by the PDSE address spaces' use of 31-bit addressable directory buffers. With z/OS V1R8, you can specify an amount of 64-bit virtual storage to be used to cache PDSE directory buffers in the PDSE address spaces. Specifying 64-bit virtual storage can help you increase the maximum number of concurrently open PDSE members and avoid possible directory space constraints.

A PDSE requires more attention and monitoring than a PDS (partitioned data set) partly because people have gotten used to the limitations of a PDS. z/OS V1R10 now allows more control to display PDSE caching information in real time.

A z/OS system has two PDSE address spaces that do caching of members and metadata:

SMSPDSE A non-restartable address space that is intended to handle global requests such as loading from LNKLST. These requests (connections) cannot tolerate interruption.

SMSPDSE1 Restartable through operator commands and intended to handle non-global processing. For SMSPDSE1 to be active, both PDSESHARING(EXTENDED) and PDSE_RETSTARTABLE_AS(YES) must be specified in the IGDSMSxx parmlib member.

The DISPLAY SMS,{PDSE|PDSE1} command now has two new options for additional cache information:

VSTOR Displays the PDSE caching information.

HSPSTATS Displays the current PDSE 64-bit directory buffer virtual storage utilization.

Command examples

The command changes are designed to help see the current utilization of the PDSE resources and provide enough information to optimize the PDSE usage. This helps in setting and modifying the parmlib for PDSE caching.

LRUCYCLES Specifies the maximum number of times (5 to 240 cycles) that the least recently used (LRU) routine passes over inactive buffers before making them available for reuse. While this parameter sets the maximum value, the LRU algorithm dynamically changes the actual number of times it passes over inactive buffers.

LRUTIME Specifies the number of seconds (5 to 60) that the system waits between calls to the LRU (least recently used) routine. The LRU routine releases inactive buffers in the dataspace that are used to cache PDSE (partitioned data set extended) directory data and in the hiperspace used to cache PDSE member data.

Note: PDSE caching is controlled using the LRUTIME and LRUCYCLES parameter fields in the IGDSMSxx parmlib member. For example:

```
PDSE1_LRUCYCLES(200)
PDSE1_LRUTIME(50)
```

To use the new parameter fields, the following syntax for the DISPLAY SMS command is required:

```
DISPLAY SMS,{PDSE|PDSE1} {,VSTOR                              }
                         {,HPSTATS[,DSN(dsname)][{,STORCLAS(sc)}
                                                 {,UNMANAGED   }]
                         [,SUMMARY             ]
                         [,MAXDSNS(maxds)      ]}
```

Figure 12-1 shows the output of a command with the new parameter field `HSPSTATS`.

```
d sms,pdse1,hspstats,unmanaged,dsn(sys1.*),maxdsns(3)
IGW048I PDSE HSPSTATS Start of Report(SMSPDSE1)
HiperSpace Size: 256 MB
LRUTime : 50 Seconds   LRUCycles: 200 Cycles
BMF Time interval 3600 Seconds
---------data set name----------------------Cache--Always-DoNot
                                          Elig---Cache--Cache
SYS1.SIEAMIGE                               Y      N      N
SYS1.SHASLNKE                               Y      N      N
SYS1.NFSLIBE                                Y      N      N
PDSE ANALYSIS  End of Report(SMSPDSE1)
```

Figure 12-1 Display current PDSE 64-bit directory storage utilization

Figure 12-2 on page 263 shows the new parameter field `VSTOR`.

```
d sms,pdse1,vstor
IGW050I PDSE Virtual Storage Start of Report(SMSPDSE1)
 Large Virtual Memory allocated for Address Space: 00004E00 Pages
   Directory Storage Size:   2000 MB
   Storage utilization in pages:   10
PDSE Virtual Storage  End of Report(SMSPDSE1)
```

Figure 12-2 Display PDSE caching information

Note: With z/OS V1R10, WLM will now manage the SMSPDSE1 address spaces in the SYSTEM service class even if they are defined differently in the WLM policy. This is intended to prevent system problems from occurring during periods of high system utilization.

12.4 STOW macro to write a PDS or PDSE member

The STOW macro is an application API that updates a partitioned data set directory or PDSE directory by adding, changing, replacing, or deleting an entry in the directory. You can also use the STOW macro to clear a partitioned data set or PDSE directory.

A practical example might be an application program that copies the data for a PDSE member from a z/OS host to a workstation and makes the member data available for a user to edit. When the user signals to the application program that the member edit is complete, the application attempts to put the modified member back into the PDSE on the host. When attempting to save the new version of the member into the PDSE, the application would like to ensure that the original PDSE member has not been otherwise altered during the time in which the user has had the member “checked out”. Therefore, the application program needs a method by which it can ensure that it does not STOW this new version of the member over any other changes to that member. To support this situation, new functionality was added in z/OS V1R10 to the STOW macro.

IFF parameter

The STOW macro now supports the IFF parameter to perform a “compare and swap” action:

```
STOW dcbaddr,plistaddr,IFF
```

The IFF directory action of the STOW macro can be used to conditionally add or replace a member of a PDSE if and only if an input timestamp matches the timestamp for the currently defined member of the same name. The IFF directory action also allows you to define metadata for the member being stowed in the form of a 16-byte type descriptor and two byte CCSID. This directory action is not supported for PDS members.

Application programs using the STOW macro

An application has to do the following consecutive actions to use this new functionality and prevent unintended replacement of incorrect versions of a PDSE member. This should be done, using STOW IFF, if and only if the user program can:

1. Save extended attributes with STOW
   - CCSID for member data (2 bytes)
   - TYPE DESCRIPTOR (16 bytes)

2. Extract extended attributes with the DESERV macro. The DESERV macro interface (functions GET and GET_ALL) can be used to retrieve the extended attributes, as shown in Figure 12-3.
   - CCSID and type descriptor
   - User ID of last change
   - Last change time stamp
3. Perform “compare and swap” STOW IFF.

```
*********************************************************************
** Extended Attributes Section
*********************************************************************
SMDE_EXT_ATTR      DSECT
SMDE_EXT_ATTR_LEN       DS H      length of this section
SMDE_CCSID              DS CL2    CCSID, or x'0000' if CCSID was not
*                                 defined
SMDE_TYPE_DESCRIPTOR    DS CL16   type descriptor, blank if not defined
SMDE_USERID_LAST_CHANGE DS CL8    userid of creater or last updater
*                                 of member
SMDE_CHANGE_TIMESTAMP   DS CL8    last member update or creation
*                                 timestamp, whichever is more recent.
```

Figure 12-3 Extended attribute section using the DESERV macro

Create a new PDSE data member using the STOW macro

The STOW macro may be used to create a new PDSE data member with new attributes such as TYPE_DESCRIPTOR and CCSID. It may also be used to update an existing member with the new TYPE_DESCRIPTOR and CCSID attributes. A valid matching timestamp of the last updater must be specified as an input to the STOW IFF for an update request. If the IFF function is specified, the DCB must be open for OUTPUT or UPDAT. This new function currently supports *only* PDSE data members, not program objects.

The IFF parameter indicates that a create or update function is to be performed on the attributes of an existing PDSE data member or new PDSE data member. To create a new PDSE data member with the new attributes, TYPE_ DESCRIPTOR and CCSID, a timestamp of ‘00000000’s needs to be specified on the compare operand. A timestamp of ‘00000000’s indicates that the member to be “replaced” must not previously exist. To update an existing PDSE data member’s attributes, a valid matching timestamp must be specified on the compare operand to match the timestamp of the last updater that is on the disk.

12.5 SMS data class override

In a z/OS environment each data set on disk is either SMS-managed or not. As described in the introduction on page 260, use the SMS storage administrator to define policies that automate the management of storage and hardware devices.

SMS has three constructs to setup the environment. Data class is one of three SMS constructs. The other two are storage class and management class. By definition, each SMS-managed data set has a storage class. If a data set does not have a storage class, it is not SMS-managed. The management class is optional and says something about the data set backup and migration criteria. Each data class is a set of data attributes (template) that

SMS uses to provide defaults for new data sets. An attribute could be the RECORD FORMAT or primary space quantity of a newly allocated data set.

Data set attributes

The installation defines data classes for SMS-managed and non-SMS-managed data sets. For each attribute, the system uses the first of these sources:

- Coded on the DD statement or dynamic allocation call
- Copied from data set referenced with DCB= or REFDD=
- Data class
- Supplied by an application program during OPEN
- Hard-coded

Using a data class is one way to create data sets with specific attributes. It is also possible to code it yourself in the DD statement of the JCL, or to copy it from another data set referenced with DCB= or REFDD=.

SMS calls an installation-provided routine called ACS (automatic class selection) to select the constructs. The ACS routines can ignore the constructs that the user requested.

12.5.1 Enforce data set size and block size

With z/OS V1R10, it is now possible to enforce an installation standard for data set size and the data set block size. Previously, users could override the data class installation standards for SPACE with any user-specified quantity.

Using data class override, the storage administrator can override user-specified space parameters with those specified in the associated data class with the following attributes:

```
AVGREC, AVG value, PRIMARY, SECONDARY and DIRECTORY.
```

The data class must have values for them. The storage administrator can interrogate and use user-specified SECONDARY SPACE and SPACE TYPE attributes in the ACS routines and force the system to determine the data set's block size. To use the new function, the storage administrator will need to add or modify data class definitions using ISMF or Naviquest. A subsequent change to the ACS routines may also be needed.

System-determined blocksize

If you request a system-determined block size, the system calculates the optimal block size, based on the logical record length (LRECL), record format (RECFM), and data set organization (DSORG) requested, and the geometry of the device selected for allocation. Each of these DCB parameters must be available from the program or JCL for the system to determine a block size. You omit the BLKSIZE or set BLKSIZE to zero to request the system to calculate a block size for the data set. Once the block size is calculated, the data set is indicated as reblockable to inform programs supporting system-determined block size that the data set's block size can be recalculated if the data set is moved.

With z/OS V1R10, if the data class "Forced System Determined Blocksize" attribute is specified, then the data set will use a system-determined block size even if BLKSIZE is specified by a user. You can cause existing data sets to use a system-determined block size by copying them using DFSMSdss and specifying the REBLOCK parameter or assign a data class with the "Forced System Determined Blocksize" attribute set to Y.

In order to use this new functionality, it is necessary to have a data class with space parameters as shown in Figure 12-4. Now to enforce the use of these space parameters, the following two data class definitions have to be set:

```
Override Space . . . . . : NO
System Determined Blocksize  : NO
```

```
  Panel  Utilities  Scroll  Help
------------------------------------------------------------------------------
                              DATA CLASS DEFINE                 Page 1 of 5
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DC1

To DEFINE Data Class, Specify:
  Description ==>
              ==>
  Recfm . . . . . .            (any valid RECFM combination or blank)
  Lrecl . . . . . .            (1 to 32761 or blank)
  Override Space  . . . . . N  (Y or N)
  Space Avgrec  . . . . . .    (U, K, M or blank)
        Avg Value . . . . .    (0 to 65535 or blank)
        Primary . . . . . .    (0 to 999999 or blank)
        Secondary . . . . .    (0 to 999999 or blank)
        Directory . . . . .    (0 to 999999 or blank)
  Retpd or Expdt  . . . . .    (0 to 9999, YYYY/MM/DD or blank)
  Volume Count  . . . . . . 1  (1 to 255 or blank)
  Add'l Volume Amount . . .    (P=Primary, S=Secondary or blank)
Use ENTER to Perform Verification; Use DOWN Command to View next Panel;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

Figure 12-4 Define data class panel with new option

ACS routine variables

There are two new ACS read-only variables that come with this functionality:

&SECOND_QTY Secondary space quantity

&SPACE_TYPE Space type (TRK, CYL, K, M, U, BLK, and black)

12.6 New SMS messages to the SYSLOG

There are also some messages that SMS always externalized (space constraint relief messages) that SMS now puts out into SYSLOG so that they are easily available to a system programmer or administrator who might want to take corrective action. They are:

```
IGD17286I, IGD17287I, IGD17288I, IGD17289I, IGD17291I, and IGD17292I
```

There is a minor change to the IGD17038I message. This message is issued when a DADSM installation exit rejects the volume SMS selected with a return code of 4 (which means a retry). This message would be issued once for each volume rejected and there can be hundreds, so SMS consolidated all these myriad messages into one summarized message.

In this release of z/OS, SMS will write more information when certain types of failures occur with SMS-managed data sets. More messages now go to the SYSLOG instead of only going

to the user. This makes it easier to do a problem analysis for a systems programmer or administrator who might want to take corrective action.

When an IEC614I DADSM message appears in the SYSLOG, it will now be followed by an SMS message IGD17053I.

```
IEC614I func FAILED - RC rc, DIAGNOSTIC INFORMATION IS (diaginfo) sss, ser,
dsname
```

IGD17053I message

You now also see an additional SMS message:

```
IGD17053I INVALID DADSM PARAMETER LIST OR VOLUME LIST FOR DATA SET dsname
HISTORIC RETURN CODE IS (rc) DIAGNOSTIC INFORMATION IS (diaginfo)
```

Note: The rightmost four hex digits of the IEC614I message field (diaginfo) are the equivalent of 17053.

IGD17038I message

For message IGD17038I the text changed slightly from previous releases of z/OS. This message is also sent to the SYSLOG. Prior to z/OSV1R10, the message was:

```
IGD17038I DADSM INSTALLATION EXIT REJECTED THIS REQUEST WITH A RETURN CODE OF 4
FOR DATA SET dsname
```

The new text with z/OS V1R10 is:

```
IGD17038I THE DADSM INSTALLATION EXIT REJECTED n VOLUMES WITH A RETURN CODE OF
4 FOR DATA SET dsname
```

12.7 DFSMSrmm enhancements for z/OS V1R10

z/OS V1R10 has made improvements to DFSMSrmm with the following new functions which are intended to make tape management easier and improve administrator productivity:

- Providing improved parmlib support for tape library and tape volume partitioning
- Improved reporting for DFSMShsm activity
- Support for new media end-of-life management policies based on media errors, volume usage, and age.

Planned improvements for DFSMSrmm also include:

- Interaction with IBM TotalStorage Productivity Center and a better interaction with IBM Integrated Removable Media Manager
- Enhanced reporting capabilities and new policies for tape scratch pool, retention, and expiration management

12.7.1 DFSMSrmm control data set

The DFSMSrmm control data set (CDS) is a VSAM key-sequenced data set that contains the complete inventory of the removable media library. DFSMSrmm records all changes made to the inventory, such as adding or deleting volumes, in the control data set. DFSMSrmm cannot track tapes that are accidentally moved to another system that has a different control data set.

Currently, the control data set information can be used as input to inventory reports and movement reports to keep track of volumes in your removable media library. You can also obtain control data set information by using RMM TSO subcommands and the DFSMSrmm ISPF dialog. Here is some of the information available:

- User ID, system, date, and time stamp of the last update before the audit record was created.
- Create date, time, user ID, and system.
- Storage location data and bin numbers.
- Original expiration date and the current expiration date. The original expiration date is the expiration date coded in the JCL when the data was originally written to the volume.
- Current location of a volume.
- Release actions for a volume.

With z/OS V1R10, more information is now provided about the inventory of the RMM tape pool through the new options and new message information available as described in this section.

12.7.2 Managing retention exceptions during inventory management

z/OS V1R10 provides additional support for inventory management of the RMM tape pool. Every tape data set can have a policy, and each policy can specify movement as well as retention.

DFSMSrmm provides several parmlib options to help you ensure that data is retained as expected and that the correct retention policies are in place. Use these options to customize how you expect DFSMSrmm to detect and handle exceptions to normal retention processing.

Vital record specification

The retention and movement policies you define to DFSMSrmm are known as vital record specifications (VRS). You use these new policies to:

- Indicate how long and where you want to keep data sets or volumes
- Define how volumes are to be moved among the libraries that DFSMSrmm supports, and the storage locations defined for vital records and disaster recovery purposes

For DFSMSrmm vital record specification processing is controlled by several SYS1.PARMLIB options. Prior to z/OS V1R10, only basic control mechanisms were available, using the VRSCHANGE, VRSMIN and NOTIFY parameters in the EDGRMMxx parmlib member.

New parmlib options

The new parmlib options are optional. Most of the volumes that DFSMSrmm manages in an installation are probably *scratch* volumes, that is, volumes that are used again and again by different users. Each time a volume is used, it is retained and managed by policies that are defined to DFSMSrmm. When the data is no longer required, the volume is returned to scratch status, and is ready for use by another user. There has always been a chance that environmental circumstances could force unintentional release or scratch action. Therefore, an automated way to recognize when too many volumes are released or expired was needed. With z/OS V1R10, new parmlib options are added which can be used to ensure that data is being retained as expected.

You can use the actions for each of these options to allow DFSMSrmm processing to continue after an information message is issued or you can have the message issued and

either a warning or an error return code set. You can build recovery actions into your production batch job to take the correct actions when one of these alerts occurs based on the job step return code.

All of these options, except for VRSCHANGE, provide you with a way to disable the checking that DFSMSrmm performs. When an option is disabled, DFSMSrmm does not count the resources and does not check whether the actions are to be taken. To disable an option set the action to OFF. Following are the new options.

VRSDROP To control the number or percentage of volumes dropped from VRS retention. Only volumes that are VRS retained are counted towards this limit. Newly assigned volumes are *not* counted here, even if they should become VRS retained during the current housekeeping run.

VRSRETAIN To control the number or percentage of newly assigned volumes that must be retained by VRS. Only newly assigned volumes are counted towards this limit. Newly assigned means that these volumes were not yet processed by inventory management.

EXPDTDROP To control the number or percentage of volumes dropped from EXPDT retention. Only volumes that are expiration date-retained and *not* VRS retained are counted towards this limit. Newly assigned volumes are *not* counted here, even if they should become expiration date-retained during the current housekeeping run.

These two options are now used and have small updates:

VRSMIN Use this to ensure that sufficient VRSs are defined to support your retention policies.

VRSCHANGE Use this to detect when a VRS might have been added, changed, or deleted, which could cause the number of data sets and volumes being retained to be different than expected.

Note: Each volume in the CDS is counted towards one of the limits only. A lot of decisions are made during the life cycle of a volume. For example whether or not it should be retained at some point based on its expiration date. All these decisions can be spoiled by whatever circumstances and volumes might get unintentionally released. The new feature for DFSMSrmm validates the results of these decisions and thus prevents accidents. it uses the new parmlib settings to control this behavior.

VRSDROP option

If this option is not specified, default values are used. The type of limit can be percentage or count. The default value is 10%. Possible actions are INFO, WARN, FAIL, and OFF. The default is INFO; see Figure 12-5 on page 270.

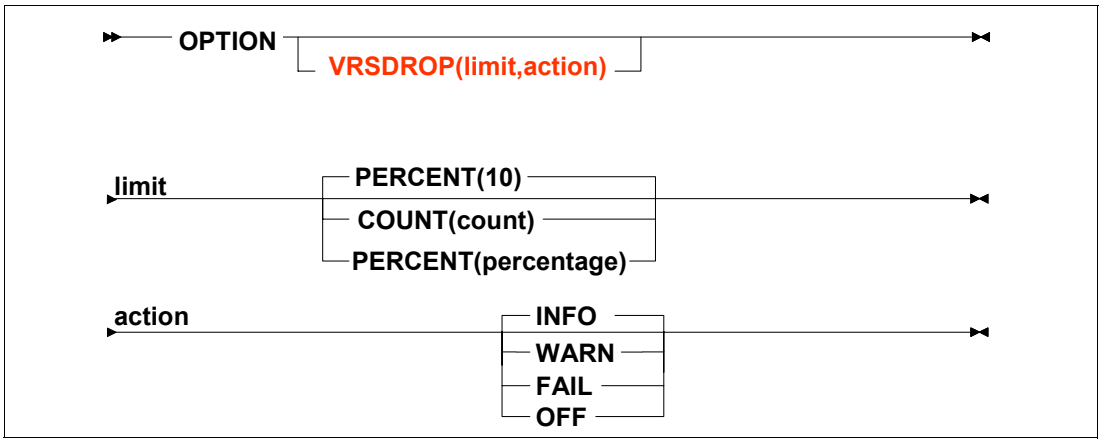


Figure 12-5 New VRSDROP option

INFO Will generate informational messages.

WARN Will issue the same messages plus, in case the limit is triggered, will set the EDGHSKP return code to 4.

FAIL Also issues the informational messages, but will in addition issue message EDG2310 and stop inventory management prior to making CDS updates in case the limit is triggered.

OFF Turns off the feature; no calculations will be done and no messages will be issued.

Note: These action options have the same meaning for VRSRETAIN and VRSDROP.

VRSRETAIN option

For VRSRETAIN only the default value (80%) is different from VRSDROP; see Figure 12-6.

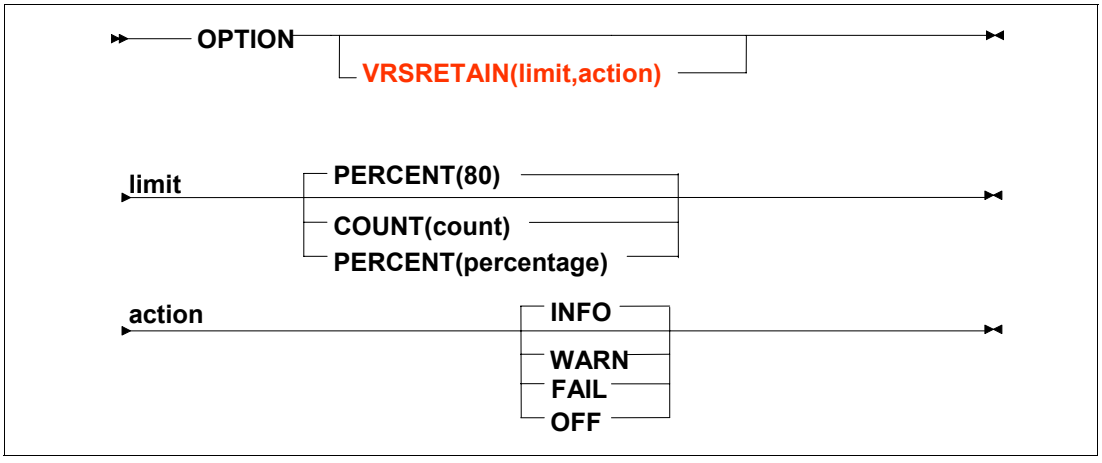


Figure 12-6 New VRSRETAIN option

VRSRETAIN processing is intended to provide limited checking for volumes containing newly created data sets that have not yet been processed by inventory management VRSEL. The data sets on the volumes have been created since the start of the last completed VRSEL run. It considers how many of these volumes become VRS-retained during the new VRSEL run. Those volumes which become VRS-retained will be considered by the VRSDROP limit checking future VRSEL runs. Those volumes that are not retained by VRS can be set pending release, depending on VRS release options and the volume expiration date. They

can also become EXPDT retained and on the next run of EXPROC, they will be considered by EXPDTDROP limit processing. The count can be 0 to 2,147,483,647. The percent can be 0 to 100.

EXPDTDROP

The EXPDTDROP option has a default limit value of 80%; see Figure 12-7.

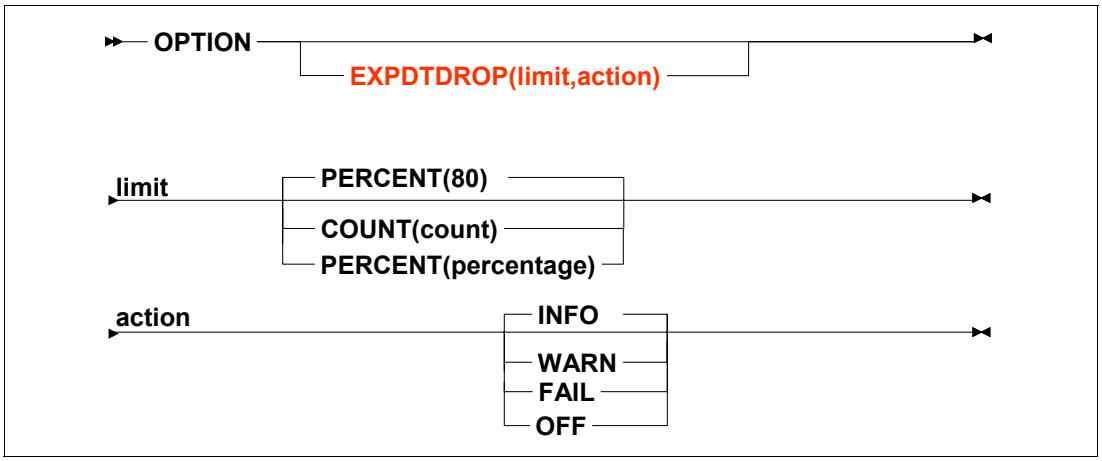


Figure 12-7 New EXPDTDROP option

Note: If you migrate to z/OS V1R10 and do not add the new options to your PARMLIB member, the default limits are used. The default action (INFO) is used.

12.7.3 DFSMSrmm ISPF panels

When entering the DFSMSrmm primary option panel, shown in Figure 12-8, select the menu options shown in Table 12-1 to display information.

```
EDG@PRIM        REMOVABLE MEDIA MANAGER (DFSMSrmm) - z/OS V1R10
Option ===>

0  OPTIONS       - Specify dialog options and defaults
1  USER          - General user facilities
2  LIBRARIAN     - Librarian functions
3  ADMINISTRATOR - Administrator functions
4  SUPPORT       - System support facilities
5  COMMANDS      - Full DFSMSrmm structured dialog
6  LOCAL         - Installation defined dialog
X  EXIT          - Exit DFSMSrmm Dialog
```

Figure 12-8 DFSMSrmm primary panel

Table 12-1 shows panel options for the changes made in z/OS V1R10.

Table 12-1 DFSMSrmm panel options

Panel ID	Panel option	Option description
EDG@PRIM	5	COMMANDS - Full DFSMSrmm structured dialog

Panel ID	Panel option	Option description
EDGP@CMD	7	CONTROL - Display system control information
EDGPC000	2	SYSTEM - Display system options and defaults

New panel information

It is possible to see the specified settings for EXPDTDROP, VRSDROP, and VRSRETAIN from the ISPF dialog panels.

Selecting Option 5 from Figure 12-8 on page 271, brings up panel EDGP@CMD, shown in Figure 12-9, where there is a display available for system control information.

```
EDGP@CMD               DFSMSrmm Command Menu - z/OS V1R10
Option ===>

0  OPTIONS   - Specify dialog options and defaults
1  VOLUME    - Volume commands
2  RACK      - Rack and bin commands
3  DATA SET  - Data set commands
4  OWNER     - Owner commands
5  PRODUCT   - Product commands
6  VRS       - Vital record specifications
7  CONTROL   - Display system control information
R  REPORT    - Report generator
```

Figure 12-9 DFSMSrmm Command Menu panel

From panel EDGP@CMD, select Option 7 to see panel EDGPC000, shown in Figure 12-10.

```
EDGPC000              DFSMSrmm Control Information Menu
Option ===>

 0  OPTIONS      - Specify dialog options and defaults
 1  CONTROL      - Display cds control record details
 2  SYSTEM       - Display system options and defaults
 3  SECURITY     - Display security classification rules
 4  VLPOOLS      - Display volume pool definitions
 5  MNTMSG       - Display mount message definitions
 6  REJECT       - Display volumes to be rejected
 7  ACTIONS      - Display volume moves and actions
 8  LOCDEF       - Display location definitions
 9  MEDINF       - Display media information definitions
10  PARTITION    - Display partition definitions
11  OPENRULE     - Display open rule definitions
```

Figure 12-10 DFSMSrmm Control Information Menu panel

Here, selecting Option 2 displays panel EDGPC200, shown in Figure 12-11 on page 273. It contains information about the VRS.

```
EDGPC200              DFSMSrmm System Options Display
Command ===>

                                                                      More:     +
Parmlib suffix . : 02

Operating mode . : PROTECT

Data sets:
  Control  . . . : RMM.CONTROL.DSET
  Journal  . . . : RMM.JOURNAL.DSET
  CDS id . . . . : SC70                       Journal threshold : 75 %
  Catalog SYSID  : NOTSET

Retention period:                           SMF:
  Default  . . . : 0                          System id . . . . : SC65
  Maximum  . . . : NOLIMIT                    Audit . . . . . . : 248
  Catalog  . . . : 6      hours               Security  . . . . : 249

PDA: ON                                     Report options:
  Block count  . : 255                        Lines per page  . : 54
  Block size . . : 27                         Date format . . . : JULIAN
  Log  . . . . . : ON

Miscellaneous:                              Auto-start procedures:
  RACF support . . . . . : NONE               Scratch procedure : EDGXPROC
  MAXHOLD limit  . . . . : 100                Backup procedure  : EDGCDSBK
```

Figure 12-11 DFSMSrmm System Options DIsplay panel

Figure 12-12 on page 274 shows the new options for managing the retention exceptions on the same panel, accessed by pressing the PF8 key on the currently displayed panel.

```
EDGPC200              DFSMSrmm System Options Display
Command ===>
Miscellaneous:                                Auto-start procedures:
  RACF support . . . . . : NONE                 Scratch procedure : EDGXPROC
  MAXHOLD limit  . . . . : 100                  Backup procedure  : EDGCDSBK
  IPL check  . . . . . : NO
  Uncatalog  . . . . . : YES                  VRS:
  System notify  . . . . : NO                   Job name  . . . . : 2
  BLP  . . . . . : RMM                          Minimum count . . : 1
  Master overwrite . . . . : USER               Minimum action  . : INFO
  Message case . . . . . : MIXED                Change  . . . . . : INFO
  Accounting . . . . . : JOB                    VRSEL processing  : NEW
  Disposition DD name  . . : DISPDD             Drop count  . . . : 10%
  Disposition message ID . : EDG4054I           Drop action . . . : INFO
  PREACS . . . . . : NO                         Retain count  . . : 80%
  SMSACS . . . . . : NO                         Retain action . . : INFO
  TVEXT purge  . . . . . : RELEASE              EXPDT drop count  : 10%
  Reuse bin  . . . . . : CONFIRMMOVE            EXPDT drop action : INFO
  COMMANDAUTH owner  . . . : NO                 Retain by . . . . : SET
  COMMANDAUTH dsn  . . . . : YES                Move by . . . . . : SET
  Default media name . . . : 3480
  Local tasks  . . . . . : 10
```

Figure 12-12 Panel EDGPC200 - DFSMSrmm System Option display

12.7.4 New messages with z/OS V1R10

The DFSMSrmm MESSAGE file used for inventory management contains all the information about the new limits. You can run inventory management BACKUP or RPTEXT functions as long as you do not run any other inventory management functions. To run VRSEL, DSTORE, or EXPROC, you must first successfully run the EDGHSKP utility vital record processing using the VERIFY parameter. DFSMSrmm issues return code zero for a successful run. Check the REPORT and ACTIVITY files from the VERIFY run and ensure that DFSMSrmm is performing the required retention and movement actions.

At the start of inventory management processing, DFSMSrmm checks the inventory management options that are in use and lists them in the MESSAGE file. The EDG2309I message shows the values that are used by DFSMSrmm, shown in Figure 12-13 on page 275.

```
EDG6001I INVENTORY MANAGEMENT STARTING ON 2008/117 AT 15:29:59 - PARAMETERS IN USE ARE
         DATEFORM(J),VRSEL,EXPROC,RPTEXT
EDG2309I THE PARMLIB OPTIONS CURRENTLY IN USE ARE
         VRSEL(NEW)
         VRSJOBNAME(2)
         VRSMIN(1,INFO)
         VRSCHANGE(INFO)
         VRSDROP(PERCENT(10),INFO)VRSRETAIN(PERCENT(80),INFO)EXPDTDROP(PERCENT(10),INFO)
         SMSTAPE(PURGE(ASIS) UPDATE(EXITS,SCRATCH,COMMAND))
         CATRETPD(6)
         UNCATALOG(Y)
         TPRACF(N)
         NOTIFY(N)
         SYSID(SC70)
         CATSYSID()
         RETAINBY(SET)
         MOVEBY(SET)
```

Figure 12-13 EDG2309I message showing information on the new options limit parameter

New messages related to VRSDROP, VRSRETAIN, and EXPDTDROP

With z/OS V1R10, the MESSAGE file also contains several new messages that show numbers calculated during inventory management and related to parmlib options VRSDROP, VRSRETAIN, and EXPDTDROP. These messages are grouped in pairs, one pair for each option, as shown in Figure 12-14.

```
EDG2242I INITIAL NUMBER OF VRS RETAINED VOLUMES         =         20  40%
EDG2244I NUMBER OF VRS RETAINED VOLUMES TO BE DROPPED   =          2  10%
EDG2243I INITIAL NUMBER OF NEWLY ASSIGNED VOLUMES       =         15  30%
EDG2245I NUMBER OF NEWLY ASSIGNED VOLUMES TO BE RETAINED=          5  33%
EDG2427I INITIAL NUMBER OF EXPDT RETAINED VOLUMES       =         10 100%
EDG2428I NUMBER OF EXPDT RETAINED VOLUMES TO BE DROPPED =          2  20%
```

Figure 12-14 New messages for VRSDROP, VRSRETAIN, and EXPDTDROP

New message EDG2435I

Beginning in z/OS V1R9, the EDGHSKP EXPROC utility can be run in parallel on multiple systems, but only one copy per system can be run regardless of the volumes that are processed. An optional EXPROC command in the SYSIN file can be used to:

- Request that only a subset of volumes are processed by EXPROC.
- Request that system-managed volumes are not returned to scratch during expiration processing. Instead, for each system-managed volume ready to return to scratch, a control statement is generated for use with the EDGSPLCS utility so that volumes can be scratched later. The control statements are written to the EDGSPLCS DD.

Message EDG2435I is new with release V1R10. Since z/OS V1R9 began using the EXPROC statement in the housekeeping JCL, a subset of volumes can be defined on which expiration processing is to be done. EDG2435I now displays the number of volumes selected with this statement. DFSMSrmm issues this message to the MESSAGE file during inventory management EXPROC processing. This message is issued for information only.

Percentage values

For some of the already established messages, with z/OS V1R10 additional percentage values are introduced. Message EDG2420I, shown in Figure 12-15, displays the absolute number of volumes read per type, what can be physical, logical, total, or stacked. The percentage value in messages EDG2420I and EDG2421I is based on the total number of volumes read from the CDS.

In the example in Figure 12-15, the calculations are as follows:

- 15 physical volumes are read from the CDS, 5 are updated, which makes 33%.
- Two stacked volumes are read from the CDS, one is updated, which makes 50%.

```
EDG2420I PHYSICAL VOLUMES READ                          =          15  75%
EDG2420I LOGICAL  VOLUMES READ                          =           3  15%
EDG2420I STACKED  VOLUMES READ                          =           2  10%
EDG2420I TOTAL    VOLUMES READ                          =          20 100%
EDG2421I PHYSICAL VOLUMES UPDATED                       =           5  33%
EDG2421I LOGICAL  VOLUMES UPDATED                       =           1  33%
EDG2421I STACKED  VOLUMES UPDATED                       =           1  50%
EDG2421I TOTAL    VOLUMES UPDATED                       =           7  35%
.....
EDG2422I TOTAL    VOLUMES, THIS RUN, KEPT FOR VRS       =           1   1%
.....
EDG2423I xxxxxxx  VOLUMES, THIS RUN, ASSIGNED TO STORES =
.....
EDG2435I TOTAL    VOLUMES SELECTED FOR EXPROC           =         178 100%
```

Figure 12-15 New message EDG2435I and percentage values

Summary messages

There are three more summary messages in the MESSAGE file that are new and are shown in Figure 12-16:

```
EDG2424I, EDG2425I and EDG2426I
```

The percentage in these messages is based on the number of volumes in the EXPROC subset. In Figure 12-16 are the following percentages:

- 75% of the volumes read were selected for EXPROC.
- 33% of the volumes selected for EXPROC were set pending release.
- 20% of the volumes selected for EXPROC were returned to scratch.

```
.....
EDG2420I TOTAL    VOLUMES READ                          =          20 100%
.....
EDG2435I TOTAL    VOLUMES SELECTED FOR EXPROC           =          15  75%
.....
EDG2424I TOTAL VOLUMES, THIS RUN, SET PENDING RELEASE   =           5  33%
EDG2425I TOTAL VOLUMES RETURNED TO SCRATCH              =           3  20%
EDG2426I TOTAL NUMBER OF SCRATCH RECORDS WRITTEN        =           0   0%
.....
```

Figure 12-16 New messages EDG2424I, EDG2425I, and EDG2426I

Special messages

Special messages in Figure 12-17 are issued to the MESSAGE file, depending on how the inventory management ended. Following are some messages for a case when the limit was triggered:

EDG2309I EXPDTDROP option is set, limit 10%, action is WARN

EDG2428I Processing identifies four volumes to be dropped, which is 20% of the initially expiration date-retained volumes

The limit is triggered, but the defined action is WARN only, which results in:

EDG2429I Updates completed

EDG2307I Task(s) completed

EDG2424I The four volumes are set pending release

EDG6901I EDGHSKP return code = 4

```
EDG6001I INVENTORY MANAGEMENT STARTING ON 2008/117 AT 15:29:59 - PARAMETERS IN USE ARE
         DATEFORM(J),VRSEL,EXPROC,RPTEXT
EDG2309I THE PARMLIB OPTIONS CURRENTLY IN USE ARE
.....
          VRSDROP ..............  EXPDTDROP(PERCENT(10),WARN)
.....
EDG2427I INITIAL NUMBER OF EXPDT RETAINED VOLUMES       =          20 100%
EDG2428I NUMBER OF EXPDT RETAINED VOLUMES TO BE DROPPED =           4  20%
.....
EDG2435I TOTAL    VOLUMES SELECTED FOR EXPROC           =          10  25%
EDG2424I TOTAL VOLUMES, THIS RUN, SET PENDING RELEASE   =           4  40%
.....
.....
EDG2429I MAIN INVENTORY MANAGEMENT UPDATES HAVE COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK VRSEL    COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK EXPROC   COMPLETED SUCCESSFULLY
EDG6901I UTILITY EDGHSKP COMPLETED WITH RETURN CODE 4
```

Figure 12-17 Special messages when inventory management ends with a return code 4

Figure 12-18 on page 278 shows a situation with a different inventory ending. The messages in the MESSAGE file for this case are for when the limit was triggered. The messages shown are:

EDG2309I EXPDTDROP option is set, limit 10%, action is FAIL

EDG2428I Processing identifies four volumes to be dropped, which is 20% of the initially expiration date-retained volumes

The limit is triggered and the defined action is FAIL, which results in:

EDG2310I Inventoring management is stopping

EDG2424I No volumes are set pending release

EDG2305E Inventory management failed

EDG6901I EDGHSKP return code = 12

```
.....
EDG2309I THE PARMLIB OPTIONS CURRENTLY IN USE ARE
.....
         VRSDROP ............... EXPDTDROP(PERCENT(10),FAIL)
.....
EDG2427I INITIAL NUMBER OF EXPDT RETAINED VOLUMES       =         20 100%
EDG2428I NUMBER OF EXPDT RETAINED VOLUMES TO BE DROPPED =          4  20%
EDG2310I INVENTORY MANAGEMENT STOPPING BECAUSE OF EXPDTDROP(PERCENT(10),FAIL)
OPTION
.....
EDG2435I TOTAL    VOLUMES SELECTED FOR EXPROC           =         10  25%
EDG2424I TOTAL VOLUMES, THIS RUN, SET PENDING RELEASE   =          0   0%
.....
.....
EDG2305E INVENTORY MANAGEMENT TASK EXPROC   FAILED WITH RETURN CODE 04
EDG6901I UTILITY EDGHSKP COMPLETED WITH RETURN CODE 12
```

Figure 12-18 New special messages with return code 12

12.7.5 TSO RMM subcommands

Another way to display the new parmlib options is with DFSMSrmm TSO subcommands. The options are returned in the output for a LISTCONTROL OPTION command. You can list the parmlib options currently in use with the RMM LISTCONTROL OPTION subcommand, or with the CONTROL ISPF dialog. Use the LISTCONTROL subcommand to display information in the control record of the control data set and information that is defined in the EDGRMMxx parmlib member. The appropriate part of TSO output changes with z/OS V1R10 is shown in Figure 12-19.

```
.....
VRS change      = INFO
VRSMIN action   = INFO     VRSMIN count   = 1
VRSDROP action  = INFO     VRSDROP count  = 0              percent = 10
VRSRETAIN action= INFO     VRSRETAIN count= 0              percent = 80
EXPDTDROP action= INFO     EXPDTDROP count= 0              percent = 10
.....
```

Figure 12-19 TSO subcommand RMM LISTCONTROL OPTION

Structured field introducer (SFI)

A structured field introducer is an 8-byte entity that either introduces the beginning of a group of data or introduces output data that immediately follows the introducer. Using the API you will get the new options returned as structured field introducers, as shown in Table 12-2.

Table 12-2 Structured field introducer names

SFI name	SFI number	SFI type	Length	Value	Commands
VDRP	x'8B280F'	4 - binary(15)	10	VRSDROP percent	LC OPT
VDRC	x'8B2802'	5 - binary(31)	12	VRSDROP count	LC OPT
VDRA	x'8B2800'	3 - binary(8)	9	VRSDROP action 0=FAIL, 1=INFO 2=WARN, 4=OFF	LC OPT

SFI name	SFI number	SFI type	Length	Value	Commands
VREP	x'8BD50F'	4 - binary(15)	10	VRSRETAIN percent	LC OPT
VREC	x'8BD502'	5 - binary(31)	12	VRSRETAIN count	LC OPT
VREA	x'8BD500'	3 - binary(8)	9	VRSRETAIN action 0=FAIL, 1=INFO 2=WARN, 4=OFF	LC OPT
XDRP	x'8C5D0F'	4 - binary(15)	10	EXPDTDROP percent	LC OPT
XDRC	x'8C5D02'	5 - binary(31)	12	EXPDTDROP count	LC OPT
XDRA	x'8C5D00'	3 - binary(8)	9	EXPDTDROP action 0=FAIL, 1=INFO 2=WARN, 4=OFF	LC OPT
VACT				+ 3=OFF	LC OPT

12.7.6 CDS fast replication

DFSMSrmm with z/OS V1R10 now supports the use of DSS copy services and exploitation of fast replication services provided by DASD subsystems. This provides you with a fast and non-intrusive way to create a backup of the DFSMSrmm control data set (CDS).

All data set, volume, and policy information is kept in the DFSMSrmm control data set (CDS). The CDS is a VSAM-key sequenced data set (KSDS) that contains all inventory information, and can be either an extended format (EF), or a non-extended format VSAM data set. This is an important data set. So the control data set backup should be a consistent copy of the CDS, that preferably is ready to use without further processing.

EDGHSKP utility

Now DFSMSrmm no longer relies on concurrent copy and virtual concurrent copy for fast and non-intrusive creation of copies or backups of the control data set. This change affects the EDGHSKP utility. The EDGHSKP utility can be used to run DFSMSrmm inventory management activities, which includes the following:

- **Vital record processing** - to determine which data sets to retain and which volume moves are required, based on VRSs
- **Expiration processing** - to identify volumes ready to be released and returned to scratch
- **Storage location management processing** - to set a destination for a volume. Optionally run storage location management to assign shelf locations in storage locations for volumes that are being sent out or returned to the removable media library
- **Backing up the CDS and the journal** - and clearing the journal
- **Creating an extract data set** - for report generation

New BACKUP option

In general BACKUP is used to back up or copy the control data set, to back up the journal, or to back up or copy both. BACKUP(AMS) is the default. DFSMSrmm uses the access method services REPRO command to perform backup processing. Specify BACKUP(AMS) to prevent DFSMSrmm from updating the control data set during control data set backup processing. Backup cannot be directly to tape.

Specifying BACKUP(AMS) prevents updates of the control data set during control data set backup processing. BACKUP(DSS) will create a portable backup of the CDS when trying to use the DFSMSdss concurrent copy environment. Updates of the DFSMSrmm control data set during backup processing are possible. If no concurrent copy environment is available, the backup will be done, but CDS updates are prevented.

With z/OS V1R10, there is a new third option for the BACKUP parameter of EDGHSKP, COPY. Specify BACKUP(COPY) to use DFSMSdss to create a copy of the CDS and optionally, a backup of the journal. The COPY uses DFSMSdss logical data set copy and enables you to use fast replication services that enable CDS updates to be made while the copy is created. COPY works just as the DSS option, but uses the COPY command instead of the DUMP command with DFSMSdss. The BACKUP DD is optional for COPY. Specify it if you want to direct the copy to a specific volume, or your environment is not SMS-managed. BACKUP(options) on the EXEC statement in the JCL are as follows:

```
BACKUP( AMS | DSS | COPY )
```

EDGHSKP JCL

Use the DSSOPT DD to override the default COPY command options used by DFSMSrmm. The BACKUP DD is optional when you request BACKUP(COPY). When you specify the BACKUP DD, DFSMSrmm builds the COPY command with the OUTDD(BACKUP) keyword to direct where the copy is created. Otherwise, your SMS-managed environment determines where the copy is created.

Figure 12-20 gives a sample of the JCL with the EDGHSKP utility and the BACKUP(COPY) parameter. The BACKUP DD is optional for COPY. Specify it if you want to direct the copy to a specific volume or your environment is not SMS-managed.

```
//EDGHSKP EXEC PGM=EDGHSKP,PARM='BACKUP(COPY)'
//MESSAGE  DD DISP=SHR,DSN=RMM.MESSAGE
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,UNIT=SYSDA,VOL=SER=myvol
//DSSOPT DD *
 FR(PREF) CONCURRENT(ANYREQUIRED) -
 RENAMEU((*.CDS.**,*.COPYCDS.**)) REPLACEU -
 DEBUG(FRMSG(DETAILED))
/*
```

Figure 12-20 EDGHSKP utility JCL

Note: For journal backup, no matter what option is used, DFSMSrmm uses IDCAMS to back up the journal.

The EDGBKUP utility

Similar to the use with the EDGHSKP utility, the parameter BACKUP(COPY) can be used with EDGBKUP also. The EDGBKUP utility can be used to reorganize and restore your DFSMSrmm CDS. Also, you can use this utility to back up both your DFSMSrmm CDS and journal and to clear the journal. The BACKUP options are as follows with the new option COPY:

```
BACKUP( NREORG | DSS | REORG | COPY )
```

DSSOPT DD statement

When using the backup (EGDHSKP) or restore (EDGBKUP) utility of DFSMSrmm, under the covers it will issue DUMP, COPY, and RESTORE commands to DFSMSdss. In the sample JCL shown in Figure 12-12 on page 274, the contents of the DSSOPT DD statement are used to replace the SECOND LINE of the default command, as shown here (Figure 12-21) for a COPY command. If you decide to change the command, specify *all* the operands you want to have processed, because DFSMSrmm uses your input in place of its own. The default is:

```
COPY DS(INCLUDE(cds_name)) [OUTDD(BACKUP)] SHARE -
   FR(PREF) CONCURRENT RENAMEU((**.CDS,**.COPYCDS)) REPLACEU
```

This sample JCL is used to create a copy of the control data set using fast replication. The copy of the control data set is created using a new data set name based on the current CDS dsname. The example ensures that fast replication is used so that an almost instant copy of the CDS can be created. No journal backup is created in this example, which is run with DFSMSrmm active so that the MASTER DD is not required. The CDS data set name is obtained from the running DFSMSrmm subsystem. The data set name of the copy is created from the existing CDS data set name. The second qualifier of CDS is renamed to COPYCDS. The copy is almost instant, and this is a non-intrusive copy of the CDS. Updates to the CDS are allowed during backup.

```
//EDGBKUP EXEC PGM=EDGBKUP,PARM='BACKUP(COPY)'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,UNIT=SYSDA,VOL=SER=myvol
//DSSOPT   DD *
  FR(PREF) CC(ANYREQ) RENAMEU((*.CDS**,*.COPYCDS.**)) REPLACEU
```

Figure 12-21 Sample JCL with the DSSOPT DD statement

The command operands that you can specify in the DSSOPT DD statement are controlled and validated by DFSMSdss, *not by* DFSMSrmm. If you specify an unsupported command operand, DFSMSdss fails the operation.

The COPY command used by DFSMSrmm includes the OUTDD(BACKUP) keyword only if you have specified the BACKUP DD. When the OUTDD keyword is not specified, processing relies on SMS ACS processing being used to determine the target volume.

Default keywords

The default keywords used by DFSMSrmm are:

FR CONCURRENT The COPY command includes FR(PREF) combined with the CONCURRENT keyword. This enables the use of FlashCopy, concurrent or virtual concurrent copy if possible. If this is possible, a non-intrusive backup is performed to create a CDS copy, otherwise the backup continues but CDS updates are prevented until the CDS copy completes. **Note**: Do not specify FR(PREF) without the CONCURRENT keyword, or the FR(REQ) keyword because this causes processing to be intrusive.

RENAMEU The RENAMEU keyword on the COPY command specifies a filter mask (**.CDS) and renaming rule (**.COPYCDS). This changes the ending qualifier of the copied data set from .CDS to .COPYCDS. If your control data set name does not end in .CDS or you want to use a different renaming rule, you must use DSSOPT to override the default.

REPLACEU REPLACEUnconditional specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If a usable preallocated target data set is found, it is replaced.

Note: Be aware of the prerequisites needed to do non-intrusive backup of the CDS. You must have the hardware and software required to establish a concurrent copy session, a virtual concurrent copy session, or fast replication with concurrent copy or virtual concurrent copy.

If you want to avoid an intrusive backup process, specify parameters FR(PREF) CC(VIRTUALREQ) or FR(PREF) CC(ANYREQ). This will cause the COPY to fail (instead of being done intrusive) in cases where the hardware or software prerequisites are not met.

Intrusive backup means that no updates to the control data set are allowed until DFSMSdss notifies DFSMSrmm that the control data set copy, or copy initialization is completed. In the special case if parameter FR(REQ) is specified there is no notification to DFSMSrmm until all DFSMSdss processing is completed. CDS updates during this time will be handled as if an intrusive backup is in progress!!

12.7.7 DELETE disposition support for tape data sets

With z/OS V1R10, DFSMSrmm now considers the normal disposition for a tape data set. If the normal disposition as coded in the JCL or dynamic allocation is DELETE, the data set record is updated to track the data set as “deleted” by disposition processing. This is a flag just like the abend flag. Prior to z/OS V1R10, DFSMSrmm did not consider the disposition of a tape data set. Also, VRSEL processing is now updated to process the new restricted DELETED VRSs. The major benefit is that tape data sets deleted via normal disposition processing can be fast tracked back to the scratch pool.

DFSMSrmm now provides support for managing deleted tape data sets. DFSMSrmm checks the normal disposition at CLOSE time and if this is DELETE, a deleted flag is recorded in the data set record. Subsequent use of a data set cannot change this. However, you can use the CHANGEDATASET command to reset the flag. The DELETE disposition does not have to be specified on the job step where the tape data set is created. It could be specified in a different step or even a different job. Tape data sets are subject to VRS processing as specified by your retention and movement policies. To ensure that deleted data sets are managed differently, you must create special DELETED VRSs. Otherwise, they are managed by the normal matching VRS.

The management of deleted tape data sets is for VRSEL(NEW) only. A VRS can use the DELETED value. DELETED is a restricted VRS data set name and a restricted VRS JOBNAME.

```
//WRITE2   EXEC PGM=IEBGENER
//SYSUT2   DD DISP=(NEW,DELETE,DELETE),
//         LABEL=(1,SL),VOL=SER=A03588,
//         DSN=USERID.TEST,UNIT=3480,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
```

Figure 12-22 Sample JCL with a DELETE disposition for a tape data set

Data set record

You can get detailed information about a data set using the RMM TSO LIST subcommand (Figure 12-23). The output listed is enhanced with an additional field about the “deleted” status and “close by abend” status. Following is an example of the TSO command:

```
RMM LISTDATASET D046059.TEST VOLUME(A03588)
```

```
Data set name = D046059.TEST
Volume        = A03588                Physical file sequence number = 1
Owner         = RMMUSER                           Data set sequence = 1
Create date   = 2008/010   Create time = 05:31:10 System ID         = W98MVS1
Expiration date       = 2008/015   Original expir. date  =
Block size            = 800        Block count           = 1
Percent of volume     = 0          Total block count     = 1
Logical Record Length = 80         Record Format         = FB
Date last written     = 2008/010   Date last read        = 2008/010
Job name              = D046059J   Last job name         = D046059J
Step name             = WRITE2     Last step name        = WRITE2
Program name          = IEBGENER   Last program name     = IEBGENER
DD name               = SYSUT2     Last DD name          = SYSUT2
Device number         = 0910       Last Device number    = 0910
Management class      =            VRS management value  =
Storage group         =            VRS retention date    =
Storage class         =            VRS retained          = NO
Data class            =            Closed by Abend       = NO
                                   Deleted               = YES
                                   Catalog status        = UNKNOWN
Primary VRS details:
...
```

Figure 12-23 RMM LISTDATASET command from TSO command line

Note: This field was also added as a new 'deleted' flag to the EDGSDREC SMF data set record.

Data set name VRS “DELETED”

The management of deleted tape data sets is an extension of that provided for OPEN and ABEND. It is for VRSEL(NEW) only. For DELETED VRSs, the matching sequence is identical to that for OPEN and ABEND. Deleted data sets which do not match to a DELETED VRS are matched to other VRSs as normal.

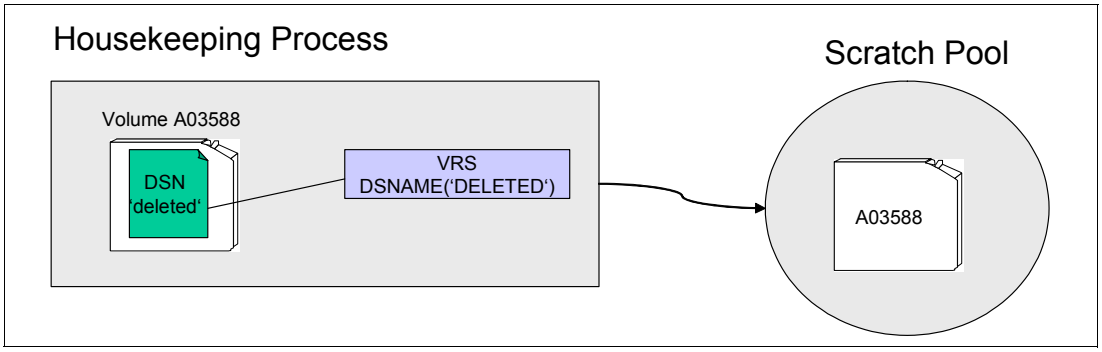


Figure 12-24 Sample housekeeping process using a DELETED VRS

Attention: It is recommended that you use VRSEL(NEW), because VRSEL(OLD) will be removed in a future release.

To ensure that deleted data sets are managed differently, you must create special 'DELETED' VRSs, otherwise they are managed by the normal matching VRS. Figure 12-25 shows the add VRS (AS or ADDVRS) command. If a data set is marked "open" and "closed by abend" and "deleted by disposition", the matching order is OPEN, then DELETED, then ABEND.

```
RMM AS DSNAME('DELETED') DAYS COUNT(0) +
RELEASE(EXPIRYDATEIGNORE,SCRATCHIMMEDIATE)
RMM AS DSNAME(mask) JOBNAME(DELETED) DAYS COUNT(0) +
RELEASE(EXPIRYDATEIGNORE,SCRATCHIMMEDIATE)
```

Figure 12-25 RMM ADDVRS command

Note: If JOBNAME(DELETED) in any data set name VRSs exists, this must be changed to JOBNAME(DELETED*) to avoid a conflict with the new 'DELETED' VRS support. This also goes for DSNAME. If DSNAME('DELETED') in any data set name VRSs exists, this must be changed to DSNAME('DELETED*') to avoid a conflict with the new 'DELETED' VRS support.

12.7.8 Using RMM TSO subcommands

Using the RMM TSO command and subcommands is an alternative to using the RMMISPF dialog and provides some additional attribute setting capability than provided by the dialog. You can issue the RMM TSO command and subcommands from within the DFSMSrmm ISPF dialog or outside the dialog, in TSO in the foreground, or in the background by submitting a batch TMP job, or from a TSO CLIST or REXX EXEC, or from System REXX, or from an operator console.

Operator console

You can now issue the RMM TSO command from your operator console and have the command output returned to your operator console and the system log. For example, use this command format to issue the TSO command from your operator console:

```
F DFRMM,CMD=LV volser
```

This subcommand lists the summary volume information just as if the command had been entered from a TSO session.

Using TSO subcommands from a TSO CLIST or REXX exec

The RMM TSO command is a regular, but authorized TSO command and should function in a TSO environment as any other TSO command. For use from a REXX environment you must address the TSO environment.

Note: For examples of the use of RMM subcommands from REXX see the EDGXMP1 and EDGXMP2 samples in SAMPLIB.

Using RMM TSO subcommands with System REXX

RMM TSO subcommands can be issued via System REXX as long as the target System REXX environment is TSO=YES (no testing has been done with TSO=NO). You should put the REXX exec containing the RMM TSO subcommands in the library supported by System REXX, currently SYS1.SAXREXEC. You can execute any System REXX exec using the following operator command from a console:

```
F AXR,rexx_name
```

Just as with regular REXX, if your System REXX exec sets the REXX variable SYSAUTH.EDGDATE, the RMM subcommand processing creates REXX variables as expected. If SYSAUTH.EDGDATE is not set, RMM subcommand processing results in no REXX variables being created and the resulting line command output is issued to the console, as shown in Figure 12-26.

```
/* REXX */
sysauth.edgdate = 'AMERICAN'
"RMM SV OWNER(*)"
say 'command = RMM SV OWNER(*)'
say 'num vols = ' edg@vol.0
say 'first vol found = ' edg@vol.1
The output on the console:
command = RMM SV OWNER(*)
num vols = 10
first vol found = V10001
```

Figure 12-26 System REXX statements and output to the console

DFSMSrmm REXX variables

The new options are returned as REXX variables as well and are shown in Table 12-3.

Table 12-3 DFSMSrmm REXX variable names

REXX Variable Name	Format	Content	Commands
VDRP	Numeric 0 – 100	VRSDROP percent	LC OPT
VDRC	Numeric 0 - 2147483647	VRSDROP count	LC OPT
VDRA	4 characters: FAIL, INFO, WARN or OFF	VRSDROP action	LC OPT
VREP	Numeric 0 – 100	VRSRETAIN percent	LC OPT
VREC	Numeric 0 - 2147483647	VRSRETAIN count	LC OPT
VREA	4 characters: FAIL, INFO, WARN or OFF	VRSRETAIN action	LC OPT
XDRP	Numeric 0 – 100	EXPDTDROP percent	LC OPT
XDRC	Numeric 0 - 2147483647	EXPDTDROP count	LC OPT
XDRA	4 characters: FAIL, INFO, WARN or OFF	EXPDTDROP action	LC OPT
VACT	+ OFF	VRSMIN action	LC OPT

TSO command add-on

There are new DELETED operands for the ADDDATASET, CHANGEDATASET, and SEARCHDATASET commands. These DFSMSrmm TSO commands can be used to manipulate the data set information stored by DFSMSrmm.

A new operand DELETED, on the ADDDATASET (AD) or CHANGEDATASET (CD) commands, either sets or resets the deleted flag in the data set record, as shown in Figure 12-27.

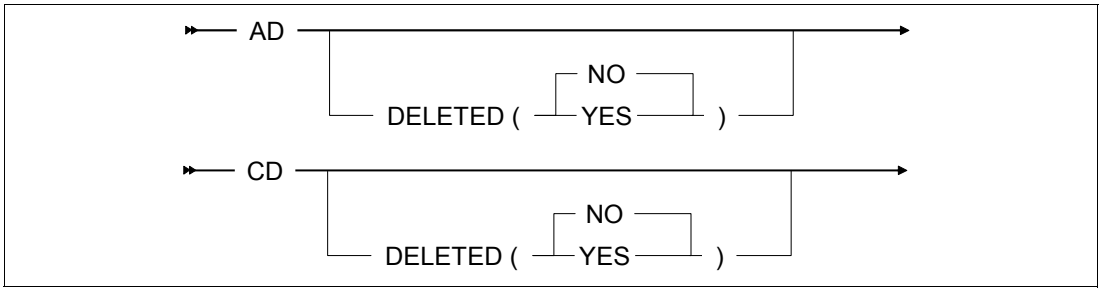


Figure 12-27 DFSMSrmm TSO command ADDDATASET and CHANGEDATASET

The new operand DELETED on the SD command limits your search result list to only those data sets which are marked deleted in case of value YES, as shown Figure 12-28.

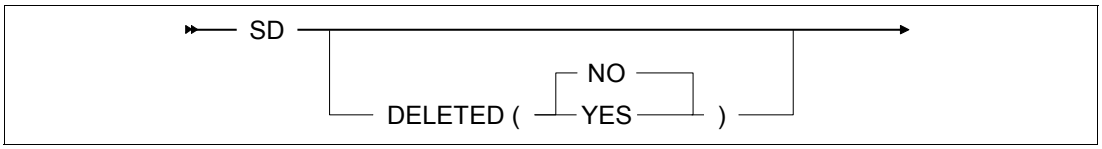


Figure 12-28 DFSMSrmm TSO command SEARCHDATASET

Note: The new field DELETED is also implemented in some of the DFSMSrmm ISPF panels.

12.7.9 Volume replacement policies

In order to manage the life-cycle of active and inactive volumes, z/OS V1R10 now has new functionality for detecting these volumes. There is a new PARMLIB option to allow the tape error threshold to be specified.

Before z/OS V1R10, it was more difficult to customize volume replacement, or to extend the criteria and to identify end-of-life tapes. It required a USERMOD to set the limit to another value than “1” for permanent tape I/O errors and to trigger a REPLACE release action. During the housekeeping process, volumes were checked against the set limit. If the limit was met or exceeded, the REPLACE release action was set.

It is possible to replace volumes when the volumes have permanent I/O errors or when the volumes reach the end of their useful life. DFSMSrmm tracks the number of I/O errors recorded for a volume and, by default, identifies it as needing to be replaced as long as you have not manually released the volume and are running DFSMSrmm expiration processing. You can now tailor this process by setting up your own volume replacement policies in PARMLIB. You can use your volume replacement policies to specify the limits for I/O errors, for Write Mount Counts, Requesting and Releasing Volumes, and for Age of the volumes.

During DFSMSrmm expiration processing, the volumes are checked against these policies and when one or more of the limits is reached or exceeded, the volumes' release action is set to REPLACE. DFSMSrmm also automatically sets the release action of a volume to REPLACE when the tape drive issues an alert that the volume should be replaced. Tape librarians must manually replace these volumes and confirm to DFSMSrmm that the action has taken place.

There are two situations when this new functionality is invoked. Both involve active and inactive volumes:

- During usage of volumes in IBM tape drives

  IBM has a function called Statistical Analysis & Recording System (SARS) that is used by current IBM tape drives for tracking media statistics to determine or predict media failure. RMM now uses the alerts from SARS Media Information Messages (MIM) to drive the setting of the REPLACE release action.

- During the Housekeeping EXPROC process

  All physical, including stacked and worm, volumes are checked if the following exceed the limits defined in the volume replacement policies:

  – Recorded temporary or permanent read or write errors
  – Age
  – Write mount count

The MEDINF command for the EDGRMMxx parmlib member

The volume replacement policies are built on the media information support of the MEDINF command in the parmlib. This allows you to customize the external used values for media types, recording format, and capacity for non-IBM media. The media information support MEDINF was introduced in 2007 with APAR OA13370.

An additional EDGRMMxx parmlib member definition for MEDINF allows you to customize the external used values for media type, recording format, and capacity for a non-IBM media. The new operand MEDINF is available to assign media information to a volume either at creation time via the RMM TSO command ADDVOLUME or when modifying the tape inventory recorded information via the RMM TSO subcommand CHANGEVOLUME. If no media information is assigned, DFSMSrmm continues to use the internally built conversion table allowing you to use MEDIATYPE values such as CST and RECORDINGFORMAT values like 128TRACK . If you assign media information to a volume for which there is no installation-defined media information available, the request is rejected. When there is no matching installation-defined media information for a recorded media type or recording format for a volume to convert internal values(n, m) to external, the default values MEDxxx and RECxxx will be used. Figure 12-29 shows the MEDINF syntax.

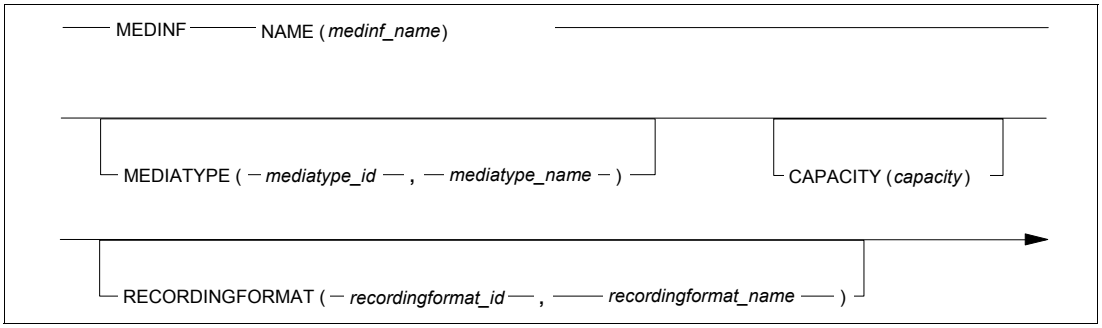


Figure 12-29 MEDINF command

NAME Specifies a name for the media information. This value is one to eight alphanumeric characters. It is used to match the recorded volume information to the MEDINF installation-defined media information. The values defined for MEDIATYPE, RECORDINGFORMAT, and CAPACITY are taken if the assigned media information for a volume matches medinf_name.

MEDIATYPE The MEDIATYPE tuple is used to convert the internal used media type mediatype_id to the external used mediatype_name in reports and in output of RMM TSO subcommands, as well as to convert the external used value mediatype_name to the internal value mediatype_id recorded in the control data set when used as input to RMM TSO subcommands. When you specify multiple MEDINF entries that use the same mediatype_id but with different mediatype_name values, the first such MEDINF entry is used for internal to external conversion. The remaining values are synonyms used for external to internal conversion.

RECORDINGFORMAT The RECORDINGFORMAT tuple is used to convert the internal used recording format recordingformat_id to the external recordingformat_name used in reports and in RMM TSO subcommands output, as well as to convert the external used value recordingformat_name to the internal value recordingformat_id recorded in the control data set when used as input to RMM TSO subcommands. When you specify multiple MEDINF entries that use the same recordingformat_id but with different recordingformat_name values, the first such MEDINF entry is used for internal to external conversion. The remaining values are synonyms used for external to internal conversion.

CAPACITY Specifies the media capacity in MB available on the non-IBM media that has a mediatype mediatype_id and a recording format recordingformat_id. The default capacity is 0 MB.

With z/OS V1R10, the new REPLACE operand (Figure 12-30) can be used to identify the policies for replacement of volumes based on such values as read and write errors, age, and number of times written. Volumes are identified for replacement when one or more of the policy values are met or exceeded.

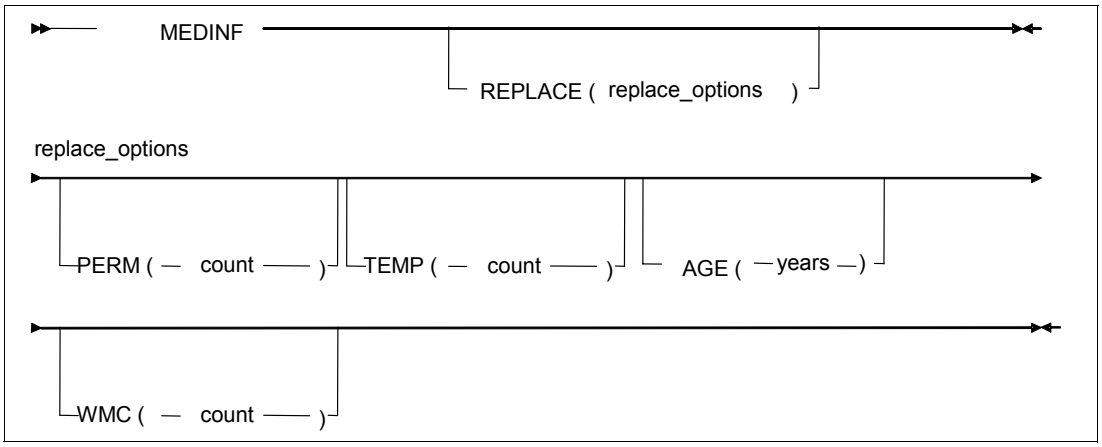


Figure 12-30 REPLACE operand on MEDINF

PERM(count) Use this operand when your policy is to be based on permanent I/O errors. DFSMSrmm compares your value with the sum of permanent

read and write errors over the life of the volume. The value range is 0 to 99999. Specify 0 to disable replacement based on permanent errors. There is no default value.

TEMP(count) Use this operand when your policy is to be based on temporary I/O errors. DFSMSrmm compares your value with the sum of temporary read and write errors over the life of the volume. The value range is 0 to 99999. Specify 0 to disable replacement based on temporary errors. There is no default value.

AGE(years) Use this operand when your policy is to be based on the age of the volume. DFSMSrmm compares your value with the volume creation, and current, date and time. The value range is 0 to 99999 years. Specify 0 to disable replacement based on age. There is no default value.

WMC(count) Use this operand when your policy is to be based on how many times the volume is mounted for output and written to. DFSMSrmm compares your value with the write mount count for the volume. The value range is 0 to 99999. Specify 0 to disable replacement based on volume write mount count. There is no default value.

Note: Until you define one or more MEDINF commands in parmlib with the REPLACE operand DFSMSrmm, processing is unchanged. A hard-coded value of PERM(1) is used.

Do not specify the REPLACE operand on a MEDINF command that specifies the same MEDINF NAME and MEDIATYPE mediatype_id and the RECORDINGFORMAT recordingformat_id as an earlier MEDINF command. If you do, an information message EDG0243I is issued at start-up time, the REPLACE operand is ignored, and the replace policy from the earlier entry is used instead. When you use synonyms, only use the REPLACE operand if the combination of MEDINF NAME, MEDIATYPE mediatype_id and RECORDINGFORMAT recordingformat_id are unique.

When you add or modify a volume replacement policy, DFSMSrmm does not implement it retrospectively. Volumes that are already set with the REPLACE release action are not reprocessed and the action is not reset. In order to have DFSMSrmm reconsider the policy for all non-pending release volumes, you first have to manually change the REPLACE release action to SCRATCH, and then run EXPROC processing. When you do not specify replacement policies with MEDINF, DFSMSrmm uses the built-in value of REPLACE(PERM(1)).

Volume replacement policy examples

Figure 12-31 gives an example of a specific volume replacement policy. This replacement policy is valid for all volumes with the specified MEDINF Name, Internal Mediatype and Internal Recording format.

```
MEDINF NAME(STK) MEDIATYPE(1,STANDARD) RECORDINGFORMAT(3,36BTRACK) +
CAPACITY(400) REPLACE(PERM(1) TEMP(4) WMC(99999) AGE(15))
```

Figure 12-31 Specific volume replacement policy

Figure 12-32 on page 290 gives an example of a Mediatype level volume replacement policy. This replacement policy is valid for all volumes with the MEDINF name “IBM”, Internal MEDIATYPE “3” and any RECORDINGFORMAT, if no specific volume replacement policy exists.

```
MEDINF NAME(IBM) MEDIATYPE(3,HPCT) REPLACE(PERM(1) TEMP(5) AGE(15) WMC(99999))
```

Figure 12-32 Mediatype level volume replacement policy

Figure 12-33 gives an example of a global MEDINF Name volume replacement policy.

```
MEDINF NAME(STK) REPLACE(PERM(2) TEMP(8) AGE(20))
```

Figure 12-33 Global MEDINF Name volume replacement policy

Displaying the volume media information

With the RMM LISTCONTROL command, shown in Figure 12-34, you can display the volume replacement policies.

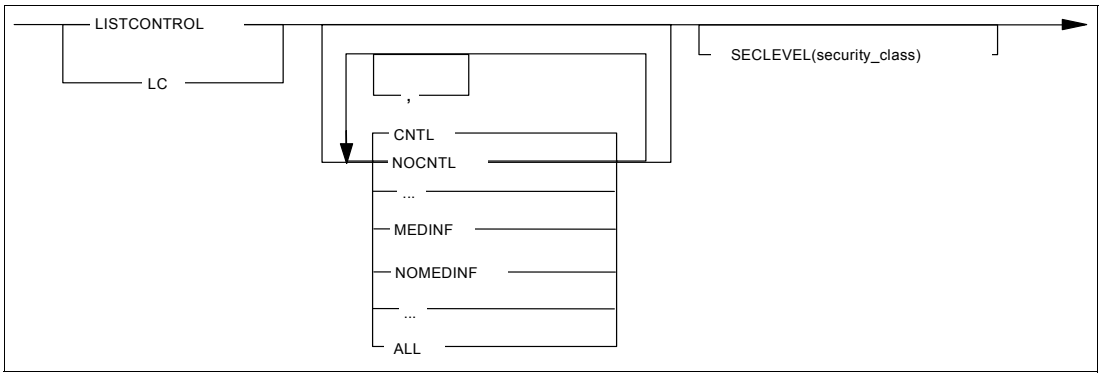


Figure 12-34 LISTCONTROL display command

The RMM LC MEDINF command provides the output shown in Figure 12-35.

```
Media Information :
Name     Media type   Recording format Capacity (MB)
-------- ------------ ---------------- -------------
MEDINFA    1 MEDIA1       1 RECTECHA         10000
MEDINFB    2 MEDIA2       1 RECTECHB        150000
MEDINFC    6 MEDIA6       6 RECTECHC     999999999
```

Figure 12-35 LISTCONTROL MEDINF command

13

Common Information Model

This chapter describes enhancements to the Common Information Model (CIM) for z/OS V1R10.

The following topics are discussed:

- An introduction to CIM as a short recap
- CIM Server currency and enhancements (J3X)
  - General updates and improvements
  - Write audit log to SMF records
  - Configuration changes through the MODIFY console command
  - PassTicket support with custom APPLID
  - RunAs security for providers
- OS management instrumentation update
  - Processor instrumentation enhancements
- CIM Client for Java Version 2

13.1 Introduction to CIM

The Common Information Model (CIM) is a standard data model developed by a consortium of major hardware and software vendors (including IBM) known as the Distributed Management Task Force (DMTF) as part of the Web Based Enterprise Management (WBEM) initiative. CIM was introduced in z/OS V1R7.

The idea behind CIM is to allow management applications from different vendors to manage a heterogeneous environment of systems via the same technology. All applications operate on the same set of common data (the standard CIM Schema), using the same access protocol (CIM-XML over http).

There is no need for vendors of management applications to either ship their own set of instrumentation or to install their own agent technology on the systems to be managed.

Specific attributes of the various platforms are still available through CIM and can be either ignored or dynamically discovered by management applications. It is also still possible to create management applications for a specific platform only, by exploiting the extended CIM classes created for that platform. In the future we will be able to avoid the installation of multiple management agents on the managed systems. The infrastructure for all types of management will be the generic CIM Server.

13.1.1 CIM cross-platform management

As shown in Figure 13-1 on page 293, with CIM, management applications from different vendors can manage a heterogeneous environment of systems via the same technology. All applications operate on the same set of common data, such as the standard CIM Schema, using the same CIM-XML over the HTTP access protocol. There is no need for vendors of management applications to either ship their own set of instrumentation or to install their own agent technology on the systems to be managed.

As shown in Figure 13-1 on page 293, a CIM client application requests the CIM Server to return information about z/OS resources, which in this case is about basic z/OS data as well as RMF metrics. The CIM Server invokes the appropriate CIM providers, which retrieve the requested data associated to z/OS system resources. The z/OS RMF monitoring provider invokes the RMF Distributed Data Server (DDS), which in turn collects RMF Monitor III performance data. The CIM Server consolidates the data from the providers and returns them back to the calling client through the CIM/XML over HTTP access protocol.

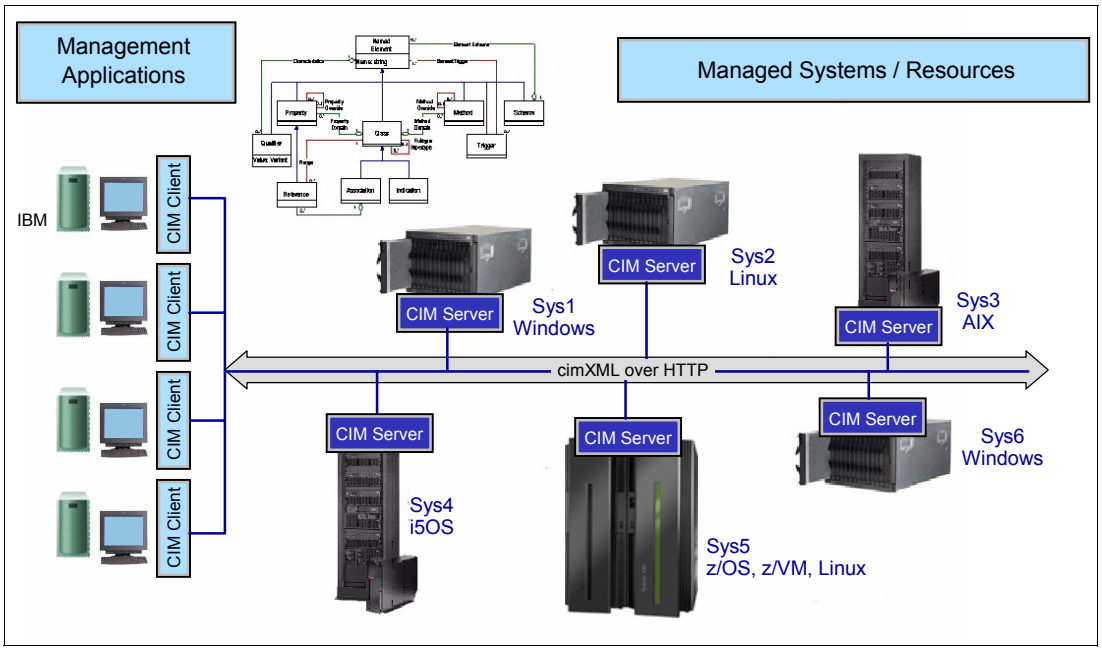


Figure 13-1 Cross platform management with CIM

13.1.2 CIM components and dependencies

The overall goal of this open architecture is to provide simplified systems management functionality through a common, easy-to-use interface. However, you cannot just “slap” a GUI onto the operating system and expect something worthwhile to happen. The ability to interact with z/OS resources requires a concerted effort from the user interface (UI), to a remote interface, to a set of abstractions that organize the management operations functions provided by the operating system into understandable resource models, down to the low-level functions that carry out these operations on z/OS itself.

With support for the CIM Server on systems running z/OS, users have the ability to access z/OS resources through an extendible industry standard model. The CIM Server, shown in Figure 13-2 on page 295, is used to receive client requests, collect the requested metrics/data from the managed system, and return the results to the client.

CIM Server support with z/OS V1R9

Starting with z/OS 1R9, the CIM Server has been exploiting the functionality of common event adapter (CEA). CEA is a z/OS component that provides the ability to deliver z/OS events to C language clients. A CEA address space is started automatically during initialization of every z/OS system. In order for the address space to start successfully, you must configure CEA to work with z/OS. Failure to do so will cause CEA to run in a minimum function mode. For details, refer to the .

CIM provider access control permits the Communications Server CIM providers to gather CIM data when the user ID associated with the client of the z/OS CIM Server is not defined as a superuser.

CIM providers

CIM data instrumentation is supplied by CIM components called *providers*. The CIM providers gather data on a system in support of the CIM classes. Clients can retrieve the data

through the Common Information Model Object Manager. On z/OS, this function is provided by the z/OS CIM Server.

This CIM providers function resides in the /usr/lpp/tcpip/lib directory. There is no configuration necessary to activate this CIM provider support. The z/OS CIM Server must be configured and activated for the data supported by the Communications Server CIM providers to be available to clients.

The z/OS Communications Server CIM classes are shipped with the z/OS CIM Server. The files that define these classes and any platform-specific properties are also installed in the /usr/lpp/tcpip/mof directory.

Using CIM provides a consistent, modeled approach to providing system interfaces. The CIM Server technology allows for clients to be running either on the same system or on other systems in the network.

The CIM providers describe and realize the model for each of the manageable resources on the system:

- They provide the XML interface for external callers
- They provide the interface code to enable an external caller (in this case, systems management applications) to obtain and alter state of various resources in the system

Note: IBM has developed providers for z/OS that support basic operating system information and some performance metrics. A CIM provider is the link between the CIM Server and the system. This interface allows CIM to access and manage the resources. Each CIM provider makes accessible the resources it represents in a standard way.

Cluster instrumentation

The cluster instrumentation work enables the cluster CIM provider to invoke the proper system interfaces to obtain and change the status of cluster-related resources:

- Internal services callable only by the Cluster Provider
- SYSREXX routines

In so doing, the providers extend the reach of sysplex systems management up to the CIM Server.

System REXX (SYSREXX)

System REXX enables an authorized program to invoke a REXX script without having to go through the typical setup and management of either invoking the TSO TMP or having to manage REXX environments. Hence, using a simple programming interface, it may be run outside the normal TSO/E or batch environments.

It enables a low-level program access to:

- Powerful parsing and string/character manipulation
- The ability to issue system commands and parse the results

It enables rapid development and deployment of system programmer tools, such as operations scripts and health checks.

Common event adapter

The z/OS common event adapter (CEA) enables a z/OS UNIX process to subscribe to various “legacy” asynchronous BCP events by type and matching criteria, as shown in Figure 13-3:

- WTO
- ENF
- Program-specified event

It extends the reach of “legacy” events to z/OS UNIX processes while components with existing events do not have to be reimplemented to communicate with CIM indication providers.

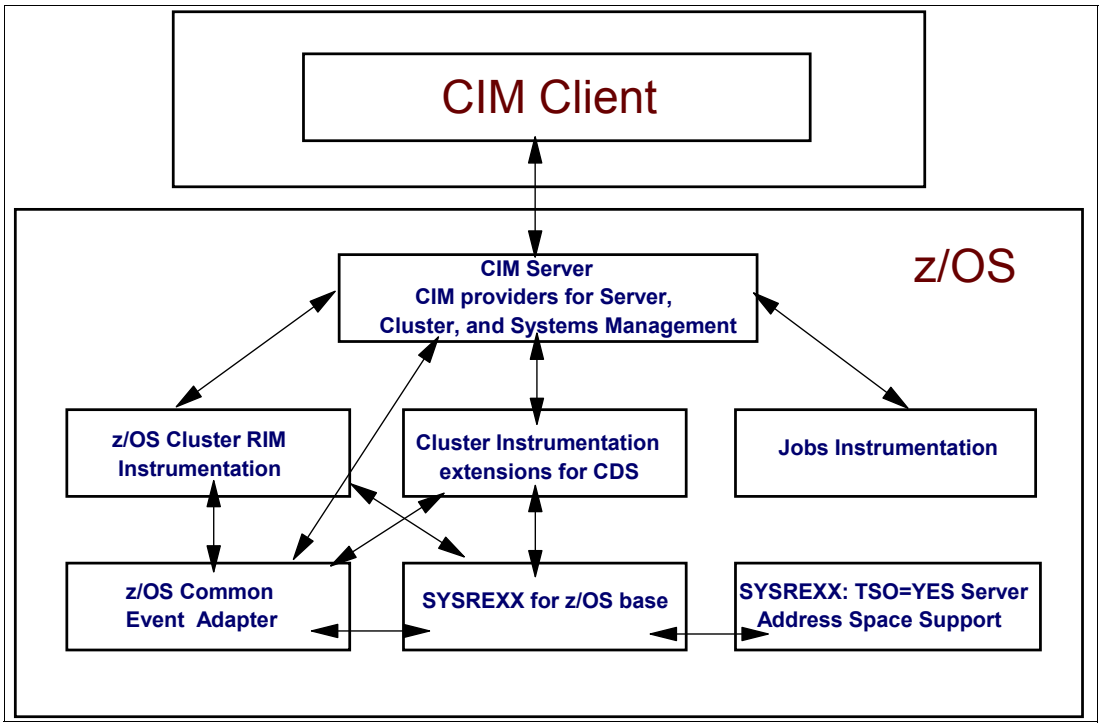


Figure 13-2 CIM components and dependencies

Jobs instrumentation

The jobs instrumentation enables the jobs and process CIM provider to invoke the proper system interfaces to obtain and change the status of batch jobs and z/OS UNIX processes; see Figure 13-3 on page 296.

The jobs instrumentation extends the reach of JES and z/OS UNIX System Services management up to the CIM Server.

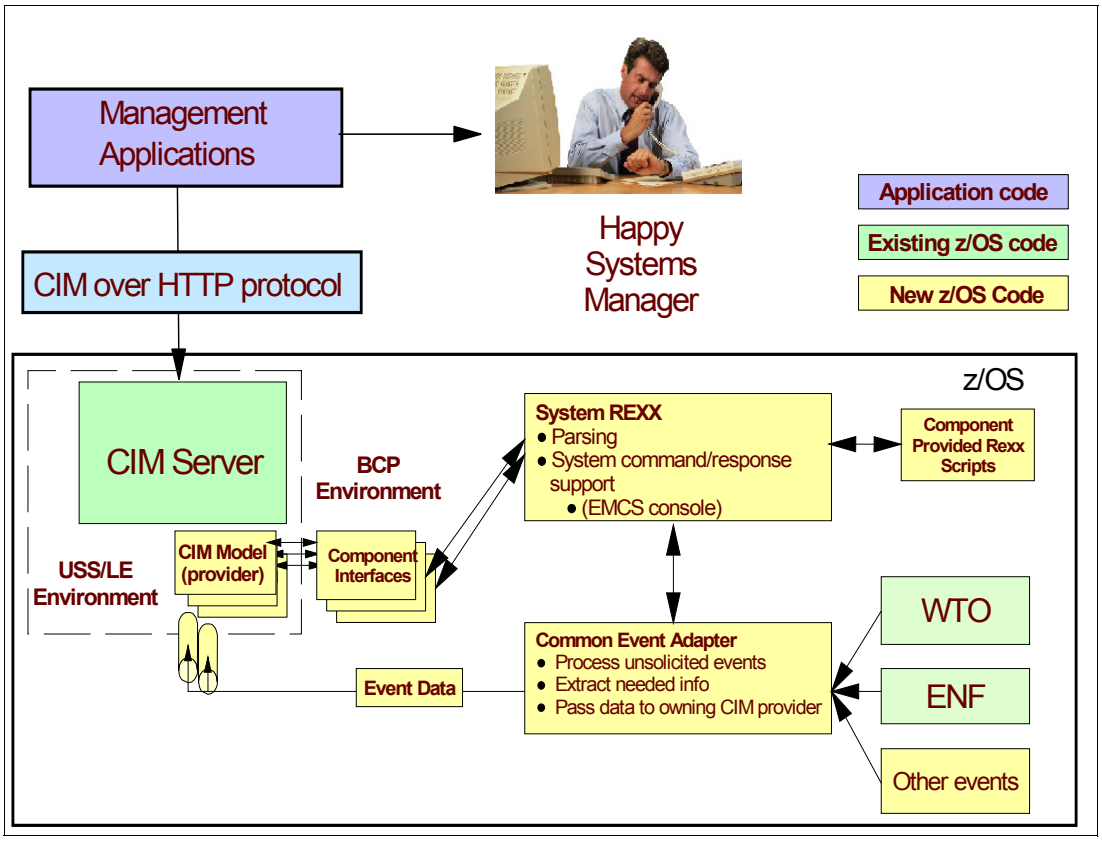


Figure 13-3 Cluster and jobs instrumentation

13.1.3 CIM Server overview

Since z/OS V1R9, the CIM Server has new components, as shown in Figure 13-4 on page 297. These new components are:

- CIM client API for Java

  The CIM client for the Java library from the SBLIM project is included with z/OS CIM. The CIM Client for Java is a programming API that enables z/OS applications written in Java for local and remote access of CIM instrumentation through the CIM-XML over HTTP access protocol. It consists of a Java library and associated online Java documentation.

- CIM Server runtime

  The CIM Server runtime environment security is split into authentication and authorization. Authentication is always enabled for the CIM Server. The CIM Server supports authorization via the RACF class WBEM, in which currently the single profile CIMSERV restricts access to the CIM Server. Since z/OS V1R9, this environment has been enhanced with the following function support:

  – General updates and improvements

  – Automatic Restart Manager (ARM) support

    To extend the server availability, the CIM Server is enabled to exploit the features of the Automatic Restart Manager (ARM). If an Automatic Restart Manager policy has been defined for CIM and the CIM Server is authorized to register with ARM, then the CIM Server will be automatically restarted by ARM.

- SSL certificate-based authentication

  The CIM Server is enabled to exploit the AT-TLS facilities to authenticate CIM clients. AT-TLS provides a feature to enable and use SSL/TLS connections and encryption for communication with the CIM clients. Now the CIM Server is aware of AT-TLS, and CIM clients can be authenticated through certificates.

- Logging facility is changed to use the syslog daemon.

- New command-line utility: cimsub

  The cimsub utility is a new CIM Server command-line utility that allows you to manage CIM indications on the local CIM Server. This command can list, enable, disable, and remove indication subscriptions, filters, and handlers.

- Operating system management instrumentation update

  The CIM base classes are extended with a new class, IBMzOS_LogicalDisk, which provides support for logical disk volumes. New instrumentation for job and sysplex management has been added for the management of jobs, sysplex, and Coupling Facility resources through CIM.

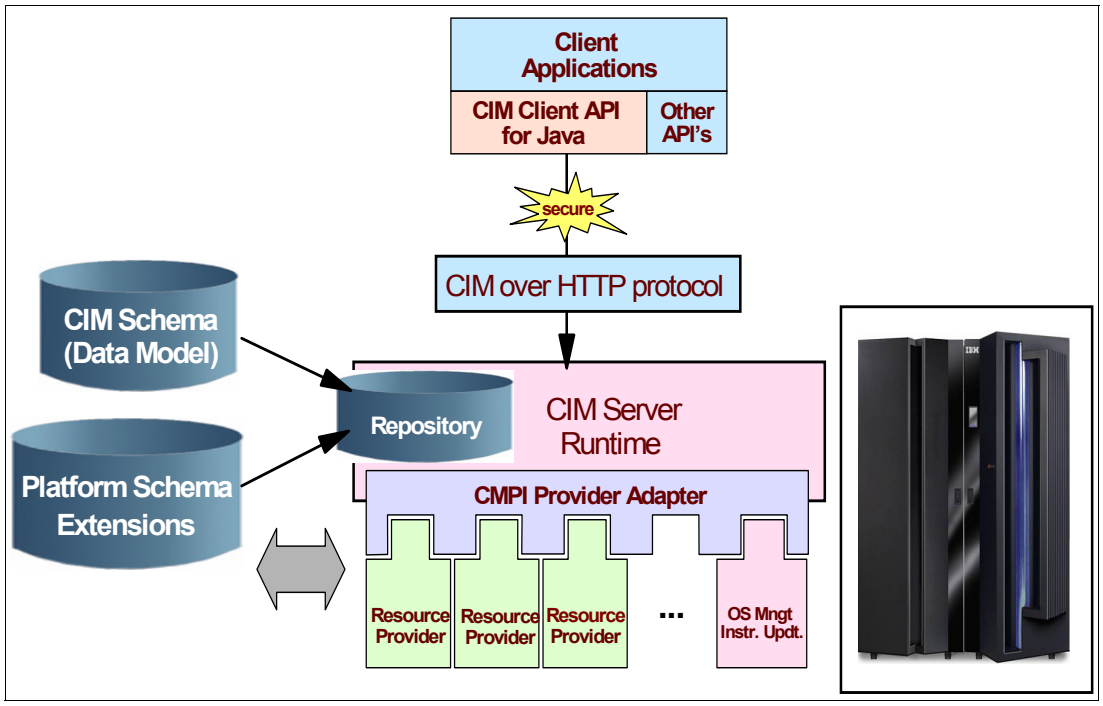


Figure 13-4 CIM Server overview

CIM security

Because this support allows for the modification of key system resources, maintaining appropriate authorization is key, as follows:

1. The CIM Server authenticates the end user.
2. The CIM Server checks the authority of the user based on provider-level checks.
3. The credentials of the user are propagated from the CIM Server, through the provider, down to the CEA, SYSREXX and other component interfaces, where authorization checks are made.

The secure check, shown in Figure 13-5, requires the following security customization definitions when CEA is to be started in full function mode:

- Configure CEA to work with z/OS by updating the RACF database to permit CEA to use the Automatic Restart Manager (ARM). Use this command:

```
ADDUSER CEA DFLTGRP(SYS1) OMVS(UID(0) HOME('/') FILEPROCMAX(1024)) SPECIAL
RDEFINE STARTED CEA.** STDATA(USER(CEA) GROUP(SYS1) TRACE)
```

- Define the OMVS segment that allows CEA to work in the UNIX environment. Use this command:

```
ADDUSER userid DFLTGRP(SYS1) OMVS(UID(0) HOME('/') FILEPROCMAX(1024))
SPECIAL
```

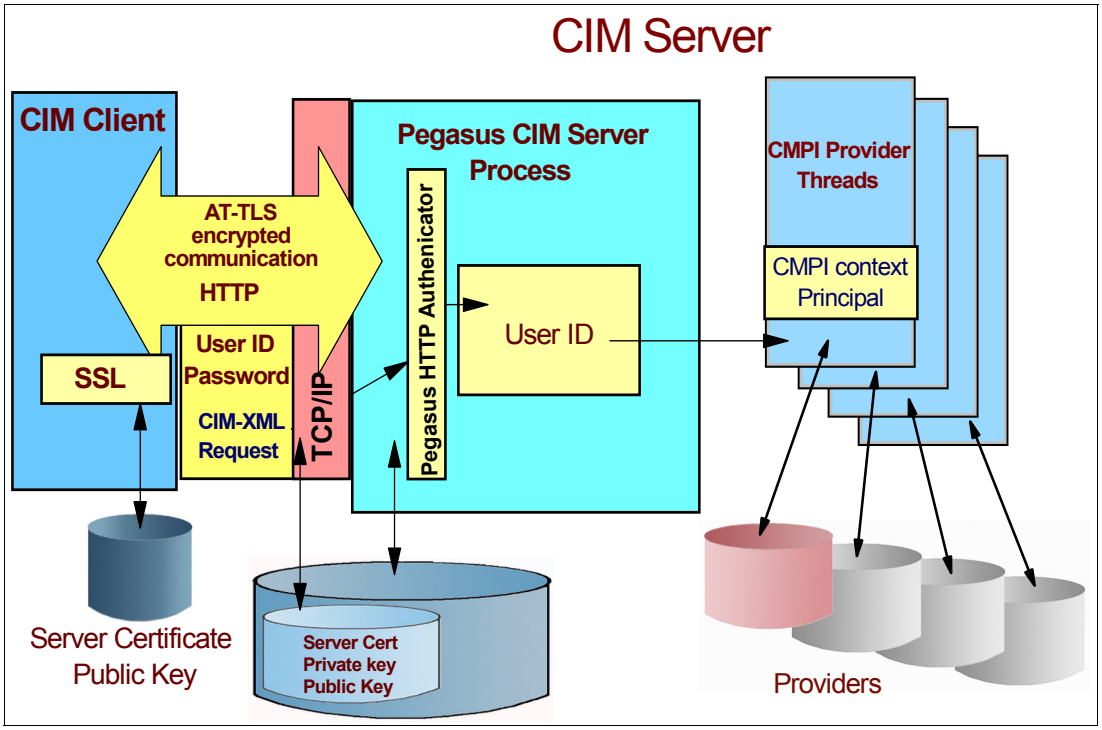


Figure 13-5 CIM client logging on to a CIM Server

The RACF user ID *may* be CEA but, because CEA is a started task, it does not *have* to be. If the STARTED class or started procedures table (ICHRIN03) contains another user ID, CEA will have that user ID assigned to it.

For example, a generic entry might specify that a user ID such as STCUSER or IBMUSER should be assigned to any started task that is not defined with its own entry. If the user ID CEA was not set up and assigned to the started task, then the generic entry would be used and an IEF695I message would indicate that START CEA was assigned to the generic user ID.

If the default user ID that is assigned does not have an OMVS segment, a default OMVS segment is sought through the FACILITY class profile BPX.DEFAULT.USER. RACF does not use the default OMVS segment unless the task is running with a RACF-defined user ID.

13.1.4 CIM client-to-CIM Server access

Communication between the CIM Server and a CIM client using AT-TLS and RACF uses a client's user ID and a password. Starting with release the z/OS V1R9 CIM Server, you can

use SSL certificates and encryption for communication with the CIM clients, because AT-TLS is enabled to authenticate CIM clients by SSL certificates in cooperation with RACF, as shown in Figure 13-5. The CIM client sends an SSL certificate to AT-TLS, AT-TLS sends the certificate to RACF, and RACF associates the certificate to the appropriate user ID, which then can access the CIM Server. Vice versa, the CIM Server returns its responses to the client's requests using SSL certificates.

CIM Server

For the CIM Server for z/OS, users log in over HTTP or HTTPS using basic authentication. When logging in, users are authenticated using their z/OS user ID and password as defined, for example, in the Resource Access Control Facility (RACF).

A CIM user must have at least READ access to the CIMSERV RACF profile in the WBEM class to access the CIM Server. In order to use any of the administrative command line tools of the CIM Server, a user instead requires CONTROL access to the CIMSERV profile.

The CIM Server authenticates users with the z/OS Security Server RACF to determine which users can log into it. Authentication is performed for every new connection (local or remote) before a user is granted access to the CIM Server, as shown in Figure 13-5 on page 298.

The CIM Server offers an optional authorization check. This check is optionally performed on a per provider basis, meaning that a RACF profile in the WBEM class can be related to a single provider library. Correlation between a provider and a RACF profile occurs during provider registration by the addition of a property in the PG_Provider class.

The provider-based authorization is defined by the vendor of a provider rather than by the CIM Server administrator. Therefore, specific RACF requirements will need to be documented on a per provider basis.

13.1.5 CIM enablement

z/OS CIM comes with the z/OS package and is installed via SMP/E. After a successful SMP/E installation, the components of z/OS CIM are located in the /usr/lpp/wbem/ hierarchical file system directory. The configuration files and repository are located in the /var/wbem/ directory.

The CIM Server runs as a daemon in z/OS UNIX. The first time installation of CIM is very straightforward. It requires a RACF security setup, a start procedure for the PROCLIB and some environment settings in z/OS UNIX. All of this is described in a short chapter of the *z/OS V1R10 Common Information Model User's Guide*.

Migrating from previous releases z/OS V1R8 or z/OS V1R9, is also very easy to do. There are some minor considerations that may be applicable to your own configuration that need to be checked before starting the CIM Server on z/OS V1R10. The migration itself will be done automatically by CIM.

During startup, the z/OS V1R10 CIM Server automatically corrects missing file tags. In addition, it detects whether an existing repository is up to date. If back-level, the CIM Server automatically upgrades the repository:

1. The CIM Server backs up the current repository in
   - repository_old19_<timestamp> for z/OS V1R9 repositories
   - repository_old18_<timestamp> for z/OS V1R8 repositories
   - vepository_old_<timestamp> for othe/wbem/repository.

2. The CIM Server migrates the previous repository content to the current repository.

If there are any quotes in the cimserver.env file in /etc/wbem, the CIM Server removes the quotes, stops the startup procedure and requires you to restart the CIM Server.

```
S CFZCIM
$HASP100 CFZCIM   ON STCINRDR
IEF695I START CFZCIM   WITH JOBNAME CFZCIM   IS ASSIGNED TO USER
CFZADM  , GROUP SYS1
$HASP373 CFZCIM   STARTED
IEF403I CFZCIM - s for 1 repository files.
CFZ17204I:  CIM Server passticket validation is using
CFZ10025I: The CIM Server is listening on HTTP port 5,988.
CFZ10028I: The CIM Serverl connection socket
CFZ10030I: Started CIM Server version 2.7.1 Development.
CFZ12532I: The CIM Server successfully registered to ARM using element name
CFZ_SRV_SC70.
CFZ00001I: PGS18204: SLP Registration Initiated
```

Figure 13-6 CIM Server startup

13.1.6 Automatic Restart Manager support

The CIM Server (only if started as a started task) automatically registers with Automatic Restart Manager (ARM), in which case it issues the following successful registration message:

```
CFZ12532I: The CIM Server successfully registered to ARM using element name
CFZ_SRV_SC63
```

The CIM Server will be deregistered from ARM during the normal shutdown procedure. In case of failure, the following message is issued and if you do not plan to use ARM, ignore the warning message:

```
CFZ12533W: The CIM Server failed to register with ARM using element name
CFZ_SRV_SC63 : return code 0x08, reason code 0x0134.
```

ARM element name

The ARM element name CFZ_SRV_<SYSNAME> has to be defined

```
DEFINE POLICY NAME(CFZARMPO) REPLACE(YES)

    RESTART_GROUP(CFZCIMRESGRP)
    /* List all systems where the CIM Server can be started  */
    TARGET_SYSTEM(SC63,SC70)
    /* Wait 10 sec before restarting to free resources       */
    RESTART_PACING(10)

    ELEMENT(CFZ_SRV_*)
      RESTART_ATTEMPTS(3,300)
      RESTART_TIMEOUT(300)
      READY_TIMEOUT(300)
      /* cross-system restart is not allowed.                */
      /* No restart after system failure                     */

      TERMTYPE(ELEMTERM)
      RESTART_METHOD(ELEMTERM,STC,'S CFZCIM')
```

Figure 13-7 Sample ARM policy

In a sysplex, no cross-system restarts are allowed, because there is always one CIM Server per MVS image.

SAF configuration for ARM

For security reasons, the IXCARM.DEFAULT.CFZ_SRV_* profile has to be defined in the FACILTY class of RACF and the CIM Server UID must have UPDATE access to that profile, as shown here:

```
RDEFINE FACILITY IXCARM.DEFAULT.CFZ_SRV_* UACC(NONE)
PERMIT IXCARM.DEFAULT.CFZ_SRV_* CLASS(FACILITY) +
       ID(CFZADM) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST (FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH
```

13.1.7 SSL certificate-based authentication

Starting with z/OS V1R9, the CIM Server provides an additional authentication mechanism using certificates, as shown in Figure 13-8 on page 302. The CIM Server uses the z/OS Communications Server Application TransparenPolicy Agent. The CIM Server is AT-TLS-aware, and it queries AT-TLS for the z/OS user ID.

Certificate management must be performed by an SAF-compliant product (for example RACF), which does the following:

- Stores CA and CIM Server certificates into key-rings.
- Maps certificate subjects to a z/OS user ID, either one-to-one or many-to-one.

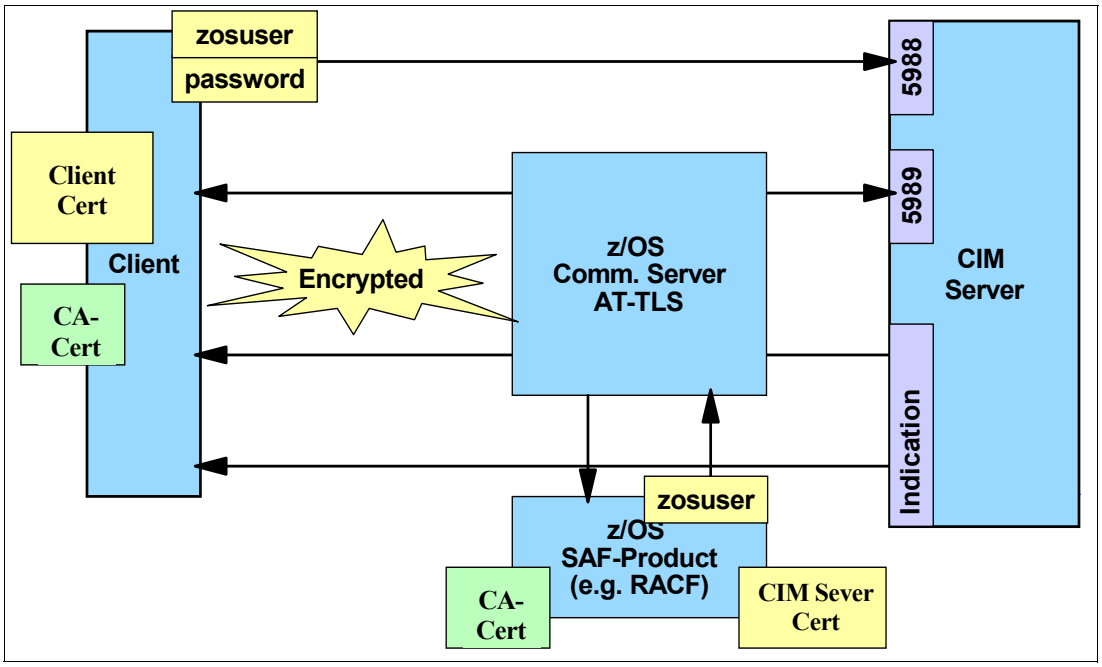


Figure 13-8 SSL certificate-based authentication

SSL certificate-based authentication configuration

The SSL certificate-based authentication configuration is executed through the **cimconfig** command, as follows:

```
-enableHttpsConnection=true
-httpsPort=5989
```

The AT-TLS configuration

The definitions are performed through a Policy Agent rule, which is part of z/OS Communications Server configuration, as follows:

```
“Jobname” or “Username” for policy selection.
“LocalPortRange” must match httpsPort of CIM Server
“HandshakeRole” set to ServerWithClientAuth
“ClientAuthType” set to SAFCheck
“TTLSKeyringParms” set to the SAF managed keyring
“ApplicationControlled” set to OFF
```

When using SSL client authentication with the z/OS CIM Server, in AT-TLS the parameter HandshakeRole has to be set to ServerWithClientAuth. Otherwise, the HandshakeRole has to be set to Server on the inbound AT-TLS policy configuration.

The z/OS CIM Server requires the HandshakeRole=ServerWithClientAuth or HandshakeRole=Server AT-TLS policy option to be set. If not set, the connection will be rejected.

13.1.8 CIM client API for Java

A CIM client API for Java is provided to address the requirement of a Java-based client API to exploit the CIM Server on z/OS; see Figure 13-9 on page 303. This is expected to enable vendors and other exploiters to write CIM client applications in Java on z/OS.

It supports the CIM-XML over HTTP(S) protocol. The interface, which is not yet standardized, uses JSR. The code is located at /usr/lpp/wbem/jclient. Version 1.3 is shipped under the form of a client library named sblimCIMClient.jar, with a configuration file located in cim.defaults. API Java documentation is available in sblim-cim-client-doc.zip. Further information can be found at:

```
http://sourceforge.net/projects/sblim/
```

The SBLIM CIM client is a pure Java-based implementation of the following:

- The WBEM operations API
- The CIM metamodel representation
- An indication listener

  In CIM terminology, an *indication* is the representation of the occurrence of an event. For example, an event can be the unexpected termination of a program, or the modification of a property value of a CIM instance. For example, an event can be the unexpected termination of a program, or the modification of a property value of a CIM instance. There is not necessarily a one-to-one correspondence between events and indications. In particular, multiple indications can be generated for the same underlying event if multiple CIM client applications had subscribed for the event.

  An event can also occur without causing a related indication to be raised (for example, if no subscription was made for the event).

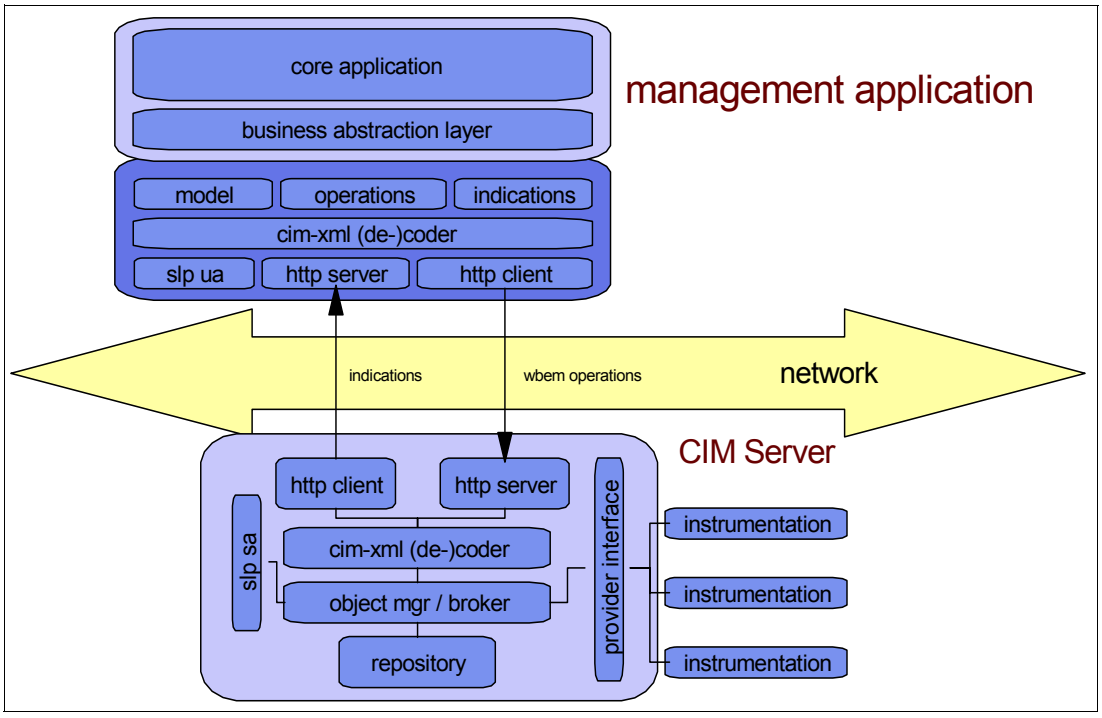


Figure 13-9 CIM client API for Java

13.1.9 CIM client/server implementation

The z/OS base element Common Information Model (z/OS CIM) implements the CIM Server, based on the OpenPegasus open source project. A CIM monitoring client invokes the CIM Server, which in turn collects z/OS metrics from the system and returns it to the calling client.

Figure 13-10 on page 304 illustrates how the CIM Server works in the z/OS environment, as follows:

1. A CIM client application requests the CIM Server to return information about z/OS resources, in this case about basic operating system (OS) data as well as RMF metrics.
2. The CIM Server invokes the appropriate CIM providers, which retrieve the requested data associated to z/OS system resources.
3. The z/OS RMF monitoring provider invokes the RMF Distributed Data Server (DDS), which in turn collects RMF Monitor III performance data.
4. The CIM Server consolidates the data from the providers and returns them back to the calling client through the CIM-XML over HTTP protocol.

In addition, Figure 13-10 shows two types of CIM providers:

- RMF monitoring providers that use the RMF DDS to access the z/OS system data

  A CIM monitoring client invokes the CIM Server which, in turn, collects z/OS metrics from the system and returns it to the calling client. To get the z/OS metrics, the CIM Server invokes the z/OS RMF monitoring provider, which retrieves the metrics associated with z/OS system resources. The z/OS RMF monitoring provider uses existing and extended RMF Monitor III performance data. The metrics obtained by this new API are common across eServer platforms, so you can use it to create end-to-end monitoring applications.

- z/OS operating system management providers that access the z/OS system data directly

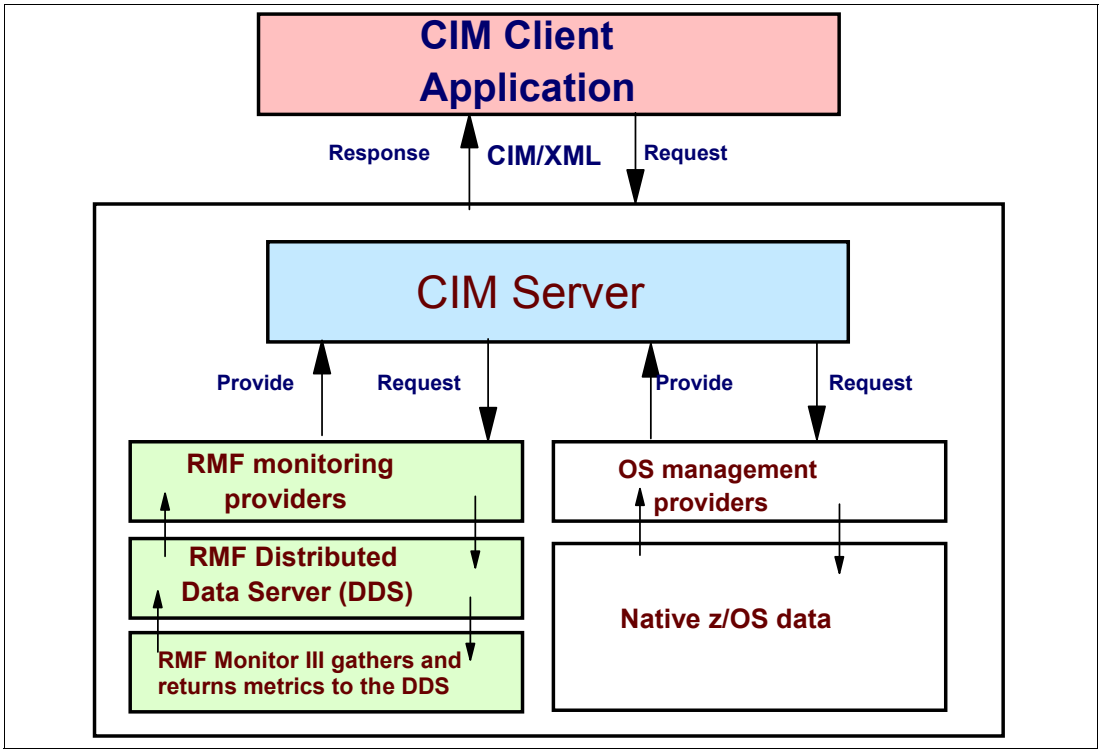


Figure 13-10 Client/server requests for data to z/OS using CIM

13.1.10 Required parmlib updates

The following parmlib parameters have to be defined to enable the job and cluster providers:

- MAXCAD limit

  This parameter defaults to 50. If the installation sets a lower limit, it may be necessary to increase this setting to accommodate the Common Event Adapter (CEA) Common Area Data Space (CADS).

- APF authorize SYS1.MIGLIB

  The following must be added to the installation's PROGxx member in PARMLIB to enable the CFRM-related CIM providers to function:

  ```
  APF ADD DSNAME(SYS1.MIGLIB) VOLUME(******)
  ```

- REXX alternate library

  The couple data set providers require the use of compiled REXX execs provided as part of SYSREXX. These execs require the use of the REXX alternate

  library. The following addition to the installation's PROGxx member in PARMLIB is one way to accomplish this:

  ```
  LNKLST ADD,NAME(LNKLST00),DSN(REXX.V1R3M0.SEAGALT),ATTOP
  ```

Note: A sample TSO CLIST is provided that performs the necessary RACF setup to permit CEA to use Automatic Restart Manager (ARM), and to permit CEA to operate in UNIX System Services (USS) with the cluster, couple data set, and JES2/JES3 jobs CIM providers.

13.1.11 CIM Server command-line utilities and commands

The CIM Server includes a set of command-line utilities that you can use to control or change the CIM Server environment. During normal use, you should rarely need to use these commands.

You must run all of the command-line utilities from a z/OS UNIX System Services shell. The command-line utilities need an environment setup. Users of these command-line utilities must have CONTROL access to profile CIMSERV in class WBEM.

Note: All of these utilities generate ASCII output. Without proper setup of your shell these utilities will show unreadable output.

CIFMOF command

The commands include **`cimmof`** and **`cimmofl`**. These commands are used to compile provider registrations and to compile CIM class descriptions written in the managed object format (MOF) language. The compiled information is put into the class schema stored in the repository. **`cimmofl`** is a version of the **`cimmof`** command that does not use the CIM Server. The CIM Server must be stopped before using this command. The usage of **`cimmofl`** is not recommended, since it bypasses the CIM Server and directly manipulates files in the file system.

13.2 z/OS V1R10 CIM Server enhancements

The major enhancements to the Common Information Model (CIM) for z/OS V1R10 are described in this section.

Installation of z/OS CIM with z/OS V1R10

After a successful SMP/E installation, the components of z/OS CIM are located in the hierarchical file system directory shown in Table 13-1.

Note: In Table 13-1, the directories shown in bold text are new with z/OS V1R10.

Table 13-1 Default installation directories for the z/OS CIM Server

Directory	Description
/usr/lpp/wbem	Base hierarchical file system directory
/usr/lpp/wbem/bin	CIM Server executables
/usr/lpp/wbem/lib	CIM Server libraries (dlls)
/usr/lpp/wbem/install	Sample profile
/usr/lpp/wbem/provider	CIM provider libraries provided with z/OS
/usr/lpp/provider/schemas	**IBM z/OS instrumentation MOF files**
/usr/lpp/wbem/msg	CIM Server message files for NLS
/usr/lpp/wbem/schemas	**CIM MOF schema files (CIM 2.13)**
/usr/lpp/wbem/repository	**CIM schema version 2.13 master repository**
/usr/lpp/wbem/jclient	**CIM client for Java version 1 and 2**
/usr/lpp/wbem/IBM	SMP/E target library path

The z/OS V1R10 enhancements for z/OS CIM are as follows.

Support of CIM Schema version 2.13

The CIM Server now supports CIM Schema version 2.13.

Audit logging with SMF records

The CIM Server can file audit log records to SMF record 86. These records contain information about authentication, configuration, provider status, and CIM operations.

MODIFY console command

Starting with z/OS V1R10, the CIM Server configuration can be changed from the z/OS system console using the MODIFY command.

Application ID CFZAPPL for authentication

The CIM Server is using the application ID CFZAPPL for authentication. This applies in using PassTickets and authorization check of the SAF APPL class.

Running providers in a designated user context

Previously, providers were always processed in the context of the user ID of a request. The provider ran under the identity of the requestor's user ID, and resource access authorization occurred against this user ID. So the requestor was required to be authorized for all resources that a provider was accessing when it processed a request. Starting with z/OS V1R10, you can process a provider alternatively in the context of a designated user ID. The designated user ID specifies a separate security context which is used to process the provider. The properties UserContext and DesignatedUserContext of CIM class PG_ProviderModule specify the provider's processing context.

CIM client for Java Version 2

Starting with z/OS V1R10, version 2 of the CIM client for Java is supported in addition to the version 1.3 CIM client for Java from the SBLIM project. One of the features provided by the CIM client for Java Version 2 is the support for embedded instances.

13.3 Support of CIM Schema version 2.13

The main objective for the CIM component is to stay up to date with the latest changes of the CIM/WBEM system management standard defined by the Distributed Management Task Force (DMTF).

In that perspective, z/OS V1R10 provides a new version of the CIM Schema version 2.13 and a new version of the OpenPegasus OpenSource CIM Server implementation (version 2.7). The detailed enhancements of these new versions are not subject of this paragraph. They can be obtained from the release status page or the following Web sites:

```
http://www.dmtf.org/standards/cim
http://www.openpegasus.org
```

One area of concern for previous releases of CIM has been the file tags of the CIM Servers data repository located in directory /var/wbem. When copied using the wrong command the file tags got lost and caused confusing error messages at CIM Server startup. To prevent this the CIM Server now checks all files in the repository at startup and restores the file tags where they were missing.

13.4 Write audit log to SMF records

The SMF record type 86 is used by the CIM Server. The SMF record mappings are divided into two sections, a common section which appears on all subtype records and a unique section for each subtype.

As depicted in Figure 13-11 on page 308, the CIM Server writes auditing information to a new SMF record type 86. To enable writing audit SMF record 86, modify SMF, the CIM Server, and the security configuration. The CIM Server can file audit log records to SMF record 86. These records contain information about authentication, configuration, provider status, and CIM operations.

Note: If the CIM Server audit logging is enabled, but SMF does not collect SMF record 86 or subtypes, or SMF is not enabled at all, no records are written.

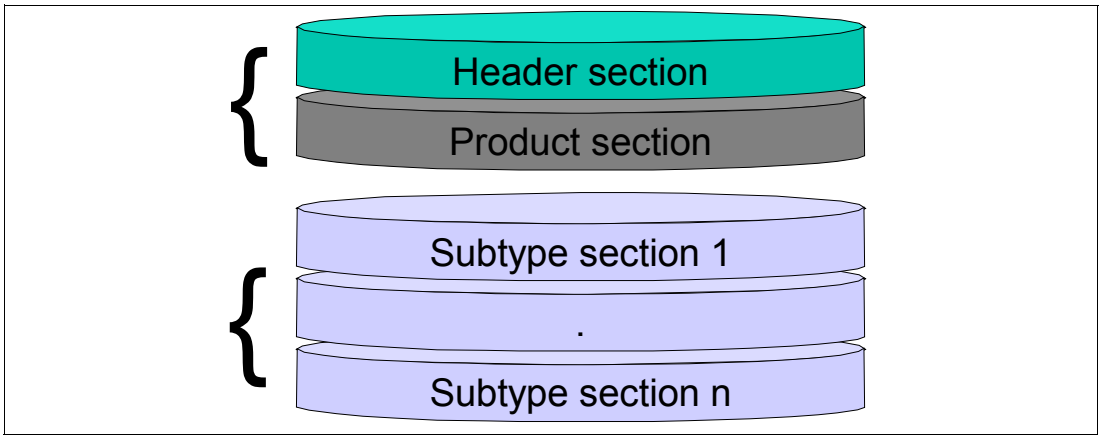


Figure 13-11 CIM audit log in SMF record type 86

Record 86 common section

As shown in Figure 13-11, the common section is divided into the header section and the product section, as follows:

Header section This section contains the common SMF record header fields and the triplet fields (offset/length/number), if applicable, they locate the other sections on the record. This triplet information should be checked prior to accessing a section of the record. All three fields being non-zero means that the section does not exist on the record. The “number” triplet field is the primary indication of the existence of the record.

Product section This section contains the general information about the server and the system that is running. The triplet SMF86PRO, SMF86PRL, and SMF86PRN of the header identifies this section.

Record 86 subtypes

The SMF type 86 subtype records contain information about authentication, configuration, provider status, and CIM operations, as follows:

Authentication The authentication record is written when a request user is authenticated to the CIM Server. Three different authentication mechanisms are supported by the CIM Server on z/OS:

- Basic - User Id and password (Passticket)
- Local - UNIX domain sockets
- ATTLS - Certificate validation with UserID mapping

All successful and unsuccessful requests are logged. For Basic and ATTLS, the IP of the client application is also recorded. The triplet SMF86AUTHO, SMF86AUTHL, and SMF86AUTHN of the header identifies this section.

Configuration In Subtype 2 the configuration of the CIM Server is monitored. Once Audit Logging is enabled by setting the configuration property enableAuditLog to true, the current configuration is recorded in this SMF record. Each CIM Server configuration value is written into an extra record. After that, all changes to the configuration are monitored. The triplet SMF86CONFO, SMF86CONFL, and SMF86CONFN of the header identifies this section.

Provider status In Subtype 3, the CIM Server registered provider modules are monitored. The status of the registered provider modules is recorded as follows:

- Current status - OK, Error, Stopped, etc.
- New status - OK, Error, Stopped, etc.
- Initial status - 0

The triplet SMF86PROVO, SMF86PROVL, and SMF86PROVN of the header identifies this section.

CIM operation In Subtype 4, CIM Operations are monitored:

- Operation type, name, and parameters
- Userid performing the operation
- IP address from where the requested operation was received
- Provider names and modules involved in the operation
- Operation result code

The triplet SMF86CIMOO, SMF86CIMOL, and SMF86CIMON of the header identifies this section.

CIM indications This is currently reserved for future use.

13.4.1 Using audit logs

The generation of audit logs for the CIM Server depends on the advanced configuration property enableAuditLog to be set for SMF record type 86. When this option is set to true, the CIM Server is writing SMF records 86. By default, enableAuditLog is set to false, preventing the CIM Server from even trying to write SMF records.

Ensure that SMF record type 86 is part of your active SMF configuration SMFPRMXX parmlib member.

The enableAuditLog configuration property can be dynamically turned on and off while the CIM Server is running. When it is turned on, for example using the MODIFY console command shown in 13.5, “Configuration changes with the MODIFY console command” on page 310, the CIM Server will first record its current status through SMF subtype 2 and 3 records for configuration and provider status. The command to enable SMF recording is as follows:

```
F CFZCIM,APPL=CONFIG,ENABLEAUDITLOG=TRUE,PLANNED
```

This command sets the CIM Server enableAuditLog configuration property to `true` since it can be dynamically switched on and off. Every time it is switched on the current status is recorded. The following messages are issued when the command is issued from the console:

```
CFZ0620I: Updated planned value for enableAuditLog to true.
CFZ0621I: This change will become effective after CIM Server restart.
```

Security setup for SMF recording

In order for the CIM Server to be authorized to write audit log records to SMF, the CIM Server user, for example CFZCIM, has to be given at least READ access to profile BPX.SMF in the RACF FACILITY class.With RACF, issue a PERMIT for the CIM Server user READ access to the BPX.SMF profile of the FACILITY class, as follows:

```
RDEFINE FACILITY BPX.SMF UACC(NONE)
PERMIT BPX.SMF CL(FACILITY) ACCESS(READ) ID(CFZCIM)
```

```
SETROPTS RACLIST(FACILITY) REFRESH
```

SMF recording messages

When SMF recording is switched on (enableAuditLog=true), a message is issued on the system console:

```
CFZ17805: Audit logging is enabled.
```

If SMF recording is switched off (enableAuditLog=false), a message is also issued on the system console:

```
CFZ17806: Audit logging is disabled.
```

This property can be changed dynamically during CIM Server runtime. If SMF audit logging is enabled for the CIM Server but SMF does not collect SMF record type 86 and subtypes is not enabled at all, message CFZ17805 is displayed anyway.

13.5 Configuration changes with the MODIFY console command

The CIM Server runs as a started task and is started and stopped from the system console. Until z/OS V1R9, configuration of the CIM Server occurred through the **cimconfig** command from the USS shell.

As z/OS system programmers and administrators prefer to use the system console, starting with z/OS V1R10, the setting of configuration options for the CIM Server is also enabled through the z/OS system console using the MODIFY command.

The syntax for the MODIFY command is

```
F jobname,APPL=CONFIG,property=[']value['][,PLANNED]

Where:
   Jobname - Usually CFZCIM
   APPL=CONFIG - Indicator for configuration changes
   property= - Configuration property to be changed
   [']value['] - New configuration value
   PLANNED - For permanent changes after CIM Server restart
```

Permanent changes are indicated using the `PLANNED` parameter. They are recorded in the following z/OS UNIX file:

```
/etc/wbem/cimserver_planned.conf
```

This requires a CIM Server restart to make the change effective. Without specifying `PLANNED`, changes are only effective until the next CIM Server restart. Case-sensitive configuration values need to be enclosed in quotes.

MODIFY command examples

When starting the CIM Server, for example:

```
S CFZCIM
```

With z/OS V1R10, the messages are:

```
CFZ12562I: Previous repository was renamed to
"/var/wbem/repository_old19_20080811191026" for backup and can be removed.
```

```
CFZ12563I: Automatic repository upgrade completed successfully.
CFZ17204I: CIM Server authentication is using application ID OMVSAPPL.
CFZ10025I:  The CIM Server is listening on HTTP port 5,988.
CFZ10028I: The CIM Server is listening on the local connection socket.
CFZ10030I:  Started CIM Server version 2.7.1.
CFZ12532I: The CIM Server successfully registered to ARM using element name
CFZ_SRV_SC65    .
CFZ18204I: SLP Registration Initiated
```

Note: During startup, the z/OS V1R10 CIM Server automatically corrects missing file tags. In addition, it detects whether an existing repository is up to date. If back-level, the CIM Server automatically upgrades the repository:

1. It backs up the current repository in:

   ```
   repository_old19_<timestamp> for z/OS V1R9 repositories
   repository_old18_<timestamp> for z/OS 1.8 repositories
   repository_old_<timestamp> for other z/OS repositories2.
   ```

2. It copies the master repository from /usr/lpp/wbem/repository to

   ```
   /var/wbem/repository
   ```

3. It migrates the previous repository content to the current repository.

If there are any quotes in the cimserver.env file in /etc/wbem, the CIM Server removes the quotes, stops the startup procedure and requires you to restart the CIM Server.

Command Example 1

In this first example tracing is turned on for a running CIM Server by setting traceComponents=all. This change becomes effective immediately, but will not last over a CIM Server restart.

```
F CFZCIM,APPL=CONFIG,TRACECOMPONENTS=ALL
CFZ06208I: Updated current value for traceComponents to all.
```

Command Example 2

In this example the ENABLEREMOTEPRIVILEGEDUSERACCESS property is set to true to allow superusers (UID=0) to use the CIM Server from remote. This change is made permanently and will only become effective after a CIM Server restart. To make administrators aware of this behavior, message CFZ06210I is displayed on the console.

```
F CFZCIM,APPL=CONFIG,ENABLEREMOTEPRIVILEGEDUSERACCESS=TRUE,PLANNED
CFZ06209I: Updated planned value for enableRemotePrivilegedUserAccess to true.
CFZ06210I: This change will become effective after CIM Server restart.
```

13.6 PassTicket support with custom APPLID

Authentication of requests against the z/OS CIM Server occurs either through SSL certificates or standard HTTP basic authentication. When using HTTP basic authentication, the HTTP header of a CIM requests carries the user ID and password in base-64 encoding.

To prevent a password from flowing between local CIM clients and the CIM Server, CIM is enabled to accept RACF PassTickets instead of passwords for HTTP basic authentication as shown in Figure 13-12 on page 312.

RACF PassTickets are cryptographically generated, single-use, short-lifespan password substitutes. They are inherently more secure than passwords.

To protect resources on the managed system from unauthorized access, first of all users have to be authenticated to ensure that the CIM Server is really communicating with a specific identity (user). In all cases after successful authentication, the user who wants to access the system is well known and now authorization checks against that specific user identity are executed.

Note: Authentication is always enabled for the CIM Server. The CIM Server checks whether the requestor is entitled to use the CIM Server. A requestor authenticates with a user ID and a password, with a user ID and a PassTicket, or with a user certificate.

PassTickets for authentication

PassTickets are generated from user ID, timestamp, and an application ID (APPLID). Though being a UNIX System Service application, the CIM Server uses a customer application ID of CFZAPPL, that has to be used when generating PassTickets for CIM Server logon. The CIM Server uses a distinct APPLID of CFZAPPL as opposed to using OMVSAPPL like other UNIX System Services daemons.

Note: For special purposes only, the CIM Server can use the application ID OMVSAPPL, if the CIM Server configuration property “enableCFZAPPLID is set to false at Server startup. In this case the system console message CFZ17204I indicates that the CIM Server PassTicket validation is using application ID OMVSAPPL.

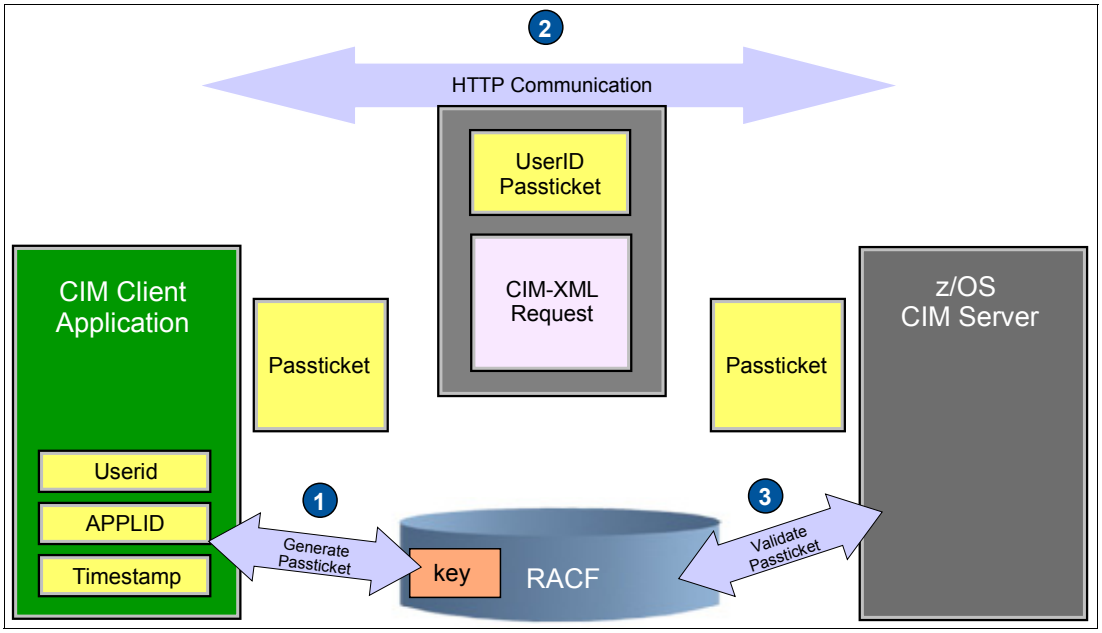


Figure 13-12 PassTicket support with custom APPLID

CFZAPPL application ID

The CIM Server application ID, CFZAPPL, needs to be configured in RACF before PassTickets can be used in requests against the CIM Server. If the APPL class for the security product is active, the CFZAPPL profile can be defined to allow only certain users to sign on to the CIM Server. You can manage access to the CIM Server application by a profile

for CFZAPPL in the APPL class with an access list that contains only those users who are allowed to use the CIM Server.

To enable PassTicket support for the CIM Server, define CFZAPPL in the PTKTDATA class in RACF, as follows:

```
SETROPTS CLASSACT (PTKTDATA)
SETROPTS RACLIST  (PTKTDATA)
RDEFINE PTKTDATA  CFZAPPL –
                  SSIGNON(KEYMASKED(<key>)
SETROPTS RACLIST(PTKTDATA) REFRESH
   with <key> being the 16 digit encryption key.
```

13.6.1 Using PassTickets for the CIM Server

There is nothing special about using PassTickets with the CIM Server. It is the same as for any other application for a CIMXML over HTTP request; the password has simply to be replaced bearing in mind that one PassTicket is only good for one single HTTP connection. PassTickets can be used for the CIM Server as for any other z/OS application.

Application implications

R_ticketserv (IRRSPK00) and R_GenSec (IRRSGS00) RACF callable services can also be used by client applications to generate PassTickets.

Java code can use a Java interface that uses a Java native interface (JNI™) and calls the r_ticketserv and r_gensec callable services. For information about this interface, see the JavaDoc shipped in the IRRRacfDoc.jar file, which is installed into the directory /usr/include/java_classes. Download the jar file to a workstation, un-jar it, and read it with a Web browser.

The example in Figure 13-13 is not a fully functional program but is just used to illustrate the key statements needed to generate a PassTicket and how to use it for connecting to the CIM Server from a Java application.

```
import com.ibm.eserver.zos.racf.*;
import org.sblim.wbem.client.*;
import org.sblim.wbem.cim.*;
...
IRRPassTicket irrPassTicket = new IRRPassTicket();
String passTicket = irrPassTicket.generate("CFZADM","CFZAPPL");

UserPrincipal userPr = new UserPrincipal("CFZADM");
PasswordCredential pwCred = new PasswordCredential(passTicket.toCharArray());

CIMNameSpace ns = new CIMNameSpace("http://boecfz1:5988","root/cimv2");

CIMClient cimClient = new CIMClient(ns,userPr,pwCred);

CIMClass cls = cimClient.getClass(
   new CIMObjectPath("CIM_ComputerSystem", "root/cimv2"));

cimClient.close();
...
```

Figure 13-13 Passticket support

13.7 Security for providers

By default all providers are always executed in the context of a requestor's user ID for all invocations that are caused by an external CIM operation. This means that the provider runs under the identity of the requestor's user ID, and resource access authorization occurs against this user ID. So the requestor is required to be authorized for all resources that a provider is accessing when it processes a request.

But for certain providers this behavior can be impractical, especially when it is not desired to grant a CIM client user ID global access to all the resources that a provider is using for gathering data. Instead the provider code itself should be authorized for the resources it accesses, and the provider itself performs custom authorization checks based on the requestors user ID.

13.7.1 Provider support with z/OS V1R10

Starting with z/OS V1R10, to avoid that a CIM client user ID has global access to all the resources that a provider uses for gathering data, a provider can be registered with a designated user ID. The designated user ID specifies a separate security context which is used to process the provider. The designated user ID must be authorized to access all the resources accessed by the provider. Instead of directly using a requestor's user ID when accessing the resource, the provider code now has to perform custom authorization checks based on the requestor's user ID, to prevent unauthorized access to resources.

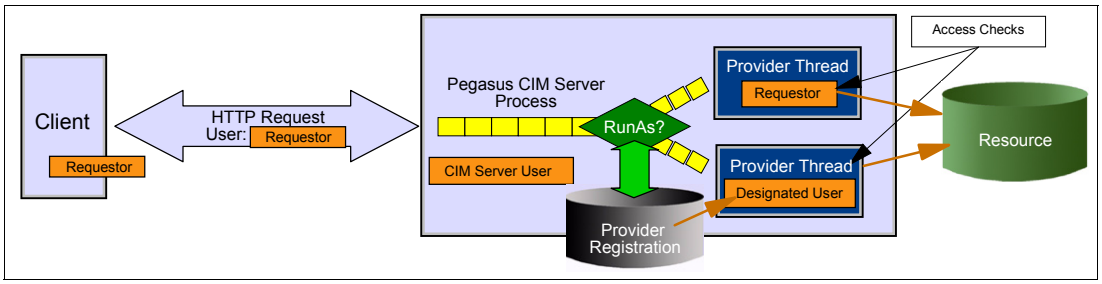


Figure 13-14 Security for CIM providers

Using providers with a designated user identity

The properties UserContext and DesignatedUserContext of CIM class PG_ProviderModule specify the provider's processing identity. You can specify the values for these properties in the provider registration MOF file for each provider module. By default, it is installed at /usr/lpp/wbem/provider/schemas/. The values for these properties can be specified as shown in Figure 13-15 on page 315.

CIM class PG_ProviderCapabilities describes the specific capabilities of a provider. Multiple instances of PG_ProviderCapabilities can be created for each provider allowing the same provider to be registered, for example, for multiple CIM classes.

Provider registration processing occurs once the provider registration MOF file has been created with the instances of classes PG_Provider, PG_ProviderModule and PG_ProviderCapabilities, the content of this MOF file can be loaded into the CIM Server's root/PG_InterOp namespace using the **`cimmof`** command.

```
Instance of PG_Provider
{
  ProviderModuleName = "RunAsProviderModule";
  Name = "RunAsProvider";
};
instance of PG_ProviderModule
{
   Name = "RunAsProviderModule";
   Location = "cmpi_RunAsProvider";
   Vendor = "IBM";
   Version = "2.0.0";
   InterfaceType = "CMPI";
   InterfaceVersion = "2.0.0";
   UserContext = 3; //Designated User
   DesignatedUserContext = "PEGUSR";
};
```

Figure 13-15 Example of an instance of class PG_ProviderModule in MOF syntax

CIMMOF command with z/OS V1R10

z/OS CIM commands are used to compile provider registrations and to compile CIM class descriptions written in the MOF language and store the information in the repository. For the **cimmof** command, the CIM Server must be started before using this command. cimmofl runs without using the CIM Server. This version of the MOF compiler does only limited error checking, may handle instance operations incorrectly, and does not protect against concurrent access to the CIM repository. Therefore, cimmof is the recommended MOF compiler.

Using the **cimmof** command for provider registration, as follows:

```
cimmof –n root/pg_interop <registration-mof-file>
cimmof -n root/PG_InterOp TestProviderRegistration.mof
```

CIMPROVIDER command

The **cimprovider** command is new with z/OS V1R10 and needs to be used to remove existing registrations, knowing that IBM registration mof files are installed in /usr/lpp/wbem/provider/schemas/. The **cimprovider** command also lets you disable, enable, as well as remove, and list registered CIM providers or CIM provider modules and the according module status.

For using the **cimprovider** command, the CIM Server must be running, and the user needs to have CONTROL access to profile CIMSERV in class WBEM.

Note: Generally it is the responsibility of the vendor of a provider (implementing a certain CIM class) to define when a provider should run under a designated user context and according documentation has to be supplied, which describes the specific setup steps needed. But in any case for the user ID which is defined as the DesignatedUserContext similar security definitions have to be made as for regular client users.

13.7.2 OS management instrumentation update

While the z/OS CIM Server supports all of the CIM operations from the DMTF's CIM Operations over HTTP specification, only a specific subset of operations is supported by the

OS management CIM providers delivered with this release of z/OS. The CIM_Processor class represents the physical processors that are available to the operating system. The z/OS specific subclass is IBMzOS_Processor.

The current CIM class, IBMzOS_Processor, now with z/OS V1R10 also supports “reserved” processors. In previous versions of CIM only “online” processors were reported. To expose the state and type of a processor, properties Role and enabledState have been implemented.

```
Property Role shows processor type (zIIP, zAAP or CP)
Property EnabledState shows processor state (Online, Offline,...)
```

Use the CIMXML over HTTP operations to obtain the new processor attributes, shown in color in Figure 13-16.

Property Name	Description
Caption	“***zSeries logical processor***”.
Description	***“This class represents instances of processors currently available to the z/OS operating system***”.
Name	CPUID of the physical processor (PCCACPID)
DeviceID [key]	CPUID:<processor number>
Role	CP, zIIP, zAAP, …
Family	200 (=“S/390 and zSeries Family”)
OtherFamilyDescription	“S/390 and zSeries Family” or specific model like “z990”
UniqueID	CPUID of the physical processor (PCCACPID)
EnabledState	Online(2), Offline(6), Reserved(3), Offline by WLM(9)
. . .	...

Figure 13-16 IBMzOS_Processor instrumentation

13.8 CIM client for Java Version 2

Since z/OS V1R9, the CIM client for the Java library from the SBLIM project is included with z/OS CIM. For z/OS V1R10, Version 2 of the CIM client for Java is included. The CIM client for Java is a programming API that enables z/OS applications written in Java for local and remote access of CIM instrumentation through the CIM-XML over HTTP access protocol. It consists of a Java library and associated online Java documentation.

With z/OS V1R10, the CIM Client for Java version 2 is supported, as follows:

- New version is not compatible with Version 1.3 because of a different interface.
- New version is better, faster and more efficient.
- New and more flexible configuration management.
- New and flexible logging and tracing.
- New implementation of XML parsers for better scalability of large requests.
- Full SSL/TLS support including trust stores (through Java).

The code is located at /usr/lpp/wbem/jclient.

- Version 1.3

```
sblimCIMClient.jar
sblim-cim-client-doc.zip
```

- Version 2

```
sblim-cim-client2.jar
sblim-cim-client2-doc.zip
```

For information about this interface, see the JavaDoc shipped in the sblim-cim-client2-doc2.zip file. Download the archive file to a workstation, extract it, and read it with a Web browser.

A new exploiter of CIM is available with z/OS V1R10, called the Capacity Provisioning Manager; see 6.1, “z10 EC and Capacity Provisioning”.

13.9 Migration and coexistence considerations

Migrations from z/OS V1R8 and z/OS V1R9 should consider the following steps:

- Automatic repository upgrade to DMTF CIM Schema Version 2.13 at CIM Server start-up.

```
CFZ12552I: Starting automatic repository upgrade.
CFZ12563I: Automatic repository upgrade completed successfully.
```

  The CIM Server staring updates /var/wbem/repositor, repairs file tags in /var/wbem/repository, and repairs errors in /etc/wbem/cimserver.env

The following updates are recommended:

- Replace the CFZCIM started task procedure with the z/OS V1R10 version.
- Update /etc/wbem/cimserver.env.
- Copy /usr/lpp/wbem/cimserver.env.

Coexistence items should be considered

APAR OA22914 or later should be installed on z/OS V1R9 for consistent usage of the CFZAPPL application ID.

Post-install customization is required by doing the following:

- Run CFZRCUST only on initial install before starting the CIM Server for the first time.
- When migrating from a previous release, the CIM Server will automatically run the required customization updates on a first time start.

Tip: The /var/wbem directory cannot be shared.

Unlike what is available in z/OS V1R9, CIM now ships as a single FMID, as shown in Figure 13-17.

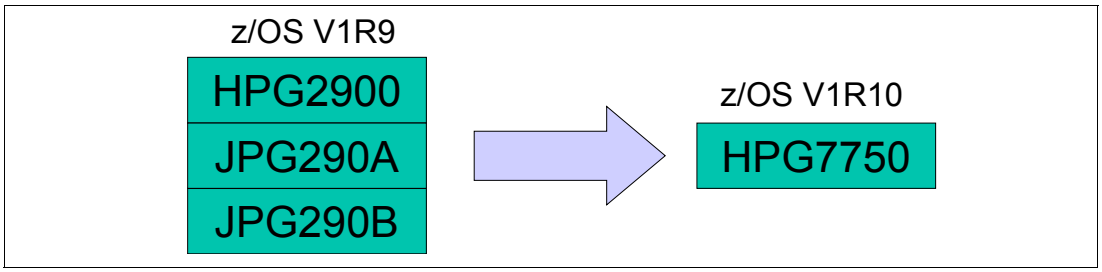


Figure 13-17 CIM FMID

Installation directory considerations

For each target system:

- Create its own /etc/wbem directory and /var/wbem.
- May copy the files/directories from /usr/lpp/wbem/repository.
- Place the /var/wbem on an extra file system data set (about 150 MB zFS).

The SMP/E installation of the z/OS V1R10 CIM Server installs a new master repository in /usr/lpp/wbem/repository. Existing versions of the repository are located in /var/wbem/repository. Then take the following steps:

1. On each target system, mount a system-specific data set of at least 100 MB size at /var/wbem, which either holds the z/OS V1R8 or z/OS V1R9 repository or is empty.
2. In the file /etc/wbem/cimserver_planned.conf, ensure that the logdir, enableNamespaceAuthorization and the httpAuthType properties do not exist. If they exist, delete them.
3. It is recommended to replace the environment file cimserver.env located in /etc/wbem with the new sample installed in this directory:

   ```
   /usr/lpp/WBEM
   ```

4. If you do not intend to replace the environment file cimserver.env with the new sample, make sure that the following directories are included in the LIBPATH defined in cimserver.env:

   ```
   /usr/lpp/wbem/lib:/usr/lpp/wbem/provider:/usr/libCIM
   ```

CFZRCUST job

The CFZRCUST job is for installing and migrating the z/OS CIM Server configuration and repository on each target machine. A sample of CFZRCUST is shipped with the default SAMPLIB. CFZRCUST is for the IPL'ed target system. The CIM repository is a file-based information store where all files have to be tagged ASCII (ISO8859-1), as depicted in Figure 13-18 on page 319. When installing with CFZRCUST, take the following steps:

1. The target system is running with configured UNIX System Services.
2. The CIM Server must be stopped.
3. The user running this job must either have UNIX user ID 0 or must be able to copy files and set the program control bit on files.
4. If you intend to mount the data on a separate file system, which is recommended, this user must be entitled to allocate a 100 MB hierarchical file system or zFS data set (if not yet allocated).

Adjust the sample job CFZRCUST, which is located in the SAMPLIB, to fit your environment.

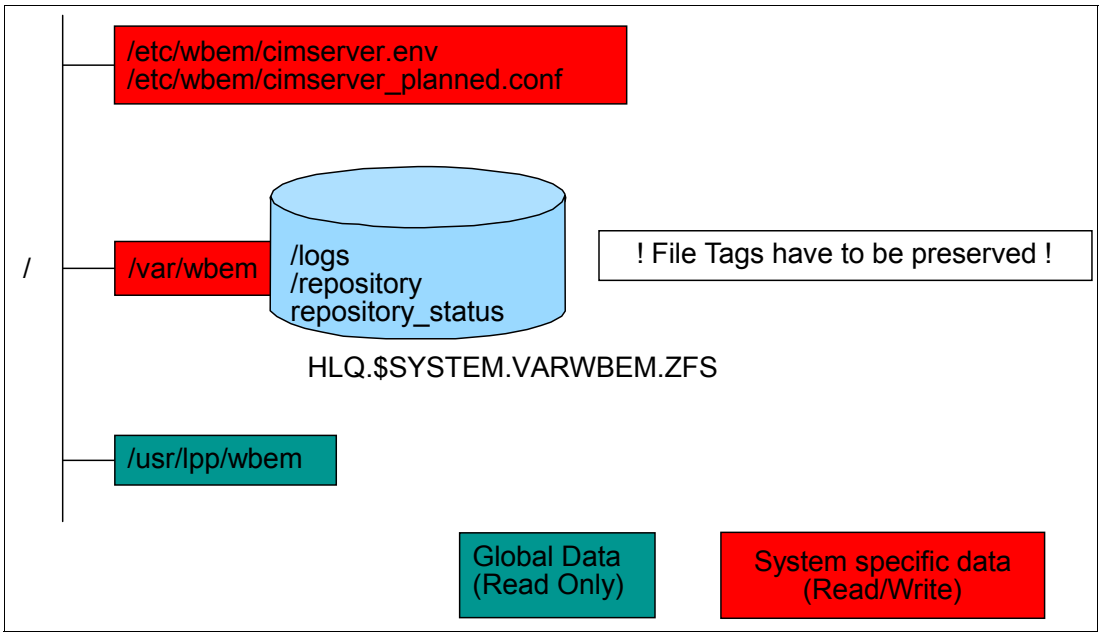


Figure 13-18 CIM directory installation

14

XML enhancements

XML allows you to tag data in a way that is similar to how you tag data when creating an HTML file. XML incorporates many of the successful features of HTML, but was also developed to address some of the limitations of HTML. XML tags may be user-defined, by either a DTD or a document written in the XML Schema language, that can be used for validation. In addition, namespaces can help ensure that you have unique tags for your XML document. The syntax of XML has more restrictions than HTML, but this results in faster and cheaper browsing. The ability to create your own tagging structure gives you the power to categorize and structure data for both ease of retrieval and ease of display. XML is already being used for publishing, as well as for data storage and retrieval, data interchange between heterogeneous platforms, data transformations, and data displays. As these XML applications evolve and become more powerful, they may allow for single-source data retrieval and data display.

This chapter describes the new enhancements to the XML part of z/OS:

- z/OS XML integration for validation
- z/OS XML Improved IPCS support
- zIIP 100% Offload – z/OS XML Enablement
- z/OS XML Enhanced Coding Support
- z/OS XML support for the XML Toolkit
- z/OS XML support for the COBOL compiler

14.1 z/OS XML with z/OS V1R10

z/OS V1R10 is planned to enable additional XML processing to be made eligible for the zIIP and zAAP specialty processors. Enhancements in z/OS XML System Services and the XML Toolkit for z/OS (5655-J51) are planned to increase the amount of XML workload eligible for the zAAP and zIIP specialty engines. IBM middleware (such as DB2 9) and other products are expected to be able to benefit from this new functionality.

In z/OS V1R10, IBM provides these functions in z/OS XML System Services, as follows:

Additional zIIP exploitation

z/OS XML System Services plans include additional zIIP exploitation, specifically enabling all z/OS XML parsing in enclave SRB mode to be eligible for zIIP. For example, with respect to DB2, z/OS XML processing may be partially directed to zIIPs when utilized as part of a distributed request (like DB2 DRDA) today. This enhancement can help further benefit DB2 pureXML® workloads by optionally directing all z/OS XML System Services parsing that is executed in enclave SRBs to the zIIP. This function is planned to be available on z/OS V1R8 and V1R9 with PTF for APAR OA23828.

A validating parser

Validation support is designed to allow a program to determine whether an XML document meets the requirements expressed in an XML schema definition (XSD). z/OS XML System Services has added a validating parsing. z/OS XML System Services validating parsing workloads are eligible for zIIP and zAAP processing.

Support for 19 additional code pages

This extends XML System Services processing to accommodate the character sets used in many additional languages. This function is also available on z/OS V1R7, V1R8, and V1R9 with PTF for APAR OA22777. The source offset support is designed to make it easier to locate or extract specific data from within an XML document.

XML Toolkit for z/OS

The XML Toolkit for z/OS is enhanced so that eligible workloads can use z/OS XML System Services. This allows eligible XML Toolkit processing for non-validating parse requests to exploit the zAAP. This function is planned to be available on the XML Toolkit for z/OS V1.9 with SPE. XML toolkit support for processing validating parse requests using z/OS XML System Services, with the appropriate offload of eligible work to zAAP, is planned at a future date.

Enterprise COBOL V4.1

Additionally, Enterprise COBOL V4.1 provides support for a new XMLPARSE compiler option to allow COBOL programs to use XML System Services for XML processing and take advantage of zAAP specialty processors.

14.2 z/OS XML integration for validation

The only z/OS parsing solution that provided validation was the XML toolkit. Poor performance made this solution too costly in the past. In z/OS V1R10, the XML validating parser is directly available to users through existing z/OS XML APIs. XML documents have characteristics that affect the way they are parsed, and the kinds of information that the

parser generates during the parse process. One such characteristic is the encoding scheme of the document, which the z/OS XML parser must know before parsing. Using the query service, gxlpQuery, will allow the caller to acquire this information, after which it can then pass it to the z/OS XML parser. The z/OS XML parser will then be able to use the correct encoding scheme to parse the document.

z/OS XML System Services includes the following three primary new functions (Figure 14-1 on page 324):

- A query service that allows callers to determine the encoding of the document and acquire information from the XML declaration.
- Parsing with schema validation.
- Parsing without validation.

The XML validating parser is invoked when a new validation feature is specified during parser initialization. Any caller of z/OS XML will now have the option to request a validating parser through the externals outlined here. Current users may continue to use the existing parser functions without change to their code. This integration enables the XML validating parser to offload to a zIIP engine.

Parsing XML documents without validation

The non-validating parse process consists of three fundamental steps: initialize the parser, parse the document, and terminate the parser. Multiple documents may be parsed using either a single instance of the parser, or several distinct instances as the caller requires.

Parsing XML documents with validation

Parsing with validation follows the same basic process as for a non-validating parse. The key difference is an additional step to load a preprocessed version of the schema used to validate the document during the parse. This binary schema, referred to as an Optimized Schema Representation (OSR) can be loaded once, and used to validate any document that conforms to it.

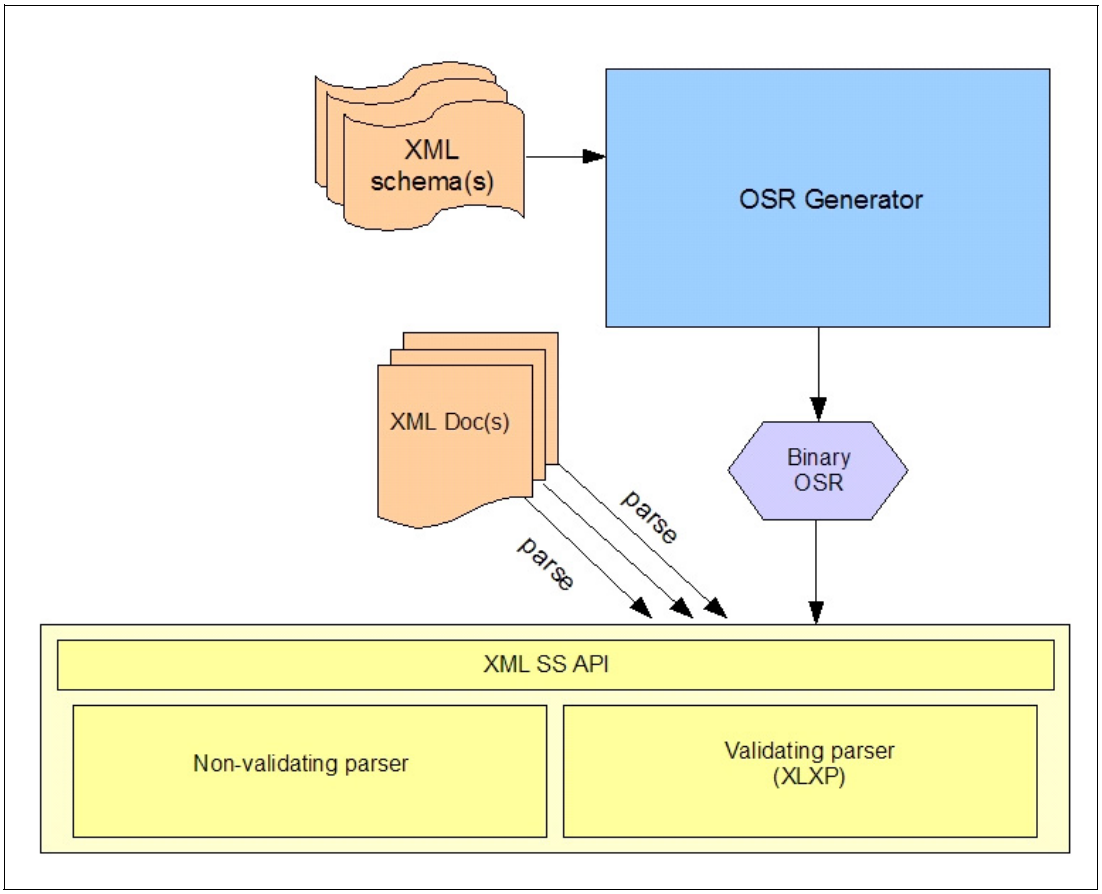


Figure 14-1 Calls to z/OS XML System Services

14.2.1 z/OS XML parser APIs

The following APIs are new with z/OS V1R10:

Support for the Metal C compiler option

Support is provided for callers who wish to use the Metal C compiler option. The same APIs available to the standard C and C++ callers are also available to Metal C users, with the following restrictions:

- All parameters must be variables.
- The functions do not return values.

Optimized Schema Representation

Optimized Schema Representations (OSRs) are preprocessed versions of schemas. They are more easily and more efficiently handled than schemas in text form. When parsing with validation, this form of schema is utilized. An OSR API is provided to assist in the generation, loading and manipulation of these specialized schemas. Optimized Schema Representations are specialized forms of schemas used during the validating parse process. They can be created from utilities provided by the OSR generator API.

There are new added C APIs to act with the validating parser:

- **gxluInitOSRG()** - Initialize an OSR generator instance
- **gxluLoadSchema()** - Load a schema into the OSR generator

- **gxluLoadOSR()** - Load an OSR into the OSR generator
- **gxluSetStrIDHandler()** - Specify the stringID handler for OSR generation
- **gxluSetEntityResolver()** - Specify the entity resolver for OSR generation
- **gxluGenOSR()** - Generate an Optimized Schema Representation
- **gxluGenStrIDTable()** - Generate a stringID table from an OSR
- **gxluControlOSRG()** - Perform an OSR generator control operation
- **gxluTermOSRG()** - Terminate an OSR generator instance

All new features are passed to the parser at initialization time through the feature flags parameter of gxlpInit. In addition to the validation flag above, this example is requesting that fully qualified names be generated for the records representing the end of an element in the parsed data stream. Validating parses are performed against Optimized Schema Representations (OSRs). Figure 14-2 shows how to use an OSR once it is created. For an example of how to create an OSR from a schema, see *z/OS XML System Services User's Guide*, SA23-1350.

Before an OSR can be used for validation, a control operation is required to load it. Once the parser is initialized with the validation feature specified, and an OSR is loaded, parsing proceeds in the same way as for non-validating parse requests.

```
#include "gxlhxml.h"

int main()
  {                                /* start of program               */
   void *pPIMA;                    /* parse instance memory area     */
   long  lPIMA;
   int   ccsid;                    /* document encoding              */
   int   fFeatures;                /* parsing features to enable     */
   int   rc, rsn;                  /* return and reason codes        */

   lPIMA = GXLHXEC_VPARSE_MIN_PIMA_SIZE;
   pPIMA =(void*) malloc(lPIMA);
   < if malloc failure, take an error path … >
   ccsid     = GXLHXEC_ENC_IBM_1047;
   fFeatures = GXLHXEC_FEAT_VALIDATE | GXLHXEC_FEAT_FULL_END;

   gxlpInit(pPIMA, lPIMA, ccsid, fFeatures,
            NULL, NULL, &rc, &rsn); /* initialize the parse instance   */

   if (rc == GXLHXRC_SUCCESS)
     {                             /* parser initialized             */
      void    *pOSRBuf;            /* OSR buffer for validation      */
      void    *pCTLData;           /* parser control data area       */
      GXLHXOSR OSRCtl              /* OSR control area               */
      char     OSRName[GXLHXOSR_MAX_OSR_NAME_LEN];

      < read an OSR into a buffer pointed to by pOSRBuf … >
      memcpy(OSRName, "PersonnelSchemaOSR", 18);
      memset(&OSRCtl, 0, sizeof(GXLHXOSR));
      pCTLData = (void *)&OSRCtl;   /* control data is for OSR load   */
```

Figure 14-2 How to use an OSR once it is created

Note that the OSR load is just one of the control operations that can be performed through the gxlpControl service. In order to facilitate both input to, and output from the control service to the caller, we pass a pointer to a pointer to the control data. This second level of indirection allows the control service to return data back to the caller.

Once an OSR is loaded via gxlpControl, any XML document that conforms to that schema can be parsed with validation. This is why the example continuation shown in Figure 14-3 shows a single OSR load followed by several parse requests.

The input buffer for the source XML document in this example has an arbitrary length of 1 MB, and the output buffer that receives the parsed data stream is 2 MB in length. In general, the parsed data stream will be significantly longer than the source XML document, although it's not possible to say exactly how long. The actual length required for the output buffer depends on the particular characteristics of the source document being parsed.

```
OSRCtl.XOSR_OSR_PTR      = pOSRBuf;
      OSRCtl.XOSR_OSR_NAME_PTR = OSRName;

      gxlpControl(pPIMA, GXLHXEC_CTL_LOAD_OSR, &pCTLData,
                  &rc, &rsn);   /* load the OSR                    */

      while((rc == GXLHXRC_SUCCESS) &&
            (< there are documents to parse >))
        {                       /* validate/parse with this schema */
         void *inBuf[0x00100000];  /* the XML document to parse    */
         long  lInBuf;
         void *outBuf[0x00200000]; /* the parsed data stream       */
         long  lOutBuf;
         int   fOptions = 0;    /* reserved flags – always zero    */

         < read the next xml doc to parse into inBuf … >

         gxlpParse(pPIMA, &fOptions,
                   &inBuf, &lInBuf, &outBuf, &lOutBuf,
                   &rc, &rsn);  /* parse the input document        */

         if (rc == GXLHXRC_SUCCESS)
           {                    /* parse succeeded                 */
            GXLHXEH_RECORD *pRecord = (GXLHXEH_RECORD *)outBuf;

            while(< there are records in outBuf that we haven't seen >)
              {                 /* consume the parsed data stream  */
               < use the data in this parsed datastream record >
               pRecord = pRecord + pRecord->XEH_RecLen;
              }                 /* consume the parsed data stream  */
```

Figure 14-3 Continuation of the OSR example

The last part of the example shown in Figure 14-4 shows how the parser is reset to prepare for the next document to be parsed. If a different OSR is required to validate the next document, it should be loaded before the parse.

```
gxlpControl(pPIMA, GXLHXEC_CTL_FIN, &pCTLData,
                    &rc, &rsn);/* reset parser for the next doc   */

       }                        /* validate/parse with this schema */

     gxlpTerminate(pPIMA, &rc, &rsn); /* end the parse instance   */
    }                           /* parser initialized              */

 }                              /* end of program                  */
```

Figure 14-4 Last part of the OSR example

JAVA APIs added

For a list of the methods available in the gxlOSRGenerator class, see the Java documentation included with z/OS V1R10 z/OS XML for more information. The gxlOSRGenerator class allows the user to generate an OSR from one or more conventional XML schemas.

- **newOSRGenerator()** - Constructor
- **loadSchema(java.lang.String uri)** - This method loads a single schema into the OSR generator.
- **loadSchema(java.lang.String[] uris)** - This method loads one or more schemas into the OSR generator.
- **controlOSRG()** - This method provides operations for controlling the z/OS XML OSR Generator. This can be used to reset the generator to allow for processing multiple schemas in one generator instance.
- **setStrIDHandler()** - This method registers a stringID handler with the OSR generator.
- **setEntityResolver()** - This method registers an XMLEntityResolver with the OSR generator.
- **genOSR()** - This method generates an Optimized Schema Representation (OSR).

Note: The Java API does not have a gxluGenStrIDTable()-like method, nor a termination method. Termination and clean-up are handled by the Java garbage collector.

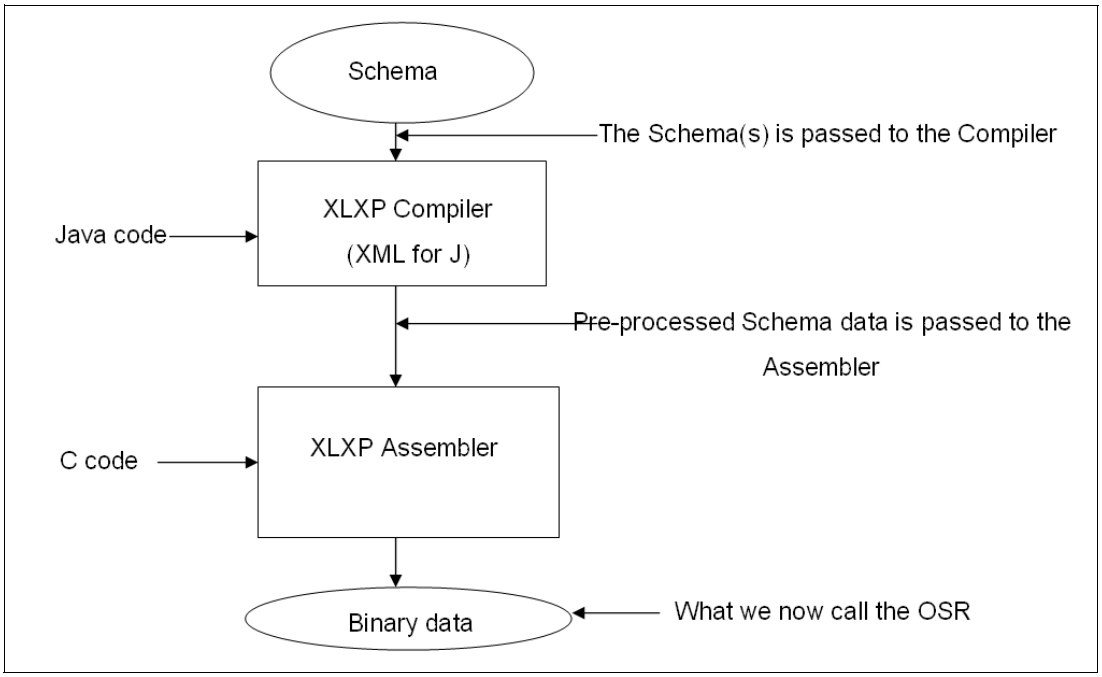


Figure 14-5 Overview of data flow

C APIs

The C APIs are packaged in gxlxosr1.dll. gxlxosr4.dll. At actual OSR generation time, XLXP functions in libGXLNVBAT.so are used. Most of the functions in the C APIs use Java methods in the background during processing. The Java APIs are packaged in gxljapi.jar and use objects from gxljosrgImpl.jar. The Java APIs call native C code in libGXLNVBAT.so at OSR generation time. The command line tool xsdosrg uses the C APIs in the background.

Important: Java 1.5 is the minimal level required for OSR processing.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gxlhosrg.h“
int main(unsigned int argc, char** argv)
{{
  void * oima_p = NULL;
  void * sys_svc_parm_p = NULL;
  void *schema_osr = NULL;
  long oima_l = GXLHXEC_MIN_OIMA_SIZE;
  int feature_flags = 0;
  int ctl_operation = 0;
  int rc = 0;  int rsn = 0;  int retVal = 0;
  char schema[] = "<?xml version='1.0' encoding='ibm-1047'?><xs:schema
xmlns:xs='http://www.w3.org/2001/XMLSchema'><xs:element name='root'
type='xs:anyType'/></xs:schema>";

  oima_p = (void *)malloc(oima_l * sizeof(void *));
  if(oima_p == NULL)
  {
     printf("[C CODE] Failed to allocate space for OIMA");
     return -1;
  }
  sys_svc_parm_p = (void *)malloc(10000);
  if(sys_svc_parm_p == NULL)
  {
     printf("[C CODE] Failed to allocate space for sys_svc_parm_p");
     return -1;
  }
  retVal = gxluInitOSRG(oima_p, oima_l, feature_flags, sys_svc_parm_p, &rc, &rsn);
  if(retVal != 0)
  {
    printf("[NATIVE CODE] gxluInitOSRG has failed!\n");
    return rsn;
  }
  retVal = gxluLoadSchema(oima_p, schema, &rc, &rsn);
  if(retVal != 0)
  {
    printf("[NATIVE CODE] gxluLoadSchema has failed!\n");
    return rsn;
  }
  retVal = gxluGenOSR(oima_p, &schema_osr, &rc, &rsn);
  if(retVal == 0)
  {
    printf("[NATIVE CODE] gxluGenOSR has failed!\n");
    return rsn;
  }
  retVal = gxluTermOSRG(oima_p, &rc, &rsn);
  if(retVal != 0)
  {
    printf("[NATIVE CODE] gxluTermOSRG has failed!\n");
    return rsn;
  }
  free(oima_p);
  free(sys_svc_parm_p);
  return 1;
}
```

Figure 14-6 C Sample of XLXP use

14.3 z/OS XML improved IPCS support

To make it easier to analyze z/OS XML System Services dumps, the XMLDATA subcommand is provided for use with the IPCS formatter. To use the subcommand, input the following under IPCS Option 6: COMMAND. Then, from the command line on the IPCS Subcommand Entry panel, issue XMLDATA with a parameter and option.

- XMLDATA address option

  The address parameter is the address of the z/OS XML parser's Parser Anchor Block (PAB); this is a required parameter. The address parameter accepts both 31- and 64-bit addresses. If you do not know the value for the address parameter, you can place a "0" in the address field, and XMLDATA will try to locate the value for you, for example: XMLDATA 0 TRACE. Although this method is not guaranteed to work, it is still an available option.

  The option parameter allows you to select what information you want to review within the provided dump. If nothing is provided for the option parameter, XMLDATA will use the default option BASIC.

  BASIC Displays to the screen widely used dump information. Such information includes the following: the PSW and any general information during the abend; the value of the registers; an API trace; a user input parameter list; feature flags, return code and reason codes; and the last 64 bytes of the input and output buffers.

  PARAM Displays the parameter list values for the GXL1PRS or GXL4PRS entry points.

14.4 zIIP and zAAP z/OS XML enablement

z/OS XML System Services provides the ability for parsing operations to be run on specialty processors, as follows:

- A zAAP (System z Application Assist Processor)

  The z/OS XML parser, when executing in TCB mode, is eligible to run on a zAAP, in environments in which one or more zAAPs are configured.

- A zIIP (IBM System z10 Integrated Information Processor).

  The z/OS XML parser, when executing in enclave SRB mode, is eligible to run on a zIIP processor, in environments where one or more zIIPs are configured.

Note: Ancillary z/OS XML System Services, such as the query service and the control service, as well as the StringID exit and memory management exits, are not eligible to run on specialty processors. Execution of z/OS XML System Services parsing operations on a specialty processor occurs transparently to the calling application.

Exploiters

The type of specialty engine used depends on the environment in which the caller makes use of the parser, either the z/OS XML parser or the XML Toolkit. DB2 V9 already makes use of z/OS XML for non-validating parses, starting with z/OS V1R7. It continues to exploit new features of z/OS XML as they are implemented. COBOL announced the use of z/OS XML with Version 4.1, in December of 2007.

14.5 z/OS XML enhanced coding support

The character attribute tables used during the non-validating parse process are created manually, making support for new encodings more difficult. With z/OS V1R10, the parse process is improved with an internal z/OS XML tooling to generate character attribute tables as support for new encodings. No explicit action is required to enable this.

The benefit obtained from this is a more timely and accurate support for native support of new encodings.

14.6 z/OS XML support for the XML Toolkit

This new support with z/OS V1R10 provides a mechanism to tell the application where to find the text in the source document that corresponds to particular XML structures in the parsed data stream. The support indicates which records in the parsed data stream are the result of default values from the DTD. The schema is a functional feature of the XML Toolkit.

Note: The purpose of a DTD (Document Type Definition) is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

As with the default value indicators, fully qualified names on end element events are a feature of the XML Toolkit, and the solution provided here allows the Toolkit to continue providing that when running with z/OS XML.

Note: Before the z/OS XML parser can perform a parse on an XML document, it must first establish a context in which it can operate. This is accomplished when the caller invokes the initialization routine and passes in a piece of memory where the z/OS XML parser establishes a parse instance memory area (PIMA). This is the area where the z/OS XML parser creates a base for the internal data structures it uses to complete the parse process. For the non-validating z/OS XML parser, the minimum size for the PIMA is 128 kilobytes. For the validating z/OS XML parser, the minimum size for the PIMA is 768 kilobytes.

This support is enabled through a new set of classes that mirror the structure and externals of the current XML Toolkit. The primary benefit is better performance for the SAX and DOM interfaces of the XML Toolkit when used with z/OS XML.

Source offsets are made available by the XML toolkit to the calling application. This functionality in particular is required by WebSphere Transformation Extender (WTX).

This solution provides the means for the Toolkit to continue to support this feature when running with z/OS XML.

Note: Validating parses are only performed against XML schema, and not DTDs. However, DTDs are tolerated in source XML documents, and processed for the purposes of entity definitions, and default values. Such definitions and values that are referenced in the root element will be substituted, and output to the parsed data stream.

Parsing XML documents

Everything that the z/OS XML parser needs to complete the parse of a document is kept in the PIMA, along with any associated memory extensions that the parser may allocate during the parse process. The caller also must provide input and output buffers on each call to the parse service (gxlpParse for C/C++ callers, GXL1PRS (GXL4PRS) for assembler callers). In the event that either the text in the input buffer is consumed or the parsed data stream fills the output buffer, the z/OS XML parser will return XRC_WARNING, along with a reason code indicating which buffer (possibly both) needs the caller's attention.

Restriction: With z/OS V1R10, The following restrictions apply when conducting a validating parse:

- When parsing in non-Unicode encodings, unrepresentable character entities are replaced with the "-" character prior to validation.
- There is a maximum of 64 KB non-wildcard attributes for a single element, and 64 KB elements in an All group.

14.7 z/OS XML support for the COBOL compiler

Currently z/OS XML does not support parsing of documents in some of the key EBCDIC code pages that are supported by the COBOL compiler. IBM has implemented new character classification tables that allow z/OS XML to support the required code pages. COBOL will not have to transcode documents into a different encoding prior to calling z/OS XML. This improves performance and reduces complexity in the COBOL compiler, since code page conversion is not required.

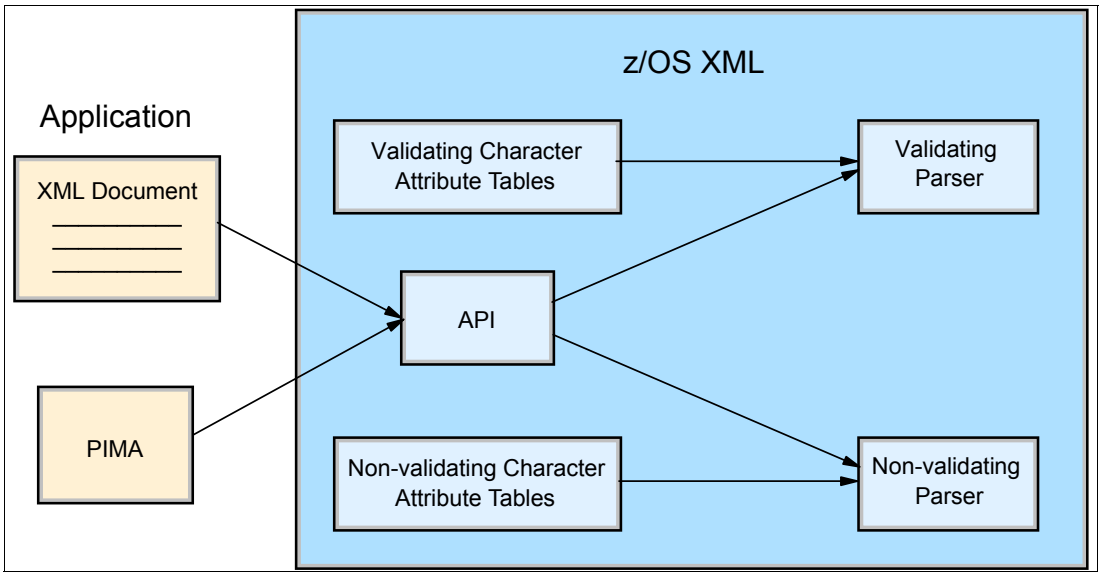


Figure 14-7 A view of z/OS XML

At initialization time, the user will select what encoding the parser will use by the CCSID parameter. With that information, z/OS XML will store pointers to the relevant character attribute tables in the PIMA. At parse time, z/OS XML will use these tables to parse the document in the proper encoding.

```
#include <gxlhxml.h>
#include <stdio.h>

int main (int argc, char **argv) {
  int rc, rsn; /* return and reason codes */
  char PIMA[GXLHXEC_MIN_PIMA_SIZE];
  long lPIMA = GXLHXEC_MIN_PIMA_SIZE;
  int feature_flags = 0,option=0; /* no features are enabled */
  GXLHXSV xsv;

  xsv.XSV_COUNT = 0; /* no exits */
  PIMA = malloc(lPIMA);
  if (PIMA == NULL) return 1;


gxlpInit((void*)PIMA,lPIMA,GXLHXEC_ENC_IBM_1140,feature_flags,xsv,NULL,&rc,&rsn
);

  if (rc != XRC_SUCCESS) {
    printf("Failed to select IBM-1140 encoding\n");
    return 1;
  }
  gxlpTerminate((void*)PIMA,&rc,&rsn);
  printf("IBM-1140 selected successfully.\n");
  return 0;
}
```

Figure 14-8 Sample C program using translation

15

RRS enhancements

Resource recovery is the protection of the resources. It consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources.

z/OS, when requested, can coordinate changes to one or more protected resources, which can be accessed through different resource managers and reside on different systems. z/OS ensures that all changes are made or no changes are made. Resources that z/OS can protect include:

- A hierarchical database
- A relational database
- A product-specific resource

This chapter describes the functional enhancements to RRS in z/OS V1R10. They are:

- RRS supports the MVS commands SETRRS ARCHIVELOGGING,DISABLE and SETRRS ARCHIVELOGGING,ENABLE to allow an installation to disable and enable RRS archive logging on a given system.
- Enhancement of the ATRSRV utility with a new Forget request and to discard an SDSRM's interest in a transaction.

15.1 RRS archive logging

IBM supplies, in SYS1.SAMPLIB, a cataloged procedure named ATRRRS that you can use to start RRS after system initialization. Your installation should copy SYS1.SAMPLIB(ATRRRS) to SYS1.PROCLIB(RRS). The membername RRS specified here can be replaced with any other membername, as long as it matches the subsystem name specified in the SYS1.PARMLIB(IEFSSNxx) used by the installation. If the names do not match, you may receive error messages when you start the subsystem.

Once the archive log stream is defined, RRS writes to the archive log for each completed UR. If archive logging is not desired, RRS must be stopped on all systems in the logging group simultaneously, in order to delete the archive log stream. This affects the subsystems and applications for which RRS is the transaction coordinator. The outages incurred by making this change are unacceptable.

With z/OS V1R10, RRS provides a means to allow you to disable or enable archive logging without impacting RRS services.

Define the archive logstream

To use archive logstreams in your system, you have to define the archive logstreams. Figure 15-1 shows an example of how an ATRRRS archive logstream is defined to the system.

```
//LUTZLOG JOB (MISY,TXX,T870270),KUEHNER,MSGCLASS=X,
//            MSGLEVEL=(1,1),
//            NOTIFY=&SYSUID,CLASS=D
/*JOBPARM ROOM=F21,S=SC70
//DEFINE   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DATA TYPE (LOGR) REPORT(YES)
   DEFINE LOGSTREAM NAME(ATR.SANDBOX.ARCHIVE)
          HLQ(LOGR)
          STRUCTNAME(RRS_ARCHIVE_1)
          HIGHOFFLOAD(80) LOWOFFLOAD(20)
          LS_DATACLAS(NO_LS_DATACLAS)
          LS_MGMTCLAS(NO_LS_MGMTCLAS)
          LS_STORCLAS(NO_LS_STORCLAS)
          LS_SIZE(700)
          STG_DUPLEX(YES)
          DUPLEXMODE(COND)
          STG_DATACLAS(NO_STG_DATACLAS)
          STG_MGMTCLAS(NO_STG_MGMTCLAS)
          STG_STORCLAS(NO_STG_STORCLAS)
          STG_SIZE(0)
```

Figure 15-1 Sample job to define the ATRRRS logstream

15.1.1 Controlling an RRS logstream

Because of the severe performance impact of running RRS with the archive log stream, customers want to run without it. This can be accomplished by deleting the archive log before starting any RRS image in the logging group or when RRS is operational.

With z/OS V1R10, a new command, SETRRS ARCHIVELOGGING, can be used to disable or enable RRS archive logging on a system:

```
SETRRS ARCHIVELOGGING,[DISABLE | ENABLE]
```

where:

DISABLE Indicates the system is to disable RRS archive logging for the subsequent transactions. RRS will stop writing the transaction completion records to the archive log and disconnect from the archive log stream.

ENABLE Indicates the system is to enable RRS archive logging for the subsequent transactions. RRS will connect to the archive log stream and start writing the transaction completion records to the archive log. The ENABLE request is done by asynchronous processing and might take as long as 15 seconds before the Archive Log Stream is connected and the enable message (ATR175I) is issued.

After successfully defining the logstream, you can activate the archive logstream using the SETRRS command, as shown in Figure 15-2. The successful activation message ATR175I is issued.

```
SETRRS ARCHIVELOGGING,ENABLE
IXC582I STRUCTURE RRS_ARCHIVE_1 ALLOCATED BY SIZE/RATIOS. 107
  PHYSICAL STRUCTURE VERSION: C2503C13 677613CE
  STRUCTURE TYPE:             LIST
  CFNAME:                     CF1
  ALLOCATION SIZE:      12288 K
  POLICY SIZE:          16384 K
  POLICY INITSIZE:      12288 K
  POLICY MINSIZE:           0 K
  IXLCONN STRSIZE:          0 K
  ENTRY COUNT:           4669
  ELEMENT COUNT:         9109
  ENTRY:ELEMENT RATIO:           1 :      2
ALLOCATION SIZE IS WITHIN CFRM POLICY DEFINITIONS
IXL014I IXLCONN REQUEST FOR STRUCTURE RRS_ARCHIVE_1 108
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0018
CONNECTOR NAME: IXGLOGR_SC70 CFNAME: CF1
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 109
STRUCTURE RRS_ARCHIVE_1, CONNECTOR NAME IXGLOGR_SC70
CFNAME     ALLOCATION STATUS/FAILURE REASON
 --------   --------------------------------
 CF1        STRUCTURE ALLOCATED AC001800
 CF2        PREFERRED CF ALREADY SELECTED AC001800
IEF196I IGD100I 611D ALLOCATED TO DDNAME SYS00116 DATACLAS (        )
IEF196I IEF237I 611D ALLOCATED TO SYS00117
IXG267I STAGING DATASET LOGR.ATR.SANDBOX.ARCHIVE.SC70 112
ALLOCATED WITH INCORRECT VSAM SHAREOPTIONS,
USE IS ACCEPTED BUT NOT RECOMMENDED.
ATR175I RRS ARCHIVE LOGGING HAS BEEN ENABLED.
```

Figure 15-2 Successful activation of the logstream

Disable archive logging

To disable archive logging, RRS stops writing new completion records to the archive log and disconnects from the logstream on that system. RRS does not delete the archive log stream.

To quiesce archive logging, RRS does not write any new completion records, but will complete any outstanding writes before the disable command is issued. A message is issued to indicate that archive logging has been disabled, as shown in Figure 15-3.

A disable setting indicates that RRS on that system will not connect to or use the log, but the log stream might still be present. Conversely, starting RRS with an enable setting and a deleted archive logstream will cause RRS to try and connect to the logstream, which will fail with this message:

```
ATR132I RRS LOGSTREAM CONNECT HAS FAILED FOR OPTIONAL LOGSTREAM
ATR.PLEX1.ARCHIVE. RC=00000008, RSN=0000080B
```

Once archive logging is disabled, the only way to resume it is to specifically enable archive logging again.

During RRS restarts, if archive logging is disabled, RRS does not try to connect to the logstream. RRS issues the message ATR174I to indicate that archive logging is disabled. If archive logging is enabled, RRS connects to the archive logstream and starts writing completion records for the subsequent transactions. Message ATR175I is issued to indicate that archive logging has been enabled. The enable and disable settings persist across RRS restarts and IPLs.

```
SETRRS ARCHIVELOGGING,DISABLE
ATR174I RRS ARCHIVE LOGGING HAS BEEN DISABLED.
IXC579I NORMAL DEALLOCATION FOR STRUCTURE RRS_ARCHIVE_1 IN 124
        COUPLING FACILITY  002094.IBM.02.00000002991E
                           PARTITION: 0F    CPCID: 00
HAS BEEN COMPLETED.
PHYSICAL STRUCTURE VERSION: C2503C13 677613CE
INFO116: 13572800 01 2800 00000012
TRACE THREAD: 00544617.
```

Figure 15-3 Successful deactivation of an archive logstream

Note: For the next RRS restart or IPL, the disable is remembered. The following messages are issued:

```
ATR221I RRS IS JOINING RRS GROUP PLEX1 ON SYSTEM SY1
ATR174I RRS ARCHIVE LOGGING HAS BEEN DISABLED.
ASA2011I RRS INITIALIZATION COMPLETE. COMPONENT ID=SCRRS
```

Display archive logging

To detect whether ATRRRS archive logging is active or not, use the DISPLAY LOGGER command, as shown in Figure 15-4 on page 339.

```
D LOGGER,LOGSTREAM.LSNAME=ATR.SANDBOX.ARCHIVE
 IXG601I   18.14.00  LOGGER DISPLAY 135
 INVENTORY INFORMATION BY LOGSTREAM
 LOGSTREAM                  STRUCTURE        #CONN  STATUS
 ---------                  ---------        ------ ------
 ATR.SANDBOX.ARCHIVE        RRS_ARCHIVE_1    000002 IN USE
   SYSNAME: SC65
     DUPLEXING: STAGING DATA SET
   SYSNAME: SC70
     DUPLEXING: STAGING DATA SET
   GROUP: PRODUCTION

 NUMBER OF LOGSTREAMS:  000001
```

Figure 15-4 D LOGGER command

15.1.2 Coexistence support

APAR OA23153 needs to be installed on any lower level systems in the sysplex. Without this APAR on the lower level system, RRS will not be able to start on that system if you are using the new ARCHIVELOGGING feature. In this case, RRS must be removed from the XCF group on the lower level system to allow the RRS restart. This scenario can be identified by the following messages:

```
ATR235I RRS FAILED TO JOIN THE RRS XCF GROUP.  RC = 00000008, RSN = 00000010
ASA2013I RRS INITIALIZATION FAILED. COMPONENT ID=SCRRS
```

Fallback support

RRS uses a single XCF group, called ATRRRS, to communicate between images in a sysplex. No special processing is required by an installation to enable RRS usage of XCF. You do not need to modify XCF transport classes.

Should the need arise where a z/OS V1R10 system needs to fall back to a lower release, a fallback toleration, APAR OA23153, should be installed to allow the lower level of RRS to start and preserve the archive logging preference from the SETRRS command. This can be done by the following two steps to remove RRS from the XCF Group:

1. When falling back to a lower level system from z/OS V1R10, you need to remove the member from the XCF group. Use the IXCM2DEL program to do this.

   In the job stream, specify a statement that indicates the system name that you want to delete. Upon successful completion of the JCL job, RRS will now be able to start on the lower level system. A sample job is:

   ```
   //IXCDELUT JOB
   //S1 EXEC PGM=IXCM2DEL,PARM='ATRRRS,mem01'
   //SYSPRINT DD SYSOUT=A
   ```

   Where `mem01` is the system name of the member to be deleted. To see the system names, use the following command:

   ```
   D XCF,GROUP,ATRRRS
   IXC332I 09.49.44 DISPLAY XCF GROUP
    GROUP ATRRRS:    SC63               SC64
                     SC65               SC70
   ```

Note: The IXCM2DEL sample job shipped by IBM can you found in SYS1.SAMPLIB.

2. Set up the proper access authorization to the RACF FACILITY class resource profile MVSADMIN.XCF.IXCM2DEL. If your installation uses the RACF component of SecureWay™ for z/OS, this can be done from an authorized userid using the following commands:

```
RDEFINE FACILITY MVSADMIN.XCF.IXCM2DEL UACC(ALTER)
SETROPTS RACLIST (FACILITY) REFRESH
```

Note: When completed, the Archive Logging Enable/Disable setting on the V1R10 system will be deleted.

15.2 Modified ATRSRV service

Prior to z/OS V1R10, the Server Distributed Syncpoint Resource Manager (SDSRM) fails. Then, SDSRM had to be restarted or else transactions waited indefinitely. Many times restarting the SDSRM may not be practical. So, how to move the transactions along without the SDSRM is resolved with z/OS V1R10 by enhancing the ATRSRV utility with a new FORGET request. By doing this, it will discard an SDSRM's interest in a transaction and thus SDSRM does not need to be restarted and transactions proceed to completion without the SDSRM.

ATRSRV macro service

The ATRSRV macro allows authorized callers to do the following:

- Remove a Resource Manager's interest in a UR.
- Resolve a UR state from In-doubt to In-Commit.
- Resolve a UR state from In-doubt to In-Backout.
- Remove a Resource Manager's identity.
- Unregister a Resource Manager's involvement with RRS.
- Perform Forget processing on a UR whose In-Doubt condition has been resolved.

Note: FORGET processing is new with z/OS V1R10.

15.2.1 FORGET request processing

RRS syncpoint processing requires a FORGET call to complete the transaction if the SDSRM fails. A SDSRM can fail either through the unregistering of the SDSRM or the cancelling or recycling of RRS. The transaction will wait for the SDSRM to restart. In Figure 15-5 on page 341 you see the modified ATRSRV service with the new FORGET option.

```
ATRSRV
One or more blanks must follow ATRSRV.
REQUEST=REMOVINT
,RMNAME=rmname
,URID=urid
REQUEST=COMMIT
,URID=urid
REQUEST=BACKOUT
,URID=urid
REQUEST=REMOVRM
,RMNAME=rmname
REQUEST=UNREGRM
,RMNAME=rmname
REQUEST=FORGET
,URID=urid
,RMNAME=rmname       rmname: RS-type address or address in register (2) - (12).
,URID=urid           urid: RS-type address or address in register (2) - (12).
                     Note: URID is optional with REQUEST=REMOVINT.
,GNAME=gname         gname: RS-type address or register (2) - (12).
,SYSNAME=sysname     sysname: RS-type address or register (2) - (12).
,RCTABLE=rctable     rctable: RS-type address or address in register (2)- (12).
,RCNUM=rcnum         rcnum: RS-type address or address in register (2) - (12).
,RETCODE=retcode     retcode: RS-type address or register (2) - (12).
,RSNCODE=rsncode     rsncode: RS-type address or register (2) - (12).L
```

Figure 15-5 Modified ATRSRV service

The FORGET request can be performed programmatically via new support in the ATRSRV service. The command syntax is shown and new return and reason codes are provided to indicate successful forget or inability to forget. URID is required for FORGET.

REQUEST=FORGET

Perform Forget processing for a UR (Unit of Recovery) whose In-Doubt condition was previously resolved to In-Commit or In-Backout. Provide the following:

- The Unit of Recovery Identifier (URID). In this case, the SDSRM interest of the UR will be forgotten. The URID specified must contain the SDSRM interest (top-level UR). Specifying a URID of a child or subordinate UR is not valid for this request and will result in a reason code of:

  ```
  ATRSRV_URID_NOT_FOUND (8).
  ```

Optionally, you may provide:

The GNAME and SYSNAME keywords. To affect units of recovery on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

16

SMP/E V3R5 enhancements

In z/OS V1R10, SMP/E is enhanced to help simplify the task of verifying required software fixes identified in Preventive Service Planning (PSP) buckets. PSP buckets identify required software fixes for new hardware devices, toleration and coexistence of new software releases, and enabling new functions.

More specifically, IBM plans to consolidate the lists of required fixes from PSP buckets and produce SMP/E-consumable metadata in the form of HOLDDATA to identify those fixes. SMP/E will use the new HOLDDATA to identify what fixes are missing in a current software environment. In addition, SMP/E will help to simplify the task of selecting and installing the required fixes identified by HOLDDATA.

This chapter describes the following changes to SMP/E:

- New FIXCAT type HOLDDATA to identify categories of fixes
- New REPORT MISSINGFIX command to identify missing fixes by category
- Long SOURCEIDs and extended wildcarding
- RECEIVE ORDER command enhancements
- Automatic FTP retry
- User- specified FTP parameters
- HOLDDATA report changes
- ZONEEDIT command extensions to add subentries if they do not exist
- UNIX file utility input for LMODs

16.1 Simplifying PSP buckets

When installing new hardware devices, new releases of software, or enabling selected new functions, z/OS platform users today are told to install fixes required for those devices, releases, or functions. Most such fixes are identified in PSP buckets (Preventive Service Planning) and also in product documentation. To utilize PSP buckets you first have to identify which buckets (upgrade/subset) to review, gather the list of fixes identified in the PSP buckets, acquire and install any fixes that are missing, and finally, monitor those PSP buckets for future additions. To help ensure that users do install all required fixes to support new hardware, new software, and new functions, the overall process is being simplified.

SMP/E V3R5 now creates metadata to associate fixes with one or more categories. Fix Categories now identify fixes for specific hardware devices, for new software releases, and to enable new hardware or software functions. This meta-data is delivered in the form of SMP/E HOLDDATA and therefore is obtained using existing acquisition procedures. Further, the verification and installation tasks are now integrated into well known SMP/E operations. The result is a simpler, less error prone, and more automated method of identifying and installing required fixes.

16.1.1 New type of HOLDDATA

A new form of HOLDDATA supports this new meta-data that will associate fixes (PTFs) with a particular categories of fixes. The new FIXCAT HOLD is applicable to FUNCTION SYSMODs (FMIDs) just like ERROR HOLDs for HIPER APARs. This associates an APAR to one or more categories of fixes, and this identifies the resolving PTF. The new version of HOLDDATA is shown in Figure 16-1.

```
++HOLD(HBB7730)                    /* Held FMID      */
  FMID(HBB7730)
  FIXCAT      /* Associates an APAR to a fix category */
  REASON(AA15968)                  /* The APAR       */
  CATEGORY(IBM.Device.2094)        /* A fix category */
  RESOLVER(UA27113)                /* The fixing PTF */ .
```

Figure 16-1 Modified HOLDDATA statement

++HOLD FIXCAT statement

The ++HOLD FIXCAT statement identifies those SYSMODs and their Fix Categories. When FIXCAT HOLDs are received, the Fix Category values are assigned as source IDs to the SYSMODs that resolve the APARs. During APPLY and ACCEPT command processing, you can use the assigned source IDs to select the SYSMODs that are associated with a particular Fix Category.

FIXCAT statements

FIXCAT statements are delivered in all existing IBM product and service offerings that currently deliver ERROR HOLDDATA, including SMP/E Internet service retrieval orders, ShopzSeries product and service orders, ServiceLink, TechSupport, the HOLDDATA web site, ServerPac, and CBPDO. The FIXCAT statement is in the same file (SMPHOLD) as the ERROR HOLDs. Prior releases of SMP/E are trained to silently ignore the new FIXCAT statement, but require a coexistence fix to do so.

Important: Prior SMP/E releases (V3R3 and V3R4) ignore FIXCAT statements when discovered in SMPHOLD data sets. The coexistence APAR IO07480 is required; PTF UO00701 is for V3R4 and PTF UO00700 for V3R3.

Fix Category values

The Fix Category values are used during RECEIVE command processing to create source IDs for the SYSMODs that will resolve the specified reason ID APAR. During APPLY and ACCEPT command processing, the Fix Category values are used to determine whether the HOLDDATA is applicable to the current command by comparing the values in the HOLDDATA to those in the active Fix Category interest list. The active Fix Category interest list is specified on the APPLY and ACCEPT command, or in the active OPTIONS entry.

16.1.2 Updated SMP/E functions using FIXCAT

The following SMP/E functions have been updated to include the new FIXCAT or HOLDFIXCAT subentry with SMP/E V3R5:

- ++HOLD MCS
- ++RELEASE MCS
- HOLDDATA entry (global zone)
- OPTIONS entry (global zone

  See “OPTIONS entry using FIXCAT” on page 349.
- SYSMOD entry (global zone)

In addition, a description of SMPHRPT has been added and the parameter COMPAT has been added to “EXEC statement”.

++HOLD MCS FIXCAT operand

An APAR provides a fix for the held SYSMOD and the fix is associated with one or more fix categories. It is optional whether the APAR will affect processing for the held SYSMOD, based on the APAR’s fix categories and the fix categories of interest specified by the user. If any one or more fix categories for the APAR match any of those of interest to the user, then the held SYSMOD will not be applied or accepted until the APAR is resolved. The APAR is resolved when a SYSMOD that matches the APAR name, or a SYSMOD that supersedes the APAR, is applied or accepted. FIXCAT holds can be read only from the SMPHOLD data set.

16.1.3 IBM fix categories

Figure 16-2 on page 346 and Figure 16-3 on page 346 show the IBM fix categories. The goal is for the Fix Category value to be as self explanatory as possible.

A Fix Category is a 1- to 64-character string. There can be one or more Fix Category value. A Fix Category value associates the reason ID APAR to a particular category of fixes. A Fix Category might be used to identify a group of APAR fixes required to support a particular hardware device, or to provide a particular software capability, similar to how a Preventive Service Planning Bucket (PSP-Bucket) identifies a group of APARs.

Software fix categories

Of particular note, the IBM.ProductInstall.RequiredService Fix Category identifies all recommended PTFs that should be installed when a new software release (FMID) is installed.

```
IBM.Function.DST2007
IBM.Function.HealthChecker
IBM.Function.SysplexDataSharing

IBM.Coexistence.DB2.V7
IBM.Coexistence.DB2.V8
IBM.Coexistence.DB2.V9
IBM.Coexistence.z/OS.V1R7
IBM.Coexistence.z/OS.V1R8
IBM.Coexistence.z/OS.V1R9
IBM.DrivingSysReq.z/OS.V1R9
IBM.TargetSysReq.z/OS.V1R8
IBM.TargetSysReq.z/OS.V1R9
IBM.TargetSysReq.WebSphere.V7

IBM.ProductInstall.RequiredService
```

Figure 16-2 IBM software fix categories

Hardware fix categories

```
IBM.Device.1750
IBM.Device.2064
IBM.Device.2066
IBM.Device.2084
IBM.Device.2086
IBM.Device.2094
IBM.Device.2094.zAAP
IBM.Device.2094.zIIP
IBM.Device.2096
IBM.Device.2096.zAAP
IBM.Device.2096.MIDAW
IBM.Device.2105
IBM.Device.2107
```

Figure 16-3 IBM hardware fix categories

16.1.4 The RECEIVE command

The RECEIVE command is updated to support reading FIXCAT HOLDs from the SMPHOLD file, and to store a HOLDDATA entry in the global zone for each HOLD, just like is done for ERROR HOLDs. The RECEIVE command processes FIXCAT (Fix Category) ++HOLDs along with ERROR ++HOLDs from the same file (SMPHOLD). Information is stored in the GLOBAL zone, just like ERROR HOLDs. Figure 16-4 shows an example of the modified ++HOLD statement.

```
++HOLD(HBB7730) FMID(HBB7730) FIXCAT REASON(AA15968)
  CATEGORY(IBM.Device.2094) RESOLVER(UA27113).
```

Figure 16-4 Modified ++HOLD statement

After receiving the HOLDATA you see the entry in the global zone as you can see in Figure 16-5. AA15968 is the Fix Category HOLD reasonid in effect for this SYSMOD.

```
HBB7730 TYPE = FUNCTION
        STATUS = REC
        DATE/TIME REC = 07.244 12:39:49
        SREL VER(1) = Z038
        HOLDERROR = AA12345
        HOLDFIXCAT = AA15968
        HOLDUSER = US56789.
```

Figure 16-5 Global zone entry for HBB7730

In addition to creating a HOLDDATA entry, the resolving PTF is updated to add a sourceid value that matches the Fix Category. The benefit is the sourceids subsequently allow the resolving PTFs to be easily applied simply by selecting PTFs with a specified sourceid. Incidentally, sourceid values can now be in mixed case and up to 64 characters long.

Like all other types of HOLDs, FIXCAT HOLDs also have a corresponding form of the RELEASE statement to allow FIXCAT HOLDs to be deleted from the global zone. In Figure 16-6 you see the appropriate example.

```
++RELEASE(HBB7730) FMID(HBB7730) FIXCAT REASON(AA15968).
```

Figure 16-6 New ++RELEASE statement

FIXCAT HOLDs and sourceids longer than 8 characters in SYSMOD entries in the global zone render that global zone incompatible with prior SMP/E releases. Hence, the UPGRADE command must be used to indicate that making incompatible changes is acceptable. If the UPGRADE command is not run for SMP/E V3R5, then FIXCAT HOLDs will not be received into the global zone.

Important: Therefore, FIXCAT HOLDDATA is received only if the UPGRADE command for SMP/E V3R5 has been run.

16.1.5 APPLY and ACCEPT commands

The APPLY and ACCEPT commands have been updated to allow mixed-case sourceids up to 64-characters long on the SOURCEID and EXSRCID operands, as well as to support extended wildcards within the sourceid values. The “*” global character indicates zero or more characters can occupy that position, and the “%” placeholder indicates any single character can occupy that position.

FIXCAT operand

The new FIXCAT operand tells APPLY and ACCEPT which fix categories are of interest during this particular command. This list determines which fix category APARs must be resolved for the SYSMODs being accepted. A fix category APAR provides a fix for a held SYSMOD and the APAR is associated with one or more fix categories. Fix Category APARs are identified by FIXCAT HOLD entries. If a fix category specified on a FIXCAT HOLD for a SYSMOD being accepted matches any of those specified on the FIXCAT operand of the command, then the SYSMOD is held for the APAR reason ID from the FIXCAT HOLD and will not be accepted until the APAR is resolved. If a Fix Category specified on a FIXCAT HOLD for a SYSMOD being accepted does not match any of those specified on the FIXCAT

operand of the command, or if the list of fix categories is null, the SYSMOD is not held for the APAR reason ID from the FIXCAT HOLD.

Finally, the BYPASS operand has been extended to allow FIXCAT HOLDs to be bypassed, just like ERROR, SYSTEM, and USER HOLDs.

```
APPLY SOURCEID(IBM.Device.2O%4*
               IBM.Function.HealthChecker
               RSU*)
      EXSRCID(RSUO7%%
              IBM.Device.*.zAAP)
      FIXCAT(IBM.Device.*.zIIP
             IBM.ProductInstall.RequiredService)
      BYPASS(HOLDFIXCAT(AA15968)) CHECK.
```

Figure 16-7 Modified APPLY statement

BYPASS(HOLDSYS) message severity changes

During APPLY and ACCEPT command processing, SMP/E writes messages to identify SYSMOD HOLD conditions that have been bypassed. In prior SMP/E releases, these messages had a severity of Warning and resulted in a minimum overall return code of 4 for the command. With SMP/E V3R5, informational messages are written for bypassed SYSTEM HOLD conditions instead of warning messages. This change to use Informational messages results in a minimum overall return code for the command of 0 instead of 4 when SYSTEM HOLDs are bypassed.

REPORT MISSINGFIX command

The new REPORT MISSINGFIX command (Figure 16-8) identifies fixes associated with particular fix categories that have not yet been installed, and whether any SYSMODs are available to satisfy those missing fixes. More specifically, the REPORT MISSINGFIX command reports on FMIDs that have been applied or accepted for which subsequent FIXCAT HOLDs have been received and whose reason IDs (APARs) have not yet been resolved. The FIXCAT HOLDs analyzed for the report processing are determined by the fix category values you specify on the command. The command only reports on the fix categories that match the interest list you specify. The command can also produce a RECEIVE ORDER command that can be used to order any missing PTFs that will resolve the identified FIXCAT HOLDs, and an APPLY or ACCEPT command to install the fixing PTFs.

```
REPORT MISSINGFIX ZONES(ZOS17T)
       FIXCAT(IBM.Device.2094.*
              IBM.Coexistence.z/OS.V1R8
              IBM.Function.HealthChecker)
       FORFMID(HBB7730 HRM7730).
```

Figure 16-8 New REPORT options

Report enhancement

Figure 16-9 shows an example of the report produced by the REPORT MISSINGFIX command. Notice how the missing APARs that were found are arranged by the Fix Category they are associated with.

```
MISSING FIXCAT SYSMOD REPORT FOR ZONE ZOS17T

                          HOLD     MISSING  HELD      ____RESOLVING SYSMOD_____
FIX CATEGORY      FMID    CLASS    APAR     SYSMOD    NAME     STATUS  RECEIVED
---------------   -------  -------  -------  -------  -------  ------  --------
IBM.Device.2094
                  HBB7730  SERVCAT  AA13644  HBB7730  UA27033  GOOD    NO
                                    AA14941  HBB7730  UA26443  GOOD    NO
                                    AA15968  HBB7730  UA27113  GOOD    YES
                                    AA16005  HBB7730  UA26745  GOOD    NO
                                    AA16529  HBB7730  UA26741  HELD    NO
                                    AA17268  HBB7730  UA27861  GOOD    NO
                  HRM7730  SERVCAT  AA16458  HRM7730  UA28236  GOOD    NO

IBM.Device.2094.zIIP
                  HBB7730  SERVCAT  AA15968  HBB7730  UA27113  GOOD    NO
                                    AA16005  HBB7730  UA26745  GOOD    NO
                  HRM7730  SERVCAT  AA16458  HRM7730  UA28236  GOOD    NO
                                    AA18772  HRM7730  ***NONE

IBM.Function.HealthChecker
                  HBB7730  SERVCAT  AA15593  HBB7730  UA28084  GOOD    NO
                                    AA16687  HBB7730  UA27678  GOOD    NO
                  HRF7730  SERVCAT  AA15290  HRF7730  UA26196  GOOD    NO
                                    AA17429  HRF7730  UA28412  GOOD    NO
```

Figure 16-9 Sample REPORT MISSINGFIX command output

16.1.6 Specifying fix categories

A list of interesting fix categories can be specified on an APPLY, ACCEPT, and REPORT command. Such a list is used solely for the processing of that specific command. For the APPLY and ACCEPT commands the default is to ignore any FIXCAT HOLDs, unless fix categories of interest are specified. Not only can you specify a list of interesting fix categories on the command, but you can also define a persistent list of interesting Fix Categories. The list is stored in the FIXCAT subentry of an OPTIONS entry. Using such a list allows you to build a list of interesting Fix Categories once, use it for SMP/E processing later, and build upon that list over time. For the APPLY, ACCEPT, and REPORT MISSINGFIX commands, if no list of fix categories is specified on the command (using the FIXCAT operand), then SMP/E will look for a list in the active OPTIONS entry.

OPTIONS entry using FIXCAT

The typical methods for updating an OPTIONS entry is to use the UCLIN command or the SMP/E Administration dialog. In Figure 16-10 on page 350 you see an example using the UCLIN command to add a FIXCAT subentry to an OPTIONS entry. This list of interesting fix categories is used for APPLY, ACCEPT, and REPORT MISSINGFIX command processing if a list of fix categories is not specified on those commands.

```
SET BDY(GLOBAL).
UCLIN.
  ADD OPTIONS(GLOBOPT) FIXCAT(
    IBM.Device.2O%4*
    IBM.Device.*.zIIP
    IBM.Coexistence.DB2.*
    IBM.Coexistence.z/OS.V1R9
    IBM.ProductInstall.RequiredService
    IBM.Function.HealthChecker).
ENDUCL.
```

Figure 16-10 Update of zone OPTIONS

FIXCAT OPTIONS ENTRY panel using the Administration dialog

Figure 16-11 shows a typical Administration dialog panel for directly updating the Fix Category Interest list of an OPTIONS entry (the FIXCAT subentry). The FIXCAT panel allows you to define a list of fix categories whose HOLDDATA is to be considered during APPLY, ACCEPT and REPORT MISSINGFIX command processing.

Explore Fix Categories - Specify YES to enter the Fix Category Explorer. You may view and select from a list of all Fix Category values from all FIXCAT HOLDs. In addition, new Fix Category values will be identified.

Fix Category - Specify one or more valid Fix Categories. The asterisk (*) and percent (%) characters may be used as placeholders in the Fix Category values. A single asterisk indicates zero or more characters can occupy that position. A single percent sign indicates any one single character can occupy that position.

```
                  FIXCAT OPTIONS ENTRY - ALLOPTS            Row 1 to 10 of 10
===> ____________________________________________________________ SCROLL ===> CSR

Enter the Fix Categories whose HOLDDATA is to be considered during processing
for the APPLY, ACCEPT and REPORT MISSINGFIX commands.  When the list is
complete, enter END.

Explore Fix Categories? ==> NO_ (YES or NO)

      Fix Category
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
''''  ____________________________________________________________
******************************* Bottom of data ********************************
```

Figure 16-11 New FIXCAT OPTIONS ENTRY panel

16.1.7 Fix Category Explorer

In addition to using UCLIN or the Administration dialog to manipulate the Fix Category interest in an OPTIONS entry, as shown in Figure 16-10 on page 350 and Figure 16-11 on page 350, you can also use the new Fix Category Explorer. The Fix Category Explorer exploits the hierarchical name structure of Fix Categories to provide an easy-to-use view of Fix Categories.

The Explorer allows you to view all Fix Categories defined by FIXCAT HOLDs in the global zone. From this view you can build a Fix Category interest list from scratch or update an existing list. The Fix Category Explorer panel shown in Figure 16-12 may be entered through the Administration dialog when manipulating an OPTIONS entry, in the command generation dialog for the APPLY, ACCEPT, and REPORT MISSINGFIX commands, and the SYSMOD Management dialog for the APPLY and ACCEPT commands.

```
--------------------------- Fix Category Explorer --------- Row 1 to 9 of 9
===>                                                         SCROLL ===> CSR

Commands: FIND -Find a string, E -Expand all, C -Collapse all, U -Unselect all

Actions:  E -Expand, C -Collapse, S -Select, U -Unselect, V -View patterns

No Fix Categories are new since 2007/09/27.

   Fix Categories                                                New Selected
   ------------------------------------------------------------- --- --------
  -IBM.*
   +IBM.Coexistence.*
   +IBM.Device.*
   +IBM.DrivingSysReq.*
    IBM.DST2007
    IBM.Function.HealthChecker
    IBM.ProductInstall.RequiredService
    IBM.SysplexDataSharing
   +IBM.TargetSysReq.*
****************************** Bottom of data ********************************
```

Figure 16-12 Fix Category Explorer

16.2 Installing new service for hardware

When you install or activate a new hardware device, it is important to have installed all the PTFs that are required and provide support for that device. Today you use PSP buckets to identify the list of PTFs that must be installed. Once you identify the appropriate upgrade and subset for the particular device you can manually compare the list of PTFs or use the EPSPT tool. With SMP/E V3R5, you first must obtain the latest HOLDDATA, using the RECEIVE ORDER command is a great method, and then use the REPORT MISSINGFIX command. On the REPORT MISSINGFIX command (Figure 16-13 on page 352) you specify the Fix Category that matches the subject device. You can use wildcards in the Fix Category specification in order to identify multiple devices, or even to identify multiple functions for a particular device.

```
SET BDY(GLOBAL).
  RECEIVE ORDER(CONTENT(HOLDDATA) ... ).
  REPORT MISSINGFIX
         ZONES(ZOS17T,ZOS18T,ZOS19T)
         FIXCAT(IBM.Device.20%4*).
```

Figure 16-13 REPORT MISSINGFIX report

The resulting report (Figure 16-14) identifies any missing APARs and their fixing PTFs associated with the interesting device. The report entries are arranged by their associated Fix Category values. The command optionally produces the appropriate APPLY or ACCEPT command that you can use to install the identified fixing PTFs.

```
MISSING FIXCAT SYSMOD REPORT FOR ZONE ZOS17T

                          HOLD     MISSING  HELD     ____RESOLVING SYSMOD_____
FIX CATEGORY     FMID     CLASS    APAR     SYSMOD   NAME     STATUS  RECEIVED
---------------  -------  -------  -------  -------  -------  ------  --------
IBM.Device.2094
                 HBB7730  SERVCAT  AA13644  HBB7730  UA27033  GOOD    NO
                                   AA14941  HBB7730  UA26443  GOOD    NO
                                   AA15968  HBB7730  UA27113  GOOD    YES
                                   AA16005  HBB7730  UA26745  GOOD    NO
                                   AA16529  HBB7730  UA26741  HELD    NO
                                   AA17268  HBB7730  UA27861  GOOD    NO
                 HRM7730  SERVCAT  AA16458  HRM7730  UA28236  GOOD    NO

IBM.Device.2094.zIIP
                 HBB7730  SERVCAT  AA15968  HBB7730  UA27113  GOOD    NO
                                   AA16005  HBB7730  UA26745  GOOD    NO
                 HRM7730  SERVCAT  AA16458  HRM7730  UA28236  GOOD    NO
                                   AA18772  HRM7730  ***NONE
```

Figure 16-14 Report MISSING PTFs

APPLY the PTFs

Instead of using the REPORT MISSINGFIX command to identify the missing fixes, you can simply and directly APPLY PTFs that are required or provide support for the desired device (Figure 16-15 on page 353). First obtain the very latest HOLDDATA and all PTFs that are applicable to your SMP/E environment. Remember, when the FIXCAT HOLDs are received, the fixing/resolving PTFs are assigned a sourceid to match the Fix Category from any FIXCAT HOLDs. After the HOLDDATA is received, use the SOURCEID operand to specify the sourceid that matches the Fix Category, thus selecting the appropriate PTFs. Once again you can use wildcards to specify multiple devices or generations of devices as well as various functions for those devices (2084, 2094, zAAP, zIIP, and MIDAW.

Note: Not every PTF received into the global must be applied. That is, do not be afraid to order and receive all PTFs that are applicable to your SMP/E environment. The presence of a PTF in the global zone does not mean it will be applied. It is the selection criteria you specify on the APPLY and ACCEPT commands that determines which PTFs will in fact be installed.

```
SET BDY(GLOBAL).
  RECEIVE ORDER(CONTENT(ALL) ... ).
SET BDY(ZOS18T).
  APPLY CHECK GROUPEXTEND BYPASS(HOLDSYS)
        SOURCEID(IBM.Device.20%4*).
SET BDY(ZOS19T).
  APPLY CHECK GROUPEXTEND BYPASS(HOLDSYS)
        SOURCEID(IBM.Device.20%4*).
```

Figure 16-15 Sample of APPLY

Install new software

Similar to when you install or activate a new hardware device, when a new software release (FMID) is installed it is important to also install all the recommended PTFs for that FMID as identified in the appropriate PSP bucket. Installing the recommended PTFs protects you from rediscovering known and fixed problems. Today once you identify the appropriate upgrade and subset for that software release you can manually compare the list of PTFs or use the EPSPT tool. With SMP/E V3R5 you first must obtain the latest HOLDDATA using the RECEIVE ORDER command and then you can simply and directly APPLY the recommended PTFs. You don't even need to know the PSP bucket upgrade and subset. Just specify a Fix Category interest list with the value IBM.ProductInstall.RequiredService. This single Fix Category is used to identify the recommended PTFs for all software releases of all products. If you specify this value on the FIXCAT operand of APPLY (Figure 16-16), it tells SMP/E that all FIXCAT HOLDs for the FMID you are applying must be resolved. Further, the GROUPEXTEND operand tells SMP/E to automatically find PTFs that resolve the FIXCAT HOLDs.

Note: If the FIXCAT operand is not specified on the APPLY, and if the active OPTIONS entry does not contain a Fix Category Interest list, then FIXCAT HOLDs have no effect on the APPLY command.

```
SET BDY(ZOS18T).
  APPLY CHECK GROUPEXTEND BYPASS(HOLDSYS)
    SELECT(HDB8810) FORFMID(HDB8810)
    FIXCAT(IBM.ProductInstall.RequiredService).
```

Figure 16-16 APPLY FIXCAT commands

Updated APPLY HOLDDATA reports

In SMP/E V3R5 the HOLDDATA reports have been updated to more completely suppress HOLDs for reasonids specified in the SUPPHOLD subentry of the active OPTIONS entry. Although a single report entry is produced for each suppressed reasonid for each FMID that has one or more SYSMODs affected by a HOLD with that reasonid, the number of report entries is greatly reduced. In addition, no HOLDs for SYSMODs that failed APPLY or ACCEPT processing will be reported in the Bypassed HOLD Reason report (Figure 16-17 on page 354). This report will only contain the HOLDs that have been bypassed for SYSMODs that have been applied or accepted and thus must be reviewed and acted upon. To this end the SYSMOD STATUS column has been removed. The result is less clutter and it is easier to focus on the HOLDs and reasonids that must be reviewed and acted upon.

```
BYPASSED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

TYPE    REASON ID  FMID     SYSMOD   ++HOLD DATA
------  ---------  -------  -------  ------------------------------------------------------------
SYSTEM  IPL        HBB7740           * ONE OR MORE IPL     HOLDS ARE SUPPRESSED FOR HBB7740

                   HJE7740           * ONE OR MORE IPL     HOLDS ARE SUPPRESSED FOR HJE7740
```

Figure 16-17 HOLD REASON REPORT

SMP/E report output

Currently all SMP/E report output is written to the SMPRPT ddname. Since the HOLDDATA reports can be quite lengthy, they can clutter and obscure the other important information written to SMPRPT, for example the Causer SYSMOD Summary report. To simplify finding specific information at the conclusion of an SMP/E operation, the HOLDDATA reports may be written to a different but optional ddname, SMPHRPT. If the SMPHRPT ddname is allocated, then the HOLDDATA reports for the APPLY, ACCEPT, and RECEIVE commands are written to it. If the SMPHRPT ddname is not allocated, then HOLDDATA reports are written to SMPRPT as always. If you want to ignore the HOLDDATA reports altogether, you can define an SMPHRPT DD statement as a dummy.

16.2.1 Bypassed SYSTEM HOLD messages

A Warning message issued by SMP/E during an SMP/E operation implies you must investigate the condition described by the message and take appropriate action, where “do nothing” may be an appropriate action. This inherently means that if a Warning message is issued you cannot easily determine the success or failure of an APPLY or ACCEPT operation without investigating that message. If in fact the Warning message is for a condition you have already instructed SMP/E to ignore, then it makes it all the more frustrating to investigate the warning only to find that it is the very condition you want ignored. During APPLY and ACCEPT command processing, unresolved SYSTEM HOLDs that are bypassed are one example of such a condition.

```
APPLY S(UA32547 UA33943) CHECK BYPASS(HOLDSYS).

GIM42001W    THE FOLLOWING CONDITIONS FOR SYSMOD UA32547 WERE NOT SATISFIED, BUT
             WERE IGNORED BECAUSE THE BYPASS OPERAND WAS SPECIFIED. PROCESSING
             CONTINUES.
GIM35966I    SYSTEM HOLD DOC ORIGINATED BY SYSMOD UA32547 WAS BYPASSED.

GIM42001W    THE FOLLOWING CONDITIONS FOR SYSMOD UA33943 WERE NOT SATISFIED, BUT
             WERE IGNORED BECAUSE THE BYPASS OPERAND WAS SPECIFIED. PROCESSING
             CONTINUES.
GIM35966I    SYSTEM HOLD DOC ORIGINATED BY SYSMOD UA33943 WAS BYPASSED.


GIM20501I    APPLY PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 04.
```

Figure 16-18 New BYPASS return code

Replace Warning with Informational

A message to identify a condition you have already instructed SMP/E to ignore should not produce a warning message at all. Rather, such a condition should produce an informational message. Therefore, messages to identify bypassed SYSMOD HOLDs will now be informational, with an associated return code of 0 such as in Figure 16-18. The result of this

change is that it should be more common for an APPLY or ACCEPT operation to end with a return code of 0. If an SMP/E operation ends with RC=0, it is obvious the operation was successful.

Since changing the severity of the message to identify bypassed SYSTEM HOLDs is a change in the basic behavior of SMP/E, there now exists a “switch” to allow SMP/E to operate using its previous behavior. The switch is in the form of a new execution parameter for PGM=GIMSMP, COMPAT. The COMPAT parameter (Figure 16-19) controls incompatible behaviors of SMP/E. Specifically, COMPAT(WARNBYPASS) indicates warning messages with an associated RC=4 will be issued for bypassed SYSTEM HOLD conditions. This matches the behavior of SMP/E releases prior to SMP/E V3R5.
COMPAT(NOWARNBYPASS) indicates informational messages with an associated RC=0 will be issued for bypassed SYSTEM HOLD conditions. This is the default behavior for SMP/E V3R5.

```
//SMP      EXEC PGM=GIMSMP,PARM='COMPAT(WARNBYPASS)'
```

Figure 16-19 New COMPAT parameter

16.2.2 HOLDDATA report changes

The following changes have been made to HOLDDATA reports:

- A new report destination, SMPHRPT, has been introduced to SMP/E. If the SMPHRPT ddname is allocated (either by a DD statement or a DDDEF entry), the HOLDDATA reports produced by the RECEIVE, APPLY, and ACCEPT commands are written to this ddname while other reports are written to the SMPRPT ddname. If the SMPHRPT ddname is not allocated, all reports are written to SMPRPT.
- The Bypassed HOLD Reason Report for the APPLY and ACCEPT commands includes only bypassed HOLDDATA for SYSMODs that are applied or accepted successfully. Bypassed HOLDDATA for SYSMODs that fail command processing will no longer appear in the report.
- In the APPLY and ACCEPT command HOLDDATA reports, suppressed reason IDs are consolidated and reported only once per reason ID per FMID. Formerly, reason IDs were reported once per SYSMOD. Reason IDs to be suppressed are identified by the SUPPHOLD subentry of the active OPTIONS entry.

16.3 Enhanced utility input

SMP/E V3R5 allows the specification of a standard UNIX filename as utility input on an INCLUDE statement found in a link-edit step of a JCLIN input stream. This increases the allowable length of the utility input name from 8 characters to 1023 characters. It also allows characters other than uppercase alphabetic (A-Z), numeric (0-9), and national (@,#,$). The utility input name can now contain any nonblank character X’41’ through X’FE’.

Long SOURCEID support

SMP/E V3R5 allows SOURCEID values that can be up to 64 characters in length and that can contain any nonblank character (X’41’ through X’FE’) except for the following characters:

- A single quotation mark (’)
- An asterisk (*)
- A percent (%)

- A comma (,)
- A left parenthesis (()
- A right parenthesis ())

16.4 ZONEMERGE command

SMP/E V3R5 has modified the ZONEMERGE command to automatically check for incompatible changes between the originating and destination zones before allowing the merge. This requires that a zone definition entry be present in the destination zone. If a zone entry is not present in the destination zone, or if it is present, but incompatible changes exist between the originating zone and the destination zone, an error message is issued. Additionally, during CONTENT processing, the SREL subentry of the zone definition entry is merged.

16.5 HTTPS and FTP enhancements

SMP/E uses HTTPS and FTP to communicate with servers for the RECEIVE ORDER and RECEIVE FROMNETWORK commands, or the GIMGTPKG service routine. As a result, SMP/E relies on an available and operational network, and operations might fail when network outages occur. SMP/E has added retry capabilities for HTTPS and FTP operations that fail because of an apparent network outage. This helps prevent SMP/E operations from failing because of short lived network outages.

SMP/E allows any value that can be specified as a parameter for the FTP client to be specified during the RECEIVE ORDER command, RECEIVE FROMNETWORK command and GIMGTPKG service processing. You can specify an FTP.DATA file override (-f) parameter. You can specify the FTP client parameters in the FTPOPTIONS tag in the CLIENT or SMPCLNT data set.

16.6 ZONEEDIT enhancement

The ZONEEDIT command now allows subentries to be added as well as changed. You can add the UNIT, VOLUME, and WAITFORDSN subentries of the DDDEF entry, and the PRINT subentry of the UTILITY entry using the ZONEEDIT command.

16.7 RECEIVE ORDER processing enhancements

When an order results in an empty package, SMP/E resubmits the order to obtain the most recent HOLDDATA.

When an order requests CONTENT PTFS and some of the requested PTFS cannot be found by the server, SMP/E resubmits the order to obtain those requested PTFs that can be found by the server.

When an order requests CONTENT APARS and fixing PTFs cannot be found for some of the requested APARs, SMP/E resubmits the order to obtain the fixing PTFs that can be found.

16.8 Coexistence considerations

Many of the changes are only available with the new SMP/E V3R5.

Utility input

SMP/E releases before SMP/E V3R5 cannot process a JCLIN input stream containing an INCLUDE statement that has a utility input value that is longer than 8 characters or contains a character other than uppercase alphabetic (A-Z), numeric (0-9) and national (@,#,$). An error message is issued if an unsupported utility input value is encountered.

Additionally, unless the required PTF is installed, SMP/E releases before SMP/E V3R5 cannot process an LMOD entry containing an unsupported UTIN subentry. If installed, the PTF updates the SMP/E Query Dialogs and the GIMAPI to retrieve and display information about the UTIN subentries created with SMP/E V5R5 or higher. For the LIST and UNLOAD commands, if the zone contains an unsupported UTIN value, the PTF causes SMP/E to issue a warning message. For all other commands, if the zone contains an unsupported UTIN value, the PTF causes SMP/E to issue an error message.

SOURCEID characters

SMP/E releases before V3R5 cannot process the MCS input that is developed specifically for V3R5, nor can they process data in the SMPCSI data set. In SMP/E V3R4, if your SOURCEID value meets either of the following conditions, a message is shown to indicate that your operand contains a value that cannot be processed by the current release of SMP/E:

- Your SOURCEID value is 9 to 64 characters in length.
- Your SOURCEID value contains a character other than uppercase alphabetic (A-Z), numeric (0-9), or national (@,#,$).

The ZONEMERGE command

Unless the required PTF is installed, SMP/E releases before SMP/E V3R5 will not merge the SREL subentry of the zone definition entry, nor will they automatically check for incompatible changes between the originating and destination zones. If installed, the PTF causes SMP/E to merge the SREL data during CONTENT processing and to check for incompatible changes between the two zones.

16.9 Migration considerations

An application program that uses the GIMAPI to query the UTIN subentry of an LMOD entry might need to be updated to allow for the longer length for these UTIN values. Additionally, because the comma is used as the separator between the ddname and the filename in the UTIN value and because it is also a valid character in the UTIN filename, the application program might have to change the manner in which it extracts the ddname from the value.

16.9.1 The ZONEMERGE command

If the zone entry does not exist in the destination zone, it must be created using either UCLIN or the SMP/E dialogs. If the zone entry is present, but incompatible changes exist between the two zones, you must run the UPGRADE command against the destination zone. Use the level of SMP/E that is equal to or higher than the UPGLEVEL subentry found in the zone entry of the originating zone. That is, if the UPGLEVEL subentry of the zone entry in the

originating zone is 34.00, the UPGRADE command must be run against the destination zone using SMP/E V3R4 or higher.

The UPGRADE command

The UPGRADE command allows you to make the trade-off between fully exploiting new SMP/E functions and preserving compatibility with prior SMP/E releases. The UPGRADE command authorizes SMP/E to exploit the new functions that make incompatible changes to SMP/E data sets, and you must use the UPGRADE command before exploiting such functions.

New SMP/E functions must sometimes make changes to SMP/E data sets that cannot be properly processed by prior SMP/E releases. For example, a new type of element requires a new entry type in SMPCSI data sets and these new entry types are typically not understood or processed correctly by SMP/E levels that have not been specifically updated to do so. Rather than arbitrarily making such incompatible changes on its own, SMP/E will continue to use processing that is compatible with prior releases until you use the UPGRADE command.

16.9.2 The APPLY/ACCEPT commands

To change APPLY and ACCEPT command processing to write warning messages for bypassed SYSTEM HOLD conditions, as was done in prior SMP/E releases, use the new COMPAT(WARNBYPASS) execution parameter for program GIMSMP. For example:

```
//SMPSTEP EXEC PGM=GIMSMP,PARM='COMPAT(WARNBYPASS)'
```

17

Language Environment enhancements

This chapter describes the enhancements to the Language Environment. The following functions are new or modified:

- CEEROPT in batch
- PARMLIB syntax checker
- Parmlib Health Check
- Multithreaded C/C++ I/O performance
- VSCR for Language Environment
- Heap Pools Multi-pools
- Support for 31-bit HEAPPOOLS in AMODE64
- IEEE Decimal Floating Point support
- Savstack fields
- Language Environment Compare and Trap
- SDUMP Serviceability
- Language Environment for Password Phrase

17.1 New CEEROPT options

Starting with OS/390 V2R8, an additional options module, CEEROPT, was introduced. This module is intended to provide a means to specify LE options tailored to specific CICS regions. It is coded using the same sample source called CEECOPT, but the CSECT name must be changed to CEEROPT. Also, when you code a CEEROPT module you link-edit it into a standalone load module named CEEROPT and store it in a DFHRPL library. You must also create a CICS program definition for CEEROPT. This then sets the LE default options for a specific CICS region. Like the CEEUOPT, you can have an arbitrary number of CEEROPT source members with different names in a single source library, as long as their CSECT and resulting load modules are named CEEROPT.

Starting with z/OS V1R7, support was introduced to help users disengage from using usermods. However, this was not realized completely since there are a few inhibitors that the users are facing. There are a certain set of run-time options, such as RTEREUS and CBLOPTS and others that can be specified only at certain places for those run-time options to get processed. Some customers have a need for different specifications of these run-time options in different batch environments in an LPAR. Since the existing support does not have the capability to change these run-time options after a certain phase of the initialization process, customers have created a workaround to address this issue. They have created different sets of SCEERUN options with modifications to those run-time options and using a JCL STEPLIB statement to the appropriate SCEERUN.

17.1.1 Region-specific run-time options

An installation might need to set different run-time options on a region basis rather than an installation-wide default, such as an IMS region. For example:

- One CICS region (Region A) can be designated to run only AMODE 31 programs, while another region (Region B) runs both AMODE 24 and AMODE 31 programs.
- This requires Region B to have the ALL31(OFF) option setting while Region A can perform better with the ALL31(ON) option setting.

A run-time options load module, CEEROPT, is available just for this purpose. CEEROPT is a stand-alone load module that can be used to set region-specific Language Environment run-time options that are different from the installation or system level defaults.

CEEROPT and CELQROPT

With z/OS V1R10, LE enables the use of CEEROPT in a non-CICS and non-LRR (library retention reuse) environment. Also provided is a 64-bit version of CEEROPT, called CELQROPT. This support is enabled by providing two new options, shown in Figure 17-1, in the CEEPRMxx parmlib member. Per default the CEEROPT support for batch is disabled. This means the old behavior of the Language Environment is still in place.

```
CEEROPT(ALL|COMPAT)
CELQROPT(ALL|NONE)
```

Figure 17-1 New CEEPRMxx parmlib member parameters

Where:

CEEROPT This parameter indicates whether region-specific run-time options should be used in a non-CICS or non-LRR environment.

- **COMPAT** - Attempts a load and use of CEEROPT in all AMODE 31 environments.
- **ALL** - Attempts a load and use of CEEROPT only in CICS or LRR environments.

CEEQROPT This parameter indicates whether region-specific run-time options should be used in AMODE 64.

- **NONE** - Does not attempt a load or use of CELQROPT in AMODE 64 environments.
- **ALL** - Attempts a load and use of CELQROPT in all AMODE 64 environments.

Suggestion: Use the CEEPRMxx support to set system level defaults for Language Environment instead of using the CEECOPT, CEEDOPT, and CELQDOPT usermods to set installation-wide defaults.

Operator commands

To set or check the current settings for CEEROPT or CEEQROPT, the SETCEE and D CEE commands have been enhanced. Figure 17-2 shows a sample SETCEE command to set the behavior of the CEEROPT parameter to ALL.

```
SETCEE CEEROPT,ALL
CEE3743I THE SETCEE COMMAND HAS COMPLETED.
```

Figure 17-2 Sample SETCEE command

The enhanced D CEE command is useful for checking the value of the CEEROPT or CEEQROPT parameters. Figure 17-3 displays the setting of the CEEROPT parameter. ALL indicates that CEEROPT is active for all users.

```
D CEE,CEEROPT
CEE3745I 09.31.09 DISPLAY CEEROPT
CEE=(00) 752
SETCEE COMMAND    CEEROPT (ALL)
```

Figure 17-3 Sample DISPLAY CEE command

Note: CICS TS 3.1 and higher supports XPLINK programs in a CICS environment. The CICS region defaults are not used for these programs. However, if the parmlib keyword CEEROPT is set to ALL, the XPLINK programs can use the region defaults that can be located in the MVS search order.

Order of precedence

The CEEROPT load module to be used is the one in the program search order. Language Environment loads and merges the options in CEEROPT with the installation default run-time options. The search order for LE load modules is as follows:

1. Options in the storage tuning exit
2. Options in the assembler user exit
3. “Invocation” level options (JCL PARM= parameter, TSO command line, _CEE_RUNOPTS)

4. “Programmer default” level options (CEEUOPT, #pragma runopts, PLIXOPT)
5. Region-wide default options (CEEROPT/CELQROPT)
6. System-level options (SYS1.PARMLIB members and **setcee** commands)
7. CEEDOPT/CEECOPT/CELQDOPT

17.2 CEEPRMxx syntax checker

Currently users who use the Language Environment, the CEEPRMxx parmlib members do not have the functionality that would enable them to syntax check the members without activating them. Users often maintain the CEEPRMxx members from the application side rather than the system side.

With z/OS V1R10, there is now the possibility to verify CEEPRMxx parmlib members with a new syntax checker. This allows the members to be checked prior to using them during a system IPL or via operator commands when activating a set of Language Environment parmlib members. This new function allows the syntax check to be done through a batch job or CLIST. This function requires the created CEEPRMxx member to reside in a PDS or PDSE. The CEEPRMxx member is then passed to the syntax checker as input.

17.2.1 Using syntax checker in batch

To syntax check a CEEPRMxx member, the new CEEPRMCC program is available to use in batch. The CEEPRMCC program reads and parses the CEEPRMxx member for syntax errors. The CEEPRMCC program displays the run-time options report if no errors are found. The run-time options report only displays options that are specified inside the CEEPRMxx members. The CEEPRMCC expects the following inputs:

- The PARM parameter

  Use the PARM parameter of the EXEC job control statement to select one or more CEEPRMxx members.

  ```
  // PARM='CEE=(xx,yy,...,nn)'
  The two alphanumeric characters, xx,yy,...,nn, are the suffix of the
  CEEPRMxx members to be checked. Embedded blanks are not allowed within the
  PARM.
  ```

- An optional CEEPRMCK DD statement

  Use a DD statement to specify the data set where CEEPRMxx members are located:

  ```
  //CEEPRMCK DD DSN=MEENAK.SYSTEM.PARMLIB,DISP=SHR
  If no DD is specified, the CEEPRMCC program uses the default data set
  SYS1.PARMLIB.
  ```

Sample batch job

Figure 17-4 on page 363 shows a sample job. You have to code the suffix of the CEERPMxx member that you want to check in the parm field of the CEEPRMCC program. The CEEPRMCK DD card points to the library that is holding the member that you want to check. More than one CEEPRMxx member can be specified, as follows:

```
// PARM='CEE=(xx,yy,...,nn)'
```

```
//LUTZCEE  JOB ,'KUEHNER',NOTIFY=&SYSUID,.....
//CEEPRMCC EXEC PGM=CEEPRMCC,PARM='CEE=(00)'
//SYSPRINT DD  SYSOUT=*
//CEEPRMCK DD  DISP=SHR,DSN=SYS1.PARMLIB
//
```

Figure 17-4 sample Job for CEEPRMCC

Return codes and messages

If there are no errors in the CEEPRMxx members, the job issues a condition code 0. Any other return codes indicate errors. Figure 17-6 lists all possible return codes.

Figure 17-5 shows error messages from the CEEPRMCC program.

```
CEE3761I The following messages pertain to the call to the Language Environment Parmlib
checker.
CEE3731I The following messages pertain to the system default run-time options in the
CEEDOPT in CEEPRM00.
CEE3611I The run-time option ALG31 was an invalid run-time option or is not supported in
this release of Language Environment.
CEE3762I The Language Environment Parmlib checker has completed.
```

Figure 17-5 output of a failed CEEPRMCC

Figure 17-6 displays the possible return codes that can be issued during the syntax check using the CEEPRMCC program.

Returned Code	Description
0	Successful completion
4	No members were specified.
8	Input members were not specified or were not valid.
12	A closing parenthesis was missing when specifying input members.
16	1 or more chars were found after the closing parenthesis of the input members.
20	1 or more embedded blanks were found.
24	SYS1.PARMLIB allocation to the CEEPRMCK DD failed.
28	The specified data set had 1 or more incorrect attributes.
32	The specified data set did not exist.
1004	The input members string contained a single char suffix.
1008	The specified members could not be read.
1012	The specified members has 1 or more syntax errors.

Figure 17-6 Return codes from the CEEPRMCC program

17.2.2 Using syntax checker in TSO/E

The other way to syntax check a CEEPRMxx member is with TSO/E. A new CEEPRMCK CLIST reads and parses the CEEPRMxx member for syntax errors. CEEPRMCK displays the run-time options report if no errors are found. The run-time options report only displays

options that are specified inside the CEEPRMxx members. Figure 17-7 on page 364 displays the syntax of the new CEEPRMCK CLIST.

```
>>-CEEPRMCK--MEMBERS(---xx-+-)-------------------------------->
>--+-----------------------------+------------------------><
   |-DSN--(--data-set-name--)------|
   |-DSN--(--'data-set-name'--)----|
   |-DSNAME--(--data-set-name--)---|
   |-DSNAME--(--'data-set-name'--)-|
```

Figure 17-7 Syntax of the CEEPRMCK CLIST

Where:

xx The two alphanumeric characters that are the suffix of the CEEPRMxx members to be checked. The MEMBERS keyword parameter must always be specified.

data-set-name The data set name that contains the specified CEEPRMxx member. The fully qualified data set name must be enclosed in single quotes if a TSO/E prefix is not required.

DSN/DSNAME These keyword parameters are optional. If both DD is allocated and DSN or DSNAME is specified, the CEEPRMCK program uses the DD and the DSN/DSNAME is ignored. DD allocation overrides DSN/DSNAME specification. If no DD is allocated and no DSN or DSNAME is specified, the CEEPRMCK program uses the default data set SYS1.PARMLIB.

Using CEEPRMCK

An input data set must be a fixed record format and a record length of 80. To invoke CEEPRMCK by using the documented syntax, SCEECLST must be allocated to a system file (SYSPROC or SYSEXEC). See *z/OS TSO/E REXX User's Guide*, SA22-7791 for more information about setting up and using REXX execs.

To invoke CEEPRMCK you can use the ISPF Option 6 command line or call it from your own REXX script. A sample command is as follows:

```
CEEPRMCK MEMBERS(OO) DSN('SYS1.PARMLIB')
```

Where:

MEMBERS The two alphanumeric characters that are the suffix of the CEEPRMxx member(s) to be checked. The MEMBERS keyword parameter must always be specified.

DSNAME/DSN The data set name that contains the specified CEEPRMxx member. The fully qualified data set name must be enclosed in single quotes if a TSO/E prefix is not desired. The DSN/DSNAME keyword parameter is optional. If both DD is allocated and DSN or DSNAME is specified, then the CEEPRMCK program will use the DD and the DSN/DSNAME will be ignored. DD allocation overrides DSN/DSNAME specification.If no DD is allocated and no DSN or DSNAME is specified, then the CEEPRMCK program will use the default data set SYS1.PARMLIB

17.3 Health Check for LE parmlib member

With z/OS V1R10, default Language Environment run-time options can be set within a CEEPRMxx parmlib member. If USERMODs are in use, they should be converted to use parmlib.

New LE health check with z/OS V1R10

This new health check checks whether you have or have not implemented a CEEPRMxx parmlib member. The health check for Language Environment is called CEE_USING_LE_PARMLIB, as shown in Figure 17-8. With this check, a CEEPRMxx parmlib member has not been activated and the check status is shown as an EXCEPTION.

This LE health check verifies use of Language Environment parmlib CEEPRMxx. The reason for the check is to indicate that the default Language Environment run-time options should be set within a CEEPRMxx parmlib member. If USERMODs are in use, they should be converted to use parmlib.

z/OS releases that this check applies to are z/OS V1R8 and later.

```
SDSF HEALTH CHECKER DISPLAY  SC70                      DATA SET DISPLAYED
COMMAND INPUT ===>                                        SCROLL ===> PAGE
NP   NAME                              CheckOwner       State              Statu
     ASM_LOCAL_SLOT_USAGE              IBMASM           ACTIVE(ENABLED)    SUCCE
     ASM_NUMBER_LOCAL_DATASETS         IBMASM           ACTIVE(ENABLED)    SUCCE
     ASM_PAGE_ADD                      IBMASM           ACTIVE(ENABLED)    SUCCE
     ASM_PLPA_COMMON_SIZE              IBMASM           ACTIVE(ENABLED)    EXCEP
     ASM_PLPA_COMMON_USAGE             IBMASM           ACTIVE(ENABLED)    SUCCE
S    CEE_USING_LE_PARMLIB              IBMCEE           ACTIVE(ENABLED)    EXCEP
     CNZ_AMRF_EVENTUAL_ACTION_MSGS     IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_CONSOLE_MASTERAUTH_CMDSYS     IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_CONSOLE_MSCOPE_AND_ROUTCODE   IBMCNZ           ACTIVE(ENABLED)    EXCEP
     CNZ_CONSOLE_ROUTCODE_11           IBMCNZ           ACTIVE(ENABLED)    EXCEP
     CNZ_EMCS_HARDCOPY_MSCOPE          IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_EMCS_INACTIVE_CONSOLES        IBMCNZ           ACTIVE(DISABLED)   GLOBA
     CNZ_SYSCONS_MSCOPE                IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_SYSCONS_PD_MODE               IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_SYSCONS_ROUTCODE              IBMCNZ           ACTIVE(ENABLED)    SUCCE
     CNZ_TASK_TABLE                    IBMCNZ           ACTIVE(ENABLED)    SUCCE
```

Figure 17-8 CK SDSF panel

Selecting CEE_USING_LE_PARMLIB, health checker displays the messages shown in Figure 17-9 on page 366.

```
CHECK(IBMCEE,CEE_USING_LE_PARMLIB)
START TIME: 08/03/2008 08:03:05.073838
CHECK DATE: 20071231  CHECK SEVERITY: LOW

* Low Severity Exception *

CEEH012E Language Environment Parmlib is not in use.

  Explanation:  Language Environment Parmlib support is not in use.

    Parmlib should be used to set default Language Environment runtime
    options. If USERMODs are in use, they should be converted to
    Parmlib.

    Automate Language Environment Parmlib support by use of a CEE=xx
    statement in IEASYSyy data set or in the IPL PARMS.

  System Action:  The system continues processing.

  Operator Response:  Report this problem to the system programmer.

  System Programmer Response:  Enable use of Language Environment
    Parmlib support.
To clear this check, update the Parmlib member for the next IPL. For
    this session run 'set cee=XX'.

  Problem Determination:  N/A

  Source:  Language Environment

  Reference Documentation:  See: "Customizing Language Environment
    run-time options" in z/OS Language Environment Customization

  Automation:  N/A

  Check Reason:  Verifies use of Language Environment Parmlib

END TIME: 08/03/2008 08:03:05.121107  STATUS: EXCEPTION-LOW
```

Figure 17-9 Display of the health check

Activating a CEEPRMxx parmlib member

Figure 17-10 shows how to dynamically activate a CEEPRMxx parmlib member using the SET CEE command.

```
D CEE
CEE3744I 09.11.28 DISPLAY
NO MEMBERS SPECIFIED 594
SET CEE=00
IEE252I MEMBER CEEPRM00 FOUND IN SYS1.PARMLIB
CEE3742I THE SET CEE COMMAND HAS COMPLETED.
```

Figure 17-10 Set the CEEPRMxx parmlib member to be used

Figure 17-11 shows the new status of the Language Environment settings after successful activation of the CEEPRM00 parmlib member.

```
CHECK(IBMCEE,CEE_USING_LE_PARMLIB)
START TIME: 05/02/2008 09:12:11.230752
CHECK DATE: 20071231  CHECK SEVERITY: LOW

CEEH010I Language Environment Parmlib is in use.
This check ran successfully and found no exceptions.

CEEH011I The following options groups are in use: CEEDOPT, CEECOPT,
CELQDOPT

END TIME: 05/02/2008 09:12:11.230946  STATUS: SUCCESSFUL
```

Figure 17-11 Successful CEEPRMxx parmlib member check

User override of IBM values

It is possible to override check values on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command.

Figure 17-12 shows keywords you can use to override check values on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. This statement may be copied and modified to override the check defaults.

```
UPDATE CHECK(IBMCEE,CEE_USING_LE_PARMLIB)
       SEVERITY(LOW)
       INTERVAL(ONETIME)
       DATE(20071231)
       REASON('Verifies use of Language Environment Parmlib')
```

Figure 17-12 Overridable CEEPRMxx values

17.4 Multithreaded C/C++ I/O performance

Multithreaded C/C++ applications that perform I/O against an individual file on a single thread incur unnecessary overhead of *mutex* lock and unlock around each I/O call. Mutex stands for mutual exclusion object. In computer programming, a mutex is a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously. When a program is started, a mutex is created with a unique name. After this stage, any thread that needs the resource must lock the mutex from other threads while it is using the resource. The mutex is set to unlock when the data is no longer needed or the routine is finished. All this behavior takes time and has a direct impact on the application's performance.

Multithreaded C/C++ applications that perform I/O against an individual file on a single thread incur unnecessary overhead of a mutex lock or unlock around each I/O call.

z/OS V1R10 enhancements

If user programs include multithreaded operations, be aware of changes in the behavior of pragma variables with z/OS V1R10. With this release, performance is improved on

multithreaded I/O C/C++ applications by providing mechanisms to reduce or eliminate mutex lock and unlock calls during I/O, as follows:

- Providing a set of C/C++ functions (unlocked versions of current I/O C/C++ functions)

  Unlocked C/C++ functions do exactly the same work as the corresponding lock versions of C/C++ functions, with the only difference being that it's not safe for multithread. Unlocked C/C++ functions work together with flockfile(), or ftrylockfile() functions to lock the stream.

- Adding the new "samethread" keyword to fopen(), freopen(), and fdopen() C/C++ functions

  These functions mark a FILE * as being restricted to the thread that did the open. It will indicate that all I/O operations will be done on the same thread. These I/O functions do not have to obtain the pthread mutex lock.

New C/C++ functions

These functions do not use the mutex algorithm and should only be used in single-threaded applications. This improves performance, since there is no locking. The additional benefit of the _unlocked functions, when used in conjunction with flockfile() and funlockfile(), is that a series of I/O operations can be performed on a single thread without the chance of another thread reading or writing data in between the sequence.

Table 17-1 lists the 52 new C/C++ functions that were implemented in z/OS V1R10.

Table 17-1 Added C/C++ functions

clearerr_unlocked();	feof_unlocked();	ferror_unlocked();	fflush_unlocked();
fgetc_unlocked();	fgetpos_unlocked();	fgets_unlocked();	fileno_unlocked();
fprintf_unlocked();	fputc_unlocked();	fputs_unlocked();	fread_unlocked();
fscanf_unlocked();	fseek_unlocked();	fseeko_unlocked();	fsetpos_unlocked();
ftell_unlocked();	ftello_unlocked();	fwrite_unlocked();	gets_unlocked();
perror_unlocked();	printf_unlocked();	puts_unlocked();	rewind_unlocked();
scanf_unlocked ();	fldata_unlocked();	ungetc_unlocked();	vfprintf_unlocked();
vfscanf_unlocked();	vprintf_unlocked();	vscanf_unlocked();	fgetwc_unlocked();
fgetws_unlocked();	fputwc_unlocked();	fputws_unlocked();	fwide_unlocked();
fwprintf_unlocked();	fwscanf_unlocked();	getwc_unlocked();	getwchar_unlocked();
putwc_unlocked();	putwchar_unlocked();	ungetwc_unlocked();	vfwprintf_unlocked();
vfwscanf_unlocked();	vwprintf_unlocked();	vwscanf_unlocked();	wprintf_unlocked();
wscanf_unlocked();	flocate_unlocked();	fdelrec_unlocked();	fupdate_unlocked();

Note: C/C++ functions such as tmpfile(), setvbuf(), setbuf(), fclose(), fdopen(), fopen(), freopen(), pclose(), and popen() do not have an unlocked version because they always get a lock when multithreaded.

Adding the new functions

To use the new functions in an application program, add the following:

- Add the Feature Test Macro in the corresponding header:

  `_OPEN_SYS_UNLOCKED_EXT`

  When defining this Feature Test Macro before the function header, use the functions flockfile(), ftrylockfile(), and funlockfile() to get explicit application-level locking of stdio (FILE*) objects. Figure 17-13 shows a sample C Program that uses the new functions.

```
#define _OPEN_SYS_UNLOCKED_EXT
#include <stdio.h>
#include <stdint.h>
#include <string.h>

int main() {
   int   rc;
   char  rbuffer[80];
   char *stream ="./myfile.data";

   if ((s1 = fopen(stream,"w,recfm=*")) == NULL)
       return  -1;
   rbuffer[0]='A';
   rbuffer[1]='\n';

   rc = (int) fwrite_unlocked(rbuffer,1,2, s1);
   return 0;
```

Figure 17-13 Sample source of unlocked functions

- The samethread keyword

  The `samethread` option used by the fopen(), freopen(), fdopen() keyword options for multithreaded C/C++ applications are new when only one thread in the process does I/O against the stream. The stream will be restricted to the thread that did the open.

  Note: This new keyword is invoked just like the other “Keyword Parameters for File Mode”.

  Figure 17-14 on page 370 shows a sample C program using the new samethread option during an fopen().

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
int main() {
   int   rc;
   char  rbuffer[80];
   char *stream ="./myfile.data";

   if ((s1 = fopen(stream,"w,samethread,recfm=*")) == NULL)
       return  -1;
   rbuffer[0]='A';
   rbuffer[1]='\n';

   rc = (int) fwrite(rbuffer,1,2, s1);
   return 0;
```

Figure 17-14 Sample program using samethread

17.5 VSCR for Language Environment

z/OS V1R10 support in Language Environment components CEL and C-RTL reduces the use of common storage both above and below the 16 MB line.

Due to reduction of memory below the line, LE is changing the location of several control blocks. Also, Language Environment load modules have been reduced in size. The following changes have been made to support this VSCR enhancement:

- The CEEDOPT CSECT is removed from CEEBINIT, CEEBINSS, CEEPIPI, and CEEBPICI. If you had multiple copies of CEEBINIT in the past to get different sets of installation default options, we suggest using CEEPRMxx or the new CEEROPT for batch support instead.
- VSAM buffers and control blocks below the line are being moved above the line.
- CEEDOPT is being moved to the CEEPLPKA load module. CEEBINIT, CEEBINSS, CEEBPICI, and CEEPIPI are being reduced in size.
- Miscellaneous obsolete code and CSECTs are being removed to reduce the size of RMODE 24 modules.
- Patch areas are being removed from several Language Environment modules.

17.6 Reduce contention when heap pools are used

Currently, AMODE 64 C/C++ applications cannot exploit heap pools for 31-bit storage (storage above the 16 MB line and below the 2 GB bar). This means that these applications could not exploit heap pools when using the __malloc31() function.

With z/OS V1R10, Language Environment is providing support for the HEAPPOOLS run-time option in AMODE 64 applications.

Note: This change is also available for z/OS V1R7, z/OS V1R8, and z/OS V1R9 via the following APARS:

PK41618, PK47298, PK49427, PK44554, and PK57579

The benefit of this implementation is that AMODE 64 C/C++ applications running under Language Environment and which already use heap pools to manage storage above the 2 GB bar with the HEAPPOOLS64 run-time option, can now exploit heap pools above the 16 MB line and below the 2 GB bar with the HEAPPOOLS run-time option.

Note: The heap pools algorithm allows for between one and twelve sizes of storage cells that are allocated from pools out of the heap. For each size, from one to 255 pools can be created where each pool is used by a portion of the threads for allocating storage. The sizes of the cells, the number of pools for each size, and cell pool extents are specified with the HEAPPOOLS run-time option, which is also used to enable the heap pools algorithm.

HEAPPOOLS parameter syntax (C/C++ only)

The HEAPPOOLS run-time option is used to control an optional heap storage management algorithm known as heap pools. This algorithm is designed to improve performance of multithreaded C/C++ applications with high usage of malloc(), __malloc31(), calloc(), realloc(), free(), new(), and delete(). When active, heap pools can eliminate contention for heap storage. With z/OS V1R10, this option is changed. Now you are able to allocate a larger pool for a specified cell size. Contention is reduced by reducing the number of threads that are allocating from the same pool.

Applications issue many storage requests with a malloc() of 65536 bytes or less, because the heap pools algorithm is not used in a malloc() that is greater than 65536 bytes.

Following is the changed syntax of the HEAPPOOLS run-time option with z/OS V1R10.

```
HEAPPOOLS = ( ( OFF | ON | ALIGN ), cell-size , percentage
( cell-size , pool-count ) ) ,  OVR | NONOVR )
```

New parameters with z/OS V1R10 are as follows:

HEAPPOOLS=OFF Specifies that the heap pools algorithm is not used.

(cell-size,pool-count) `pool-count` is new; it specifies the number of pools to create for the cell size. pool-count must be in a range from 1 to 255. This option can be repeated 11 times.

Examples are as follows:

```
Non-CICS default:
HEAPPOOLS(OFF,8,10,32,10,128,10,256,10,1024,10,2048,10,0,10,0,
10,0,10,0,10,0,10,0,10)

CICS default:
HEAPPOOLS(OFF,8,10,32,10,128,10,256,10,1024,10,2048,10,0,10,0,10,0,
10,0,10,0,10,0,10)

AMODE 64 default:
HEAPPOOLS(OFF,8,10,32,10,128,10,256,10,1024,10,2048,10,0,10,0,
10,0,10,0,10,0,10,0,10)
```

Choosing the number of pools for a cell size

Contention occurs when two or more threads are allocating or freeing cells that are the same size at the same time. Using multiple pools should eliminate some of this contention because only a portion of the threads will be allocating from each pool. For most cell sizes, there is little contention and one pool is sufficient. However, there may be one or two cell sizes where a lot of successful get heap requests are occurring and the maximum cells used is high. These sizes can be candidates for multiple pools. Determining the optimum number of pools to use for these cell sizes involves comparing performance measurements, like throughput, when different values are used for a representative application workload.

Tuning the heap pools algorithm for an application

This is a three-step process:

1. This option is changed with z/OS V1R10. Run your application with the run-time options HEAPPOOLS(ON) or HEAPPOOLS(ALIGN), as appropriate, (using the default cell sizes and percentages) and RPTSTG(ON) for some time with a representative application workload. Then examine the HEAPPOOLS Statistics and HEAPPOOLS Summary sections of the Storage Report for Enclave report.
2. Change the cell sizes in the HEAPPOOLS run-time option to the Suggested Cell Sizes from the first run. Re-run the application with a representative workload, using the default percentages in the HEAPPOOLS option. Examine the storage report.
3. The values listed as Suggested Percentages for current Cell Sizes should be the optimal values to minimize storage use.

   Figure 17-15 shows a cell size with multiple pools. The pool number is displayed in the form a.b where a is the number for the cell size and b is the number for the pool within the cell size. The number of successful get requests for each pool was added. When there are multiple pools for a cell size, this can be used to see how the get requests are distributed among the pools. If most of the get requests are in one pool, a different number of pools may work better.

```
HEAPPOOLS Statistics:
  Pool  1     size:     8 Get Requests:           3
    Successful Get Heap requests:     1-    8               3
  Pool  2     size:    32 Get Requests:         268
    Successful Get Heap requests:     9-   16              36
    Successful Get Heap requests:    17-   24               3
    Successful Get Heap requests:    25-   32             229
.
.
.
  Pool  5.1   size:  1024 Get Requests:         230
  Pool  5.2   size:  1024 Get Requests:           3
  Pool  5.3   size: et Heap requests:   257-  264              225
    Successful Get Heap requests:   273-  280               2
    Successful Get Heap requests:   281-  288              10
    Successful Get Heap requests:   841-  848               1
  Pool  6     size:  2048 Get Requests:           2
    Successful Get Heap requests:  1113- 1120               1
    Successful Get Heap requests:  1137- 1144               1
```

Figure 17-15 HEAPPOOLS statistics with multiple pools for a cell size

Figure 17-16 shows that for a cell size with multiple pools, one line is displayed for each pool of that size. In the suggested percentages for current cell sizes, the runtime option is displayed with the multiple pools specified.

```
HEAPPOOLS Summary:
  Specified Element   Extent   Cells Per  Extents    Maximum      Cells In
  Cell Size Size      Percent  Extent     Allocated  Cells Used   Use
  ------------------------------------------------------------------------
        8        16     10         204          1           1           0
       32        40     10          81          3         226         225
      128       136     10          24          4          88          77
      256       264     10          12          1           1           0
     1024      1032     10           4         57         228         227
     1024      1032     10           4          1           2           0
     1024      1032     10           4          1           3           0
     2048      2056     10           4          1           2           2
  ------------------------------------------------------------------------
  Suggested Percentages for current Cell Sizes:
    HEAPP(ON,8,1,32,28,128,37,256,1,(1024,3),90,2048,13,0)
  Suggested Cell Sizes:
    HEAPP(ON,
          32,,56,,88,,120,,128,,168,,
          208,,248,,288,,848,,1144,,2080,)
```

Figure 17-16 HEAPPOOLS Summary with multiple pools for a cell size

17.7 IEEE decimal floating-point support

z/OS V1R9 XL C/C++ supports the decimal floating-point formats in addition to the current hex and binary floating-point formats. The decimal formats are specified by the revised IEEE 754 Floating Point Standard. This support assists with avoiding potential rounding problems, which can result from using binary or hexadecimal floating-point types to handle decimal calculations. z/OS XL C/C++ provides the following:

- Decimal type specifiers through the _Decimal32, _Decimal64, and _Decimal128 reserved keywords
- Decimal literal support via conversion between decimal types and the other floating-point types

z/OS V1R10 enhancements

The support provided is based on a core subset of the following standards:

- ANSI/IEEE Standard P754/D0.15.3

  IEEE Standard for floating-point arithmetic
- ISO/IEC TR24732

  Extensions for the programming language C to support decimal floating-point arithmetic
- ISO/IEC TR24733

  Extensions for the programming language C++ to support decimal floating-point arithmetic

Support is required by the XL C/C++ Run-Time Library to fulfill the overall System z and z/OS support of decimal floating-point numbers. There are several new functions added in the XL C/C++ Run-Time Library:

- z/OS V1R10 supports features in the following standards at the system level:
  - ISO/IEC 9945-1:1990 (POSIX-1)/IEEE POSIX 1003.1-1990
  - The core features of IEEE POSIX 1003.1a, Draft 6, July 1991
  - IEEE Portable Operating System Interface (POSIX) Part 2, P1003.2
  - The core features of IEEE POSIX 1003.4a, Draft 6, February 1992 (the IEEE POSIX committee has renumbered POSIX.4a to POSIX.1c)
  - The core features of IEEE 754-1985 (R1990) IEEE Standard for Binary Floating-Point Arithmetic (ANSI), as applicable to the IBM System z environment.
  - A subset of IEEE Std. 1003.1-2001 (Single UNIX Specification, Version 3).
- z/OS XL C/C++ also supports IEEE 754 floating-point representation (base-2 or binary floating-point formats). By default, float, double, and long double values are represented in z/Architecture floating-point formats (base-16 floating-point formats). However, the IEEE 754 floating-point representation is used if you specify the FLOAT(IEEE) compiler option.
- Since support is required by the XL C/C++ Run-Time Library to fulfill the overall System z and z/OS support of decimal floating point numbers, z/OS V1R10 is expanding the available set of XL C/C++ Run-Time Library decimal floating point functions and macros. The benefit of this change is that decimal floating-point applications can begin to use an additional set of functions and macros.

17.7.1 New decimal floating-point functions

The new decimal floating-point functionality made available in this release is an expansion of the initial support from z/OS V1R9. C/C++ applications can now take advantage of functions and macros that were previously unavailable. The following support is added for decimal floating-point in z/OS V1R10:

- New decimal floating-point math functions are provided

  Following are the math.h math functions new for C/C++ decimal floating point:

acosd32()	acosd64()	acosd128()
acoshd32()	acoshd64()	acoshd128()
asind32()	asind64()	asind128()
asinhd32()	asinhd64()	asinhd128()
atand32()	atand64()	atand128()
atan2d32()	atan2d64()	atan2d128()
atanhd32()	atanhd64()	atanhd128()
__atanpid32()	__atanpid64()	__atanpid128()
coshd32()	coshd64()	coshd128()

  C/C++ decimal floating point math functions continued:

erfd32()	erfd64()	erfd128()
erfcd32()	erfcd64()	erfcd128()
lgammad32(0	lgammad64()	lgammad128()
remainderd32()	remainderd64()	remainderd128()
sinhd32()	sinhd64()	sinhd128()
tand32()	tand64()	tand128()
tanhd32()	tanhd64()	tanhd128()
tgammad32()	tgammad64()	tgammad128()

For C++ applications, the following functions are overloaded for the _Decimal32, _Decimal64, and _Decimal128 data types:

acosh()	asinh()
atanh()	erf()
erfc()	lgamma()
remainder()	tgamma()

- New macros are added to float.h

  Following are new macros that expand to various limits and evaluation formats for decimal floating-point types. The number of base 10 digits required to ensure that values that differ by only one smallest unit in the last place (ulp) are always differentiated.

```
FLT_MAXDIG10
DBL_MAXDIG10
LDBL_MAXDIG10
```

Requirements for decimal floating-point

The following are required for using the XL C/C++ Run-Time Library decimal floating-point support:

- Applications must be running on z/OS V1R10 or higher.

  APAR PK54438 rolls back the V1R10 decimal floating-point support to V1R8.
- Hardware with the Decimal Floating Point Facility must be installed.
- The DFP and ARCH(7) compiler options are required.
- The __STDC_WANT_DEC_FP__ feature test macro must be defined.

17.8 Savstack fields

In prior releases of z/OS an application was unable to use register R13 or R4 even if this program did not access the stack. But any signal or exception that occurred should have been handled by Language Environment.

z/OS V1R10 provides control block fields where the Language Environment stack pointer can be saved. This provides for a better recovery environment if an exception occurs when the stack pointer is not in its register, and possible better performance for the application that has an extra register to use.

In addition, provided an additional fields in the CAA/LCA control block where the Language Environment stack pointer can be saved. This additional fields provide also better RAS if an

exception occurs when the stack pointer is not in its register. Possible better performance for the application that has an extra register to use.

LE common anchor area (CAA)

Each thread is represented by a common anchor area (CAA). The CAA is generated during thread initialization and deleted during thread termination. It is pointed to by CEELCA_CAA. Language Environment provides new fields where the stack pointer can be saved:

- For AMODE 31 applications, the two fields will be defined in the CAA:
  - CEECAA_SAVSTACK

    The CEECAA_SAVSTACK field can be used by an application or a compiler to save the stack pointer before calling a routine by using the OS_NOSTACK linkage. After the call returns, the CEECAA_SAVSTACK field must be set back to zero.
  - CEECAA_SAVSTACK_ASYNC

    The CEECAA_SAVSTACK_ASYNC field can be used by applications that have large sections of code that does not require access to the Language Environment stack but can benefit from having an additional register available.
- For AMODE 64 applications, the two fields will be defined in the LCA:
  - CEELCA_SAVSTACK

    The CEELCA_SAVSTACK field can be used by an application or a compiler to save the stack pointer before calling a routine using OS_NOSTACK linkage. After the call returns, the CEELCA_SAVSTACK field must be set back to zero.
  - CEELCA_SAVSTACK_ASYNC

    The CEELCA_SAVSTACK_ASYNC field can be used by applications that have large sections of code that does not require access to the Language Environment stack but can benefit from having an additional register available.

Note: The field CEELCA_SAVSTACK was added in z/OS V1R6. This line item provides better documentation for it. The other fields are new for z/OS V1R10.

Using in AMODE 31

The CEECAA_SAVSTACK field may be used by an AMODE 31 application or a compiler to save the stack pointer before calling a routine using OS_NOSTACK linkage. After the call returns, the CEECAA_SAVSTACK field must be set back to zero. When the Language Environment ESPIE exit routine, ESTAE exit routine or signal interrupt routine (SIR) gets control and the value in CEECAA_SAVSTACK is not zero, that value is used as the current stack frame. For asynchronous signal processing, typically the interrupt PSW will be outside the routine that owns the stack frame and the signal is put back. The C macro __LE_SAVSTACK_ADDR defined in the sample header file edcwccwi.h is the address of the CEECAA_SAVSTACK field.

The CEECAA_SAVSTACK_ASYNC field may be used by AMODE 31 applications that have large sections of code that do not require access to the Language Environment stack but could benefit from having an additional register available. The CEECAA_SAVSTACK_ASYNC field is a pointer to the field where the stack pointer will be saved. Language Environment initializes CEECAA_SAVSTACK_ASYNC to zero. It is the responsibility of the application to set up the field where the stack pointer will be saved and store the address of that field in CEECAA_SAVSTACK_ASYNC. The storage for the field should be in the application key and persist for the life of the thread. When initializing LCEECAA_SAVSTACK_ASYNC, appropriate action should be taken if CEECAA_SAVSTACK_ASYNC is not zero. Since it is possible to access the field where

stack pointer will be stored, consider the consequences if some part of the application is doing so.

Whenever the Language Environment stack is being used, either CEECAA_SAVSTACK_ASYNC must be zero or the field pointed to by CEECAA_SAVSTACK_ASYNC must be zero. When the Language Environment ESPIE exit routine, ESTAE exit routine or signal interrupt routine (SIR) gets control and CEECAA_SAVSTACK_ASYNC is not zero and the value in the field pointed to by CEECAA_SAVSTACK_ASYNC is not zero, that value is used as the current stack frame. For asynchronous signal processing, the signal will always be handled as if the interrupt PSW as inside the routine that owns the stack frame. The C macro __LE_SAVSTACK_ASYNC_ADDR defined in sample header file edcwccwi.h is the address of the CEECAA_SAVSTACK_ASYNC field.

Using in AMODE 64

The CEELCA_SAVSTACK field may be used by an AMODE 64 application or a compiler to save the stack pointer before calling a routine using the OS_NOSTACK linkage. After the call returns, the CEELCA_SAVSTACK field must be set back to zero. When the Language Environment ESPIE exit routine, ESTAE exit routine, or signal interrupt routine (SIR) gets control and the value in CEELCA_SAVSTACK is not zero, that value is used as the current stack frame.

For asynchronous signal processing, typically the interrupt PSW will be outside the routine that owns the stack frame and the signal is put back. The C macro __LE_SAVSTACK_ADDR defined in sample header file edcwccwi.h is the address of the CEELCA_SAVSTACK field. The CEELCA_SAVSTACK_ASYNC field may be used by AMODE 64 applications that have large sections of code that do not require access to the Language Environment stack but could benefit from having an additional register available. The CEELCA_SAVSTACK_ASYNC field is a pointer to the field where the stack pointer will be saved. Language Environment initializes CEELCA_SAVSTACK_ASYNC to zero. It is the responsibility of the application to set up the field where the stack pointer will be saved and store the address of that field in CEELCA_SAVSTACK_ASYNC.

The storage for the field should be in the application key and persist for the life of the thread. When initializing CEELCA_SAVSTACK_ASYNC, appropriate action should be taken if CEELCA_SAVSTACK_ASYNC is not zero. Since it is possible to access the field where stack pointer will be stored, consider the consequences if some part of the application is doing so. Whenever the Language Environment stack is being used, either CEELCA_SAVSTACK_ASYNC must be zero or the field pointed to by CEELCA_SAVSTACK_ASYNC must be zero.

When the Language Environment ESPIE exit routine, ESTAE exit routine, or signal interrupt routine (SIR) gets control and CEELCA_SAVSTACK_ASYNC is not zero and the value in the field pointed to by CEELCA_SAVSTACK_ASYNC is not zero, that value is used as the current stack frame. For asynchronous signal processing, the signal will always be handled as if the interrupt PSW was inside the routine that owns the stack frame. The C macro __LE_SAVSTACK_ASYNC_ADDR defined in sample header file edcwccwi.h is the address of the CEELCA_SAVSTACK_ASYNC field.

17.9 Language Environment Compare and Trap instructions

The Compare and Trap instructions, CRT and CGRT, are a new addition to z/Architecture machine instructions. They are used to test the contents of an operand. If a test fails, a data exception with data-exception code (DXC) X'FF' is generated. Language Environment does

not create proper information to differentiate a Compare and Trap instruction data exception from other types of data exceptions. This made it difficult for applications to exploit performance advantages of the new instructions.

With z/OS V1R10, Language Environment is enhanced to provide information to condition handling routines so that they can detect that a Compare and Trap data exception has occurred.

Attention: The General-Instructions-Extension Facility of z/Architecture, which is implemented on System z10, must be available.

Two enhancements have been made to Language Environment for this support, as follows:

- New messages and feedback code are added to allow user-written condition handlers and 64-bit exception handlers to recognize that a Compare and Trap data exception has occurred.
- A new signal code (si_code) is added to allow C signal handlers to recognize that a Compare and Trap data exception has occurred.

Furthermore, the new message CEE3234S was added to represent the condition where a Compare and Trap instruction has caused a data exception. Figure 17-17 shows an example.

```
CEE3234S The system detected a Compare and Trap data exception.
Explanation:  A Compare and Trap instruction (such as CRT, CGRT, CGFRT, CIT,
CGIT, CLRT, CLGRT, CLGFRT, CLFIT, CLGIT) in your program generated a data
exception.
Programmer response:  You can use a condition handling routine to correct the
problem and resume the application.
Operator response:  None.
Other response:  None.
System action:  The thread is terminated.
Symbolic feedback code:  CEE352
```

Figure 17-17 sample CEE3234S message

The C signal handlers use a signal number and signal code (found in the siginfo_t structure that is passed as input to the handler) to help determine the cause of the signal. For this support, when a Compare and Trap instruction has caused a data exception, signal SIGFPE is generated with a new signal code, FPE_CTDXC, that is shown in Figure 17-18.

```
 #define FPE_CTDXC  44 /* SIGFPE caused by a
                          Compare and Trap
                             data exception      */
```

Figure 17-18 Added SIGPIPE signal

17.10 SDUMP serviceability

In prior releases of z/OS, Language Environment AMODE 64 dumps may not include all of the right information to diagnose a problem. This was caused by the following factors:

- Sheer amount of storage that needs to be dumped

- Order and grouping of Language Environment storage within and among multiple memory objects
- Order in which RSM presents the memory objects to be dumped

With z/OS V1R10, there are several changes in the way that AMODE 64 Language Environment allocates memory objects; also, taking advantage of RSM and Dump Services mechanisms to prioritize the order of dumping. Applications are now allowed to do obtain and free memory objects with a user-defined dump priority, and to specify a dump priority for a shared memory segment. The enhancement improves the chance that the right information will be captured in AMODE 64 Language Environment dumps.

17.10.1 The _moservices() function

The new __moservices() function is added to AMODE 64 Language Environment to allow an application to do the following:

- Create a memory object that has a user-specified dump priority and is associated with the current Language Environment enclave.
- Free a memory object that had been created using __moservices().
- Specify a shared memory dump priority to be used when allocating shared memory.

The application program can create memory objects with user-specified characteristics that are treated as part of the enclave:

- Dumped along with those of Language Environment, in priority order
- Propagated on a fork()
- Cleaned up during environment termination

The _moservices() format

```
int __moservices(int reqtype, size_t mopllen,
                 struct __mopl_s * mopl, void ** moorigin);
```

Calling parameters for the _moservices() function

Figure 17-19 shows the calling parameters for __moservices().

Argument	*Description*
reqtype	__MO_GETSTOR : Use IARV64 REQUEST(GETSTOR) to create a memory object __MO_DETACH : Use IARV64 REQUEST(DETACH) to free a memory object __MO_SHMDUMPPRIORITY : Set a shared memory dump priority
mopllen	The length of the passed mopl structure
mopl	Points to a structure describing additional characteristics for the request
moorigin	Contains the origin address of the memory object that was obtained or to be detached

Figure 17-19 callng parameter of __mosservices()

The __mopl_s structure

The __mopl_s structure is defined in stdlib.h and has the following format:

```
struct __mopl_s {
unsigned long __moplrequestsize;
int __mopldumppriority;
unsigned int __moplgetstorflags;
long __moplreserved;
int __mopl_iarv64_rc;
int __mopl_iarv64_rsn;
};
```

The entire __mopl_s structure must be set to binary zeroes prior to use. The fields are used as follows:

- __mopldumppriority

  For __MO_GETSTOR, the desired memory object dump priority (1-99). For __MO_SHMDUMPPRIORITY, the desired shared memory dump priority (1-99).

- __moplsize

  For __MO_GETSTOR, the size of the memory object to be created, in MB.

- __mopliarv64rc

  The return code from the call to IARV64

- __mopliarv64rsn

  The reason code from the call to IARV64

Specifying the dump priority in a C program

In Figure 17-20 on page 381 you see a sample C program that uses the newly implemented __moservices(). The program obtains a 100 MB memory object whose dump priority falls between the dump priorities of the Language Environment stacks and heaps. All memory objects obtained by __moservices() are unguarded and count towards the address space memlimit.

Constants are available that identify the dump priorities of memory objects obtained by Language Environment, as follows:

- __MO_DUMP_PRIORITY_STACK

  Dump priority of memory objects to be used as Language Environment stacks. Value is 5.

- __MO_DUMP_PRIORITY_HEAP

  Dump priority of memory objects to be used as Language Environment heaps. Value is 15.

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>

int main(void)  {
    __mopl_t mymopl;
    int  mos_rv;
    void * mymoptr;

    memset(&mymopl, 0, sizeof(__mopl_t));

    /* Set a shared memory dump priority for subsequent shmget() */
    mymopl.__mopldumppriority = 8;

    mos_rv = __moservices(__MO_SHMDUMPPRIORITY, sizeof(mymopl),
                          &mymopl, &mymoptr);

    if (mos_rv != 0) {
      perror("moservices(SHMDUMPPRIORITY) call failed");
    }
    mymopl.__mopldumppriority = __MO_DUMP_PRIORITY_STACK + 5;
    mymopl.__moplrequestsize = 100;

    mos_rv = __moservices(__MO_GETSTOR, sizeof(mymopl),
                          &mymopl, &mymoptr);

    if (mos_rv == 0) {
      printf("moservices(GETSTOR) successful, MO addr: %p\n",
             mymoptr);
/* Free the 100MB memory object.                        */
      mos_rv = __moservices(__MO_DETACH, 0, NULL, &mymoptr);

      if (mos_rv != 0) {
        perror("moservices(DETACH) call failed");
      }
    }
    else {
      perror("moservices(GETSTOR) call failed");
    }

    return 0;
}
```

Figure 17-20 Sample source for __moservices

17.10.2 Abnormal termination exits and dump formats

With Language Environment services in a batch environment, abnormal termination exit CEEBDATX is automatically linked at installation time; the sample exit is no longer required. The sample exit was available only in the sample library provided with AD/Cycle® LE/370 V1R3. The sample exit allows you to produce a system dump (with abend code 4039) automatically whenever an abnormal termination occurs.

17.10.3 Changes in stderr output under CICS

Output from stderr is sent to the CICS transient data queue, CESE, which is also used for Language Environment run-time error messages, dumps, and storage reports. If you previously used this file exclusively for C/370™ stderr output, you should note that the output might be different than you expect.

17.11 Language Environment for password phrase

The XL C/C++ Run-Time Library is enhanced to provide password phrase support. This allows customers to enter a password phrase in place of a password for authentication purposes, which provides a more secure authentication method.

A password phrase allows a much longer value than a traditional 8-character password. This provides extra security. It is now possible to use a 9 -100 character password phrase as an alternative authentication method to the traditional 8-character password, if the security product supports a password phrase. Two modified C functions are implemented:

- getpass() – Read A String of Characters Without Echo
- sysconf() – Determine System Configuration Options

  With both functions, PASS_MAX is increased from 8 to 255 to cover the maximum length for a password phrase. PASS_MAX only grows to 255 if the TARGET compilation environment is R10 (0X410A0000) or higher. This change affects getpass() by allowing it to return a pointer to a buffer holding the user's input password string of up to 255 bytes. This change affects sysconf(_SC_PASS_MAX) by returning the new maximum.

A getpass() function example

Figure 17-21 shows you an example of the getpass() C function.

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <unistd.h>
main() {
  char *result, *pass;
  int length;
  pass = "Enter a password: ";
  puts("examining GETPASS() limit");
  if ((result = getpass(pass)) != NULL) {
    printf("password is %s\n", result);
    printf("length of result is %d\n", strlen(result));
  }
  else
    perror("getpass() error");
}
```

Figure 17-21 sample C program for use of getpass()

In Figure 17-22 you see the output of the sample getpass() program. In our case the password length was 26 characters.

```
examining GETPASS() limit
Enter a password:
password is This is a password phrase.
length of result is 26
```

Figure 17-22 Sample output of getpass()

A sysconf() function example

Figure 17-23 shows how to use the sysconf() functions. This is to get the maximum allowable length of a passphrase in your system.

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
main() {
   long result;
   errno = 0;
   puts("examining PASS_MAX limit");
   if ((result = sysconf(_SC_PASS_MAX)) == -1)
      if (errno == 0)
        puts(“PASS_MAX is not supported.");
      else perror("sysconf() error");
   else
      printf(“PASS_MAX is %ld\n", result);
}
```

Figure 17-23 sample C program for use of setconf()

Figure 17-24 shows the output produced on our test system. The password phrase limit was set to 255 characters.

```
examining PASS_MAX limit
PASS_MAX is 255\n
```

Figure 17-24 Sample output of setconf()

Password phrase implementation for other functions

The new support for password phrases was implemented to the following functions:

- __login(), __login_applid() – Create a New Security Environment for Process

  The __login() and __login_applid() functions provide a way for a process to change its identity so as to be different than the address space identity and create a new security environment for the process. The pass_length parameter specifies the length of the password or passticket, or the password phrase defined by pass. The pass parameter specifies a user password, passticket, or password phrase

- __passwd(), __passwd_applid() – Verify/Change User Password

  The __passwd() and __passwd_applid() functions verify and/or change the username password or the password phrase. The oldpass and newpass parameters point to a NULL-terminated character string of a password or a password phrase.

- pthread_security_np(), pthread_security_applid_np() – Create or Delete Thread-level Security

  The pthread_security_np() and pthread_security_applid_np() functions create or delete a thread-level security environment for the calling thread. The password parameter specifies a user password or passticket, or password phrase.

Attention: rexec() and rexec_af() do not support the password phrase.

17.11.1 Migration considerations

A previously existing application that uses the getpass() function or the PASS_MAX macro may experience problems running on z/OS V1R10.

For getpass(), getpass() now truncates the response to its prompt at 255 characters, making it possible for an incorrectly entered password longer than 8 to get passed along by the application to one of the authentication functions. It can cause authentication failure with the assumption that a previous interaction with the application was successful (since it was truncated at 8 characters on prior releases).

For PASS_MAX, one example of a problem arises when the PASS_MAX macro is used by the application to define a password buffer and strcpy() is used to copy the getpass() returned string into the application buffer. In this case, an overlay will result if an end user response to the getpass() prompt exceeds 8 characters.

It is recommended to examine all source code using the getpass() function or the PASS_MAX macro to determine if an application is susceptible to these problems.

18

z/OS UNIX System Services

The UNIX System Services element of z/OS is a UNIX operating environment, implemented in the z/OS operating system. It is also known as z/OS UNIX. The z/OS support enables two open systems interfaces on the z/OS operating system: an application program interface (API) and an interactive shell interface.

This chapter provides new information about changed and updated functions for z/OS UNIX System Services, as follows:

- Replacing or migrating the sysplex root file system
- Password phrase support in z/OS UNIX
- Submit command
- APPLID and password phrase support
- Loading USS files into common storage
- OMVS UNIX command amblist
- Minor zFS updates

18.1 Replacing or migrating the sysplex root file system

In z/OS V1R7, IBM announced that the zFS file system was the strategic and preferred file system for z/OS. There are migration tools to facilitate the migration from an HFS file system to a zFS file system.

However, all the aids require that all UNIX work that relies on file system structure availability in the sysplex needs to be stopped before replacing the current sysplex root with a new one. This is a big restriction that inhibits installations from doing a full migration to the zFS file system.

18.1.1 The sysplex root

The sysplex root is used by the system to redirect addressing to other directories. It is very small and should be mounted read-only. It contains directories and symbolic links that allow redirection of directories, as shown in Figure 18-1. Only one sysplex root file system is used for all systems participating in a shared file system.

The sysplex root is a file system that is used as the sysplex-wide root. This file system can be initially mounted read/write and designated AUTOMOVE. Then you should switch the mode to read-only for normal production mode.

No files or code should reside in the sysplex root file system. It consists of directories and symbolic links only, and hence the size of the data set representing the sysplex root is very small.

In general, the contents of the sysplex root should only change when you need a new version root or system-specific root file system directory for your shared file system configuration. When a system is IPLed (initialized), the mount processing for the sysplex root file system will include defining the appropriate $SYSNAME or $VERSION directory in the sysplex root file system if the sysplex root is mounted as read/write at that time. These new directories can also be defined before IPLing a system that needs them.

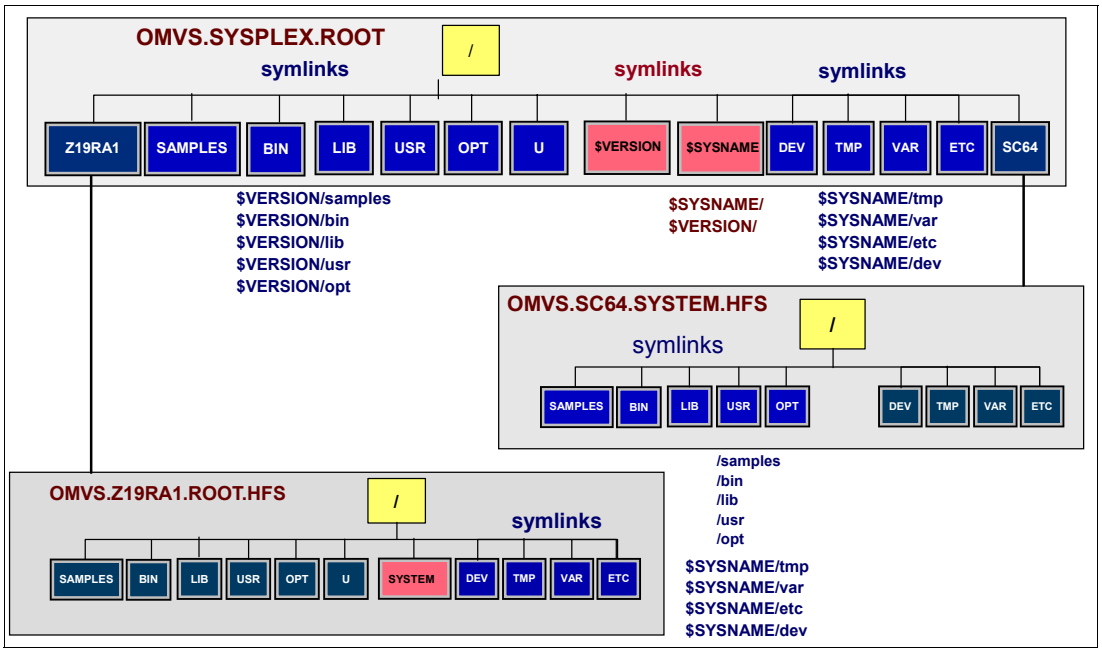


Figure 18-1 z/OS UNIX files systems in a sysplex

Migration tools

Several tools and commands are available for general migration of HFS file systems to zFS:

- BPXWH2Z tool – located in SYS1.SBPXEXEC; creates a zFS file system from the contents of an HFS file system

> **Note:** You can use the JCL sample ISPBTCH in SYS1.SAMPLIB to invoke BPXWH2Z as an ISPF batch job. Be sure to read the Notes section before you run the job. When you run BPXWH2Z on your system, it uses the z/OS V1R7 or later level of the **pax** command. This level has been enhanced for sparse file support and other characteristics that are of concern when migrating from an HFS to a zFS file system. You can manually migrate from an HFS to a zFS file system without using the BPXWH2Z tool. When doing that, the z/OS V1R7 or later level of the **pax** command is suggested. However, you would need to allocate and format the target zFS file systems.

- The **pax** command – a supported utility for data movement
- The migration tool MIGRTOOL – described in *HFS to zFS Migration Tool*, REDP-4328.

18.1.2 Sysplex root replacement

The migration of the sysplex root file system in a USS sysplex file system sharing environment from HFS to zFS required that all file systems be unmounted before the replacement could be done.

In z/OS V1R10, the replacement of the sysplex root file system can be initiated with an operator command on any system of the USS sharing environment. The benefit is that the sysplex root file system can be dynamically replaced while it is in use without disruption to active workloads.

Using this dynamic root replacement, it is possible to do the following:

- Change the sysplex root file system to the same or different PFS, either HFS or zFS.
- Installations can do a migration of the sysplex root file system to the zFS file system.
- The sysplex root file system can be migrated without unmounting all file systems.

The current sysplex root file system and the new sysplex root file system must be either HFS or zFS in any combination. In addition to being useful for HFS to zFS migration, this dynamic root replacement is also useful for moving the sysplex root generally. For example, when a installation gets a new storage subsystem and moves the data from the old storage subsystem to the new and discontinues use of the old one, one of the data sets to be moved might be the sysplex root. Being able to do this nondisruptively is an asset of this new function.

New command to migrate the sysplex root

The replacement of the sysplex root file system is initiated via the following operator command on any system in the shared file system configuration:

```
F OMVS,NEWROOT=new.root.file.system.name,COND=<Yes|No>
```

The following values are accepted:

COND=Yes Proceed conditionally if any active usage was found in the current sysplex root file system - (Default).

COND=No Proceed unconditionally if any active usage was found in the current sysplex root file system.

The command can be issued on any system in the USS file system sharing environment. It is processed on the system that is the owner of the current sysplex root file system.

Mount parameters are preserved for the same PFS file system type but will be dropped for different PFS types on the current and new sysplex root file systems.

Using COND=Yes

When the COND=Yes parameter is used and if activity in the current sysplex root file system is found, the following messages are displayed:

```
BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILE SYSTEM:
Path Name: pathname INODE: InodeNumber
BPXF244E F OMVS,NEWROOT COMMAND FAILED. RETURN CODE=00000072, REASON
CODE=124F0626
```

If the active path entry is located in a subdirectory of the sysplex root file system and not at the root level, only the plain pathname without the directory structure is displayed. In addition, the inode number is provided to identify the entry. Otherwise, the inode number is not given because it is not needed.

If sysplex root file system resources are currently being used by active workloads, wait until the current active workloads are completed, or cancel the active workloads and reissue the command.

Using COND=No

The command F OMVS,NEWROOT=new.root.file.system.name,COND=No can be used to proceed unconditionally even if activities are found in the current sysplex root file system. When the COND=No parameter is used, the active connections that will be broken on replacement of the sysplex root file system include the following:

- Open files in the root
- Open directories in the root
- Current working directories (CWDs) in the root, except the root CWD

Old connections in the old sysplex root file system may get EIO errors.

After replacement of the sysplex root

Remember to update the ROOT statement in the BPXPRMxx parmlib member that is currently in use.

- The root CWD (‘/’) is updated on all systems in the sysplex to point to the new sysplex root file system.
- New open requests go to the new sysplex root file system.
- The current sysplex root for the root directory is replaced for all processes in all systems.
- The current working directory for the sysplex root is replaced for any active processes using it.

New console messages

Following are sysplex root replacement messages indicating failure or success of the operator command:

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

BPXF244E F OMVS,NEWROOT COMMAND FAILED. RETURN CODE=retcode REASON CODE=rsncode

BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILE SYSTEM:
Path Name: pathname INODE: InodeNumber

BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED
SUCCESSFULLY.

BPXF247I SYSPLEX ROOT MOUNT PARMS ARE DROPPED ON REPLACEMENT.
```

18.1.3 Restrictions for replacement

Although a shared file system configuration is required, the sysplex can be a single system. When the sysplex root is mounted read-only, there are not any function-shipping clients as long as physical paths to the DASD are available to all systems in the sysplex. Directories, data, files, and links are not copied from one file system to another.

The following constraints must be met for dynamic replacement of the sysplex root file system:

- The system must be in a shared file system configuration.
- The sysplex root file system must be mounted read-only.
- There must not be any function shipping clients on the sysplex root file system.
- The current and the replacement sysplex root file system physical file system (PFS) must be active on all the systems in the configuration.
- The user allocates and sets up the new sysplex root file system containing all active mountpoints and symlinks as the current.
- The UID, GID, and the permission bits for the root directory of the new sysplex root file system must be the same as the current one.
- The current sysplex root file system and any directories on it cannot be exported by the DFS or SMB server.
- The new sysplex root file system cannot be DFHSM migrated or mounted or in use.
- The current and new sysplex root must be either HFS or zFS in any combination.
- All systems in the shared file system environment must be at the minimum system level of z/OS V1R10.
- If the SECLABEL class is active and the MLFSOBJ option is active, the security labels of the new and current sysplex root file system must match.

This support does not allow a dummy sysplex root for the current sysplex root, a file system must be mounted as the current root.

Migration and coexistence considerations

The minimum system level for all systems in the shared file system configuration is z/OS V1R10 to use this function. Systems at a lower release level may join the sysplex after the sysplex root replacement is completed.

18.1.4 Sample processing

We went through the steps to be taken for dynamically migrating the sysplex root file system from HFS to zFS or simply replacing the current version.

Preparation for dynamically migrating the sysplex root

The following rules need to be taken into account or followed:

- All systems in the sysplex must be at minimum z/OSV1R10.

  This is not the case in the example shown in Figure 18-5 on page 392.
- The new zFS sysplex root file system needs to be allocated and set up.
  - You can simply create the new version root and use utility copytree because the structure of the sysplex root should be very simple and small.
  - We provide another example using the alternate HFS to zFS migration tool named MIGRTOOL and described in REDP-4328.
- The new zFS sysplex root file system cannot be HSM migrated, mounted or in use.
- The new zFS sysplex root file system must contain all the active mountpoints and symlinks similar to the current HFS sysplex root.
- The UID, GID and the permission bits of the root directory in the new sysplex root must be the same as in the current root.
- All of the other restrictions mentioned in “Restrictions for replacement” on page 389 must be met.

Allocation and setup of the new sysplex root file system

If you are not changing the type of the new sysplex root file system utility ADRDSSU is probably the best means to create the new sysplex root. This assures that the internal structure of the new sysplex root is exactly the same as the old one.

You can use either of the following:

- ADDRSSU COPY
- ADDRDSSU DUMP followed by ADRDSSU RESTORE

This example of a migration from HFS to zFS uses MIGRTOOL (see *HFS to zFS Migration Tool Named MIGRTOOL*, REDP-4328 for details), using the DEFMIGR command entered from TSO and then changing the initial data presented, as shown in Figure 18-2 on page 391.

```
EDIT       SYS08129.T141324.RA000.HERING.R0300614          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 ZFS_MIGRATE_DEFINE_DSN=HERING.ZFS.MIGRATE.DEFINE
000003 ZFS_MIGRATE_DEFINE_MBR=sysroot
000004 ZFS_MIGRATE_DEFINE_DSP=replace
000005 # ------------------------------------------------------------------- #
000006
000007 # ------------------------------------------------------------------- #
000008 ZFS_DEF_DATACLASS=
000009 ZFS_DEF_MANAGEMENTCLASS=
000010 ZFS_DEF_STORAGECLASS=
000011 # ------------------------------------------------------------------- #
000012
000013 # ------------------------------------------------------------------- #
000014 HFS_DATA_SETS_TO_MIGRATE=WTSCPLX2.SYSPLEX.ROOT
000015 ZFS_AGGRNAME_CHANGE_CMD=
000016 # ------------------------------------------------------------------- #
```

Figure 18-2 Defining MIGRTOOL migration statements with DEFMIGR, part 1

After pressing PF3 to save the changes, the suggested migration statements are shown for reviewing and making changes. Figure 18-3 shows the changes of specifying the new sysplex root.

```
EDIT       HERING.ZFS.MIGRATE.DEFINE(SYSROOT) - 01.00      Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 # MIGRATE CONTROL DEFINITIONS, CREATED 2008-05-08 14:27:35             #
000003 # ------------------------------------------------------------------- #
000004
000005 # ------------------------------------------------------------------- #
000006 HFS_NAME=                      WTSCPLX2.SYSPLEX.ROOT
000007 # ------------------------------------------------------------------- #
000008 #HFS_#_VOLUMES=                1
000009 #HFS_DEVICE_TYPE=              3390
000010 #HFS_ALLOC_UNIT=               CYLINDER
000011 #HFS_ALLOC_SPACE=              1 0
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  7 Line(s) not Displayed
000019 ZFS_NAME=                      WTSCPLX2.SYSPLEX.ROOT.NEW
000020 #ZFS_#_VOLUMES=                1
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 Line(s) not Displayed
000024 ZFS_ALLOC_NUM_CAND_VOLUMES=    0
000025 ZFS_ALLOC_NUM_SEC_ALLOCS=      0
000026 ZFS_DATACLASS=
000027 ZFS_MGMNTCLASS=
000028 ZFS_STORCLASS=
000029 ZFS_REPLACES_HFS=              N
000030
```

Figure 18-3 Defining MIGRTOOL migration statements with DEFMIGR, part 2

Then use the MIGRDATA JCL to create the zFS aggregate and copy the data into the new sysplex root using **copytree** as shown in Figure 18-4.

```
EDIT       HERING.ZFS.JOB.CNTL(MIGRDATA) - 01.41          Columns 00001 00072
000004 //* ------------------------------------------------------------------
000005 //* Migrate HFS data sets to zFS compat mode aggregates
000006 //* Property of IBM  (C) Copyright IBM Corp. 2002-2007
000007 //* ------------------------------------------------------------------
000008 //  SET MGRTOOL=COPYTREE                 <=== Copy utility to be used
000009 //*             COPYMIGR: Use copy tool provided with migration tool
000010 //*             COPYPAX : Use accessible (std) pax version for copying
000011 //*             COPYTREE: Use accessible (std) copytree for copying
000012 //  SET VERBOSE=N                <=== Y or N, list all objects copied
000013 //*               VERBOSE is used only if COPYMIGR or COPYPAX are set.
000014 //  SET DEFMIGR=HERING.ZFS.MIGRATE.DEFINE(SYSROOT)   <=== MIGRATE DEFs
000015 //* ------------------------------------------------------------------
000016 //  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
000017 //  SET STEPLIB=HERING.ZFS.MIGRTOOL.LOADLIB      <=== STEPLIB library
000018 //* ------------------------------------------------------------------
000019 //COPYMIGR EXEC PGM=IKJEFT01,
000020 //  PARM='COPYMIGR &MGRTOOL. &VERBOSE. &OVERWRT.'
000021 //STEPLIB  DD DSNAME=&STEPLIB.,DISP=SHR
000022 //*        DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
000023 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000024 //STDENV   DD DATA,DLM=##
000025 # ------------------------------------------------------------------
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 18 Line(s) not Displayed
```

Figure 18-4 Using the MIGRTOOL JCL to copy the data using copytree

Replacing the sysplex root file system

If all requirements are met, you can issue the command from an operator's console.

To force a situation that blocks the switching of the sysplex root we created a file named `testfile` in the root directory before copying the contents of the file system and opened the file for reading.

Furthermore, we started with a situation of two systems not running at level z/OS V1R10, as shown in Figure 18-5.

```
F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEWX,COND=YES
BPXF243E F OMVS,NEWROOT COMMAND HAS BEEN TERMINATED DUE TO THE FOLLOWING
REASON(S):
ONE OR MORE SYSTEM IS NOT AT THE REQUIRED LFS VERSION
```

Figure 18-5 Failing the "F OMVS,NEWROOT" command

Then we carefully took down OMVS on the systems not running at level z/OS V1R10 with the F OMVS,SHUTDOWN command, as shown in Figure 18-6.

```
 F OMVS,SHUTDOWN
 BPXI055I OMVS SHUTDOWN REQUEST ACCEPTED
 D OMVS
 BPXO042I 08.07.42 DISPLAY OMVS 038
 OMVS     0011 SHUTTING DOWN    0    OMVS=(9B)
...
*BPXI056E OMVS SHUTDOWN REQUEST HAS COMPLETED SUCCESSFULLY
 BPXN002I UNIX SYSTEM SERVICES PARTITION CLEANUP COMPLETE FOR SYSTEM SC64
```

Figure 18-6 Shutting down OMVS on systems not running at level z/OS V1R10

Figure 18-7 shows the F OMVS,NEWROOT command issued again, and it fails because of the one file being opened. This is not a normal situation because you should not have any files located directly in the sysplex root file system. It could also have been a situation with a process having its current working directory set to a directory in the sysplex root, with the directory not used as an active mountpoint.

```
 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEW,COND=YES
 BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILESYSTEM:
 PATH NAME: /testfile
 BPXF244E F OMVS,NEWROOT COMMAND FAILED.
 RETURN CODE = 00000072, REASON CODE = 124F0626
 IOEZ00048I Detaching aggregate WTSCPLX2.SYSPLEX.ROOT.NEW
```

Figure 18-7 F OMVS,NEWROOT fails because of some activities in the sysplex root

The reason code 124F0626 that is shown (=JrActivityFound) means that an activity is found on the sysplex root file system. You should look for the BPXF245I message explanation and system programmer response for further details.

Finally, after closing the file and issuing the F OMVS,NEWROOT command once more, this time it successfully replaced the sysplex root HFS with the new zFS file system, as shown in Figure 18-8.

```
 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEW,COND=YES
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED
 SUCCESSFULLY.
 D OMVS,F,N=WTSCPLX2.SYSPLEX.ROOT.NEW
 BPXO045I 08.41.04 DISPLAY OMVS 070
 OMVS     0011 ACTIVE             OMVS=(1A)
 TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
 ZFS             1 ACTIVE                      READ  05/08/2008  L=210
   NAME=WTSCPLX2.SYSPLEX.ROOT.NEW                    08.29.50    Q=0
   PATH=/
   AGGREGATE NAME=WTSCPLX2.SYSPLEX.ROOT.NEW
   OWNER=SC70     AUTOMOVE=Y CLIENT=N
```

Figure 18-8 F OMVS,NEWROOT showing the “Successfully Completed” message

Steps after the migration of the sysplex root file system

The BPXPRMxx parmlib member needs to be updated with the new sysplex root file system information.

Even better is what we did later: we moved the sysplex root again twice to end up with the original name of the sysplex root name, as shown in Figure 18-9.

```
tso alter 'wtscplx2.sysplex.root' newname('wtscplx2.sysplex.root.old')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT ALTERED

 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.OLD,COND=YES
...
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED SUCCESSFULLY.
 IOEZ00048I Detaching aggregate WTSCPLX2.SYSPLEX.ROOT.NEW

tso alter 'wtscplx2.sysplex.root.new' newname('wtscplx2.sysplex.root')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT.NEW ALTERED
tso alter 'wtscplx2.sysplex.root.new.data' newname('wtscplx2.sysplex.root.data')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT.NEW.DATA ALTERED

 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT,COND=YES
 IEF196I IEF285I   WTSCPLX2.SYSPLEX.ROOT.OLD                   KEPT
...
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED SUCCESSFULLY.
```

Figure 18-9 Moving the sysplex root twice more to have the original name back

Doing so, the update in the BPXPRMxx member is not needed at once.

Note: Because specifying type HFS in BPXPRMxx on the ROOT statement also mounts a zFS file system, the correction to use ZFS™ instead can be done later.

18.2 Password phrase support in z/OS UNIX

A password phrase is a string consisting of mixed-case letters, numbers, and special characters, including blanks, that is used to control access to data and systems.

The z/OS UNIX security functions provided by RACF include validating users, file access checking, privileged user checking, and user limit checking. z/OS UNIX users are defined with RACF commands. When a job starts or a user logs on, RACF verifies the user ID and password or now with z/OS V1R10 a password phrase if defined instead of a password. When an address space requests a z/OS UNIX callable service for the first time, RACF does the following:

- Verifies that the user is defined as a z/OS UNIX user.
- Verifies that the user's current connect group is defined as a z/OS UNIX group.
- Initializes the control blocks needed for subsequent security checks.

This new support for a password phrase enables a user to enter a password phrase of 9-100 characters. The following commands and daemons are affected or enhanced:

passwd Use **passwd** to change user passwords or password phrases.

su Use **su** to change the user ID associated with a session.

OpenSSH IBM Ported Tools for z/OS - This ported tool automatically supports the new password phrases as it previously supported them, being an Open Source program.

rlogind The rlogind program is the server for the remote **rlogin** command commonly found on UNIX systems. It validates the remote login request and verifies the password or password phrase of the target user.

Refer to 18.4, “APPLID and password phrase support” on page 398 for more details of the C functions used internally.

18.3 Submit command

A new **submit** command, with z/OS V1R10, sends a batch job to the primary job entry subsystem (JES) for processing. The filename argument can either be a UNIX path name, data set, or input from standard input (stdin). The command uses the existing REXX submit() function. The format is:

```
submit [-jq] [filename ...]
```

Options:

-j Displays only the job ID when the **submit** command is successful. By default the output contains both the jobid and the filename.

-q Quiet mode. In quiet mode, all error messages are suppressed and the exit status is preserved.

filename For the filename you can specify one of the following:

- A UNIX pathname
- A sequential data set as "//'full.data.set.name'"
- A partitioned data set member as "//'full.pds.name(member)'"
- If no argument is given, **submit** reads from standard input.

Note: When standard input is being used, enter **///** or Ctrl-d to end the input.

18.3.1 Using the submit command

If the **submit** command is successful, the jobid of the submitted job and the filename are displayed as part of the output written to standard output, unless modified by the -j option. The JCL must have a job card. If a job card is not present, the **submit** command will fail.

submit command examples

The following examples show the use of the **submit** command from an OMVS shell session:

- To submit a job that resides in a z/OS UNIX file:

```
submit buildjcl.jcl

Output to standard output:

JOB JOB05990 submitted from path 'buildjcl.jcl'
```

- To submit a job that resides in a partitioned data set:

```
submit "//'POSIX.RTL.UT13.JCL(TESTJCL)'"

Output to standard output:

JOB JOB06075 submitted from data set 'POSIX.RTL.UT13.JCL(TESTJCL)'
```

- To submit a job from standard input, you can specify the following command:

```
cat test.jcl | submit

Output to standard output:

JOB JOB21532 submitted from stdin
```

- The following job was submitted without a proper HLQ in the data set name and the following error messages appeared:

```
ROGERS @ SC65:/u/rogers>submit "//'jcl.vers5(emaila)'"
IKJ56228I DATA SET JCL.VERS5 NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED
FSUMB411 submit: 'jcl.vers5(emaila)' Could not allocate data set.
```

18.3.2 submit command considerations

The following considerations are limitations when using the **submit** command:

- The JOBNAME can only be specified from within the JCL
- Only fully qualified data set names are accepted.
- This version of submit cannot be run directly from TSO.
- If multiple job cards are present in a file, submit only displays the job ID of the last job submitted within that file.
- If a PDS member is specified, and the data set exists but the member does not, it produces the following messages to standard output:

```
IRX0250E System abend code 013, reason code 00000024.
IRX0255E Abend in host command execio or address environment routine MVS.
IRX0670E EXECIO error while trying to GET or PUT a record.

submit then attempts to submit an empty job, and produces message FSUMB409
on stderr.
```

- Do not include ANSI control characters in data sets (for example, RECFM=FBA) that are submitted to JES; they will not be accepted.

Note: In the file system, /bin/submit contains the REXX code which has the same limitations as the REXX submit() function, especially that UNIX files have a line limit of 1024 characters.

18.3.3 Migration and coexistence considerations

The new **submit** command differs from the one available on “Tools & Toys”. It is recommended to modify any shell scripts that use the Tools & Toys version of **submit** and then remove the Tools & Toys version.

Attention: The Tools & Toys version of **submit** is not supported.

Only this official version of **submit** will be supported by IBM.

To see whether you are using the Tools & Toys version, simply add a question mark. Figure 18-10 shows an example with the new version.

```
$> submit \?
FSUMB407 submit: '?' Could not open file. errno=ENOENT: No such file,
directory, or IPC member exists
$>
```

Figure 18-10 Testing whether using the new submit version or not

Since the question mark is an option in the old version and a valid file name with the new one, the message indicates that the new version is used.

18.3.4 Using the submit command from TSO

In TSO, REXX procedures are often used to dynamically create jobs using the TSO command **submit** with operand ***** (asterisk). However, this interface has an old restriction.

Note: All characters in the job stream are converted to uppercase before being processed.

This is not useful for jobs running USS commands or dealing with files having mixed-case names. This problem is solved by using the UNIX **submit** command, as shown in Figure 18-11 on page 398.

```
/* REXX submjob */

Trace O
Parse Source . . myname .
Parse Upper Arg jobname
If jobname="" Then Do
  Say "Syntax:" myname "jobname"
  Exit 4
End

job.1 = "//"||Left(jobname,8) "JOB ,'Submit',NOTIFY=&SYSUID.,REGION=OM"
job.2 = "/*JOBPARM SYSAFF=SC70"
job.3 = "//OMVS     EXEC PGM=BPXBATCH"
job.4 = "//STDPARM  DD DATA,DLM=##"
job.5 = "SH echo This is a test job.; sleep 10"
job.6 = "##"
job.7 = "//STDOUT   DD SYSOUT=*,LRECL=137,RECFM=VB"
job.8 = "///"
job.0 = 8
Call Bpxwunix "/bin/submit -j", job., out., err.
If result<>0 Then Do
  Say "Rc("||result||")"
  Do i=1 To err.0
    Say err.i
  End
  Exit result
End
Else Do
  Say "Full jobid is:" jobname||"("||out.1||")"
  Exit 0
End
```

Figure 18-11 REXX sample submjob to submit jobs with mixed-case characters

In Figure 18-12 we show some examples run from TSO.

```
submjob *heri
 Rc(1)
 FSUMB408 submit: stdin Not accepted by JES.
submjob herijob
 Full jobid is: HERIJOB(JOB04134)
```

Figure 18-12 Examples using REXX submjob from TSO

In more sophisticated procedures you can submit jobs containing mixed-case characters and control the job while it is running.

18.4 APPLID and password phrase support

We provide an overview of the new and changed services that will allow the specification of an Application Identification (APPLID) and/or password phrases.

The use of PassTickets in conjunction with an APPLID and the use of password phrases provides customers with more secure authentication methods than are currently available.

18.4.1 Problems before getting the new support

There were several problems in the past:

- The use of PassTickets is severely limited on the **__passwd()**, **__login()**, and **pthread_security_np()** API functions.
- PassTickets generated with a user ID and APPLID cannot be used since there is no way to specify an application identification (APPLID) value on all these calls.
- Authentication could be done only with a password up to 8 characters long, but longer passwords improve system security.

18.4.2 New support

New C interfaces are provided for password, login, and pthread security functions that allow an APPLID to be specified:

- All UNIX system services that allow a password or PassTicket today will be modified to support password phrases. This includes pthread_security_np(), __passwrd(), and __login().
- New C functions are provided that are equivalent to the _pthread_security_np(), __passwrd(), and __login() services that will allow an APPLID value to be specified on the C function.
- Enhanced password, pthread security, and login functions accept password phrases up to 100 characters long.

  This provides increased system security and increased interoperability in heterogeneous environments.

Using APPLID support, installations can

- Create PassTickets that are created with specific user IDs and APPLIDs
- Specify an APPLID value on the passwd(), login(), and pthread_security_np() functions.

18.4.3 Details for using the new support

The new services provided are equivalent to old services with the added feature that they allow an APPLID to be specified, as shown in Table 18-1.

Table 18-1 New and existing C functions

New function	Existing function
pthread_security_applid_np()	pthread_security_np()
__passwd_applid()	__passwd()
__login_applid()	__login()

Assembler callable services

An application can use the z/OS UNIX Assembler callable services to specify the APPLID:

- BPX1TLS or BPX4TLS, being equivalent to pthread_security_np() or

- BPX1PWD or BPX4PWD, being equivalent to __passwd() or
- BPX1SEC or BPX4SEC to create a new security environment for a process

The caller must already be dubbed and in addition must update the BPXYTHLI, as follows:

- The ThliEP_FunctionCode must be set to constant ThliEP_ApplSet.
- The ThliEP_ApplIDLen must be set with the length of the APPLID.
- The ThliEP_Applid must be set with the APPLID value.

No error checking is done when specifying the values in the BPXYTHLI:

- If the ThliEP_FunctionCode is not set to ThliEP_ApplSet, then the APPLID is ignored.
- If the APPLIDLEN is not a value from 1-8, the APPLID is ignored.

Note: When you use the C interface to specify the APPLID, error checking on the length is done.

Using an APPLID in conjunction with a PassTicket

When using an APPLID in conjunction with a PassTicket, the RACF class PTKTDATA must be activated, as shown in Figure 18-13.

```
SETROPTS CLASSACT(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)
RDEFINE PTKTDATA MYAPPL SSIGNON(KEYMASKED(E001193519561977)) UACC(READ)
SETROPTS RACLIST (PTKTDATA) REFRESH
```

Figure 18-13 Setting up the RACF class PTKTDATA

The security administrator needs to generate a PassTicket using the MYAPPL APPLID and specifying a user ID. The PassTicket is valid for that user ID for one use.

Services for the password phrase support

The password phrase support is added to the following services:

- z/OS UNIX Assembler callable services
  - BPX1PWD and BPX4PWD
  - BPX1TLS and BPX4TLS
  - BPX1SEC and BPX4SEC
- C Functions
  - __passwd() and __passwd_applid()
  - pthread_security_np() and pthread_security_applid_np()
  - __login() and __login_applid()

When specifying a password phrase the number of parameters remains the same as before:

- Pass lengths 1-8 characters assumed to be password or PassTicket.
- Pass lengths 9-100 characters assumed to be a password phrase.

On using the passwd() or __passwd_applid()service the caller is not allowed to change the password or password phrase unless both the pass and new_pass parameters are the same type.

Note: RACF supports password phrases 9-13 characters in length with an installation exit installed. Without the exit installed, RACF only supports password phrase 14-100 characters in length.

18.4.4 Some examples

Example 1

Figure 18-14 shows three C function calls, made one after the other.

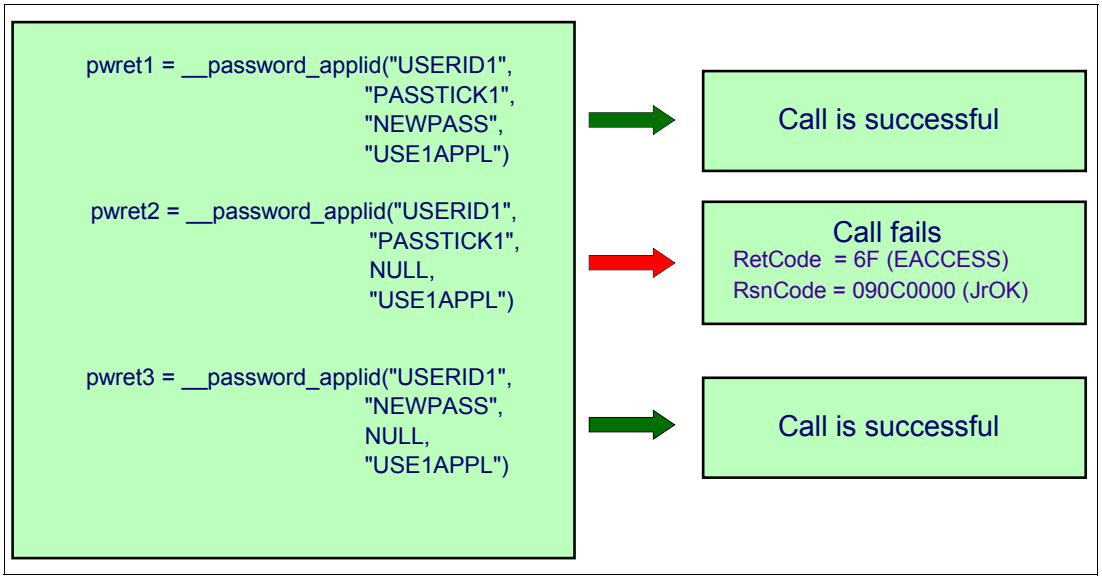


Figure 18-14 Example 1 with three C function calls

In the first call, the PASSTCK1 value is a PassTicket that was generated with the APPLID value USE1APPL. The passwd_applid call is successful.

Note: The PassTicket PASS1 cannot be used again. It is a one-time use.

The NEWPASS value is honored. The password is changed.

Note: If a new password phrase were specified, it would not be honored. The password phrase cannot be changed when specifying a PassTicket.

The second call attempts to use the PassTicket for the 2nd time. The call fails since PassTickets are a one-time use.

The third call attempts to use the NewPass value specified on the 1st call. The call is successful since the NewPass parameter was changed on the 1st call.

Example 2

Figure 18-15 shows two password APPLID calls with password phrases.

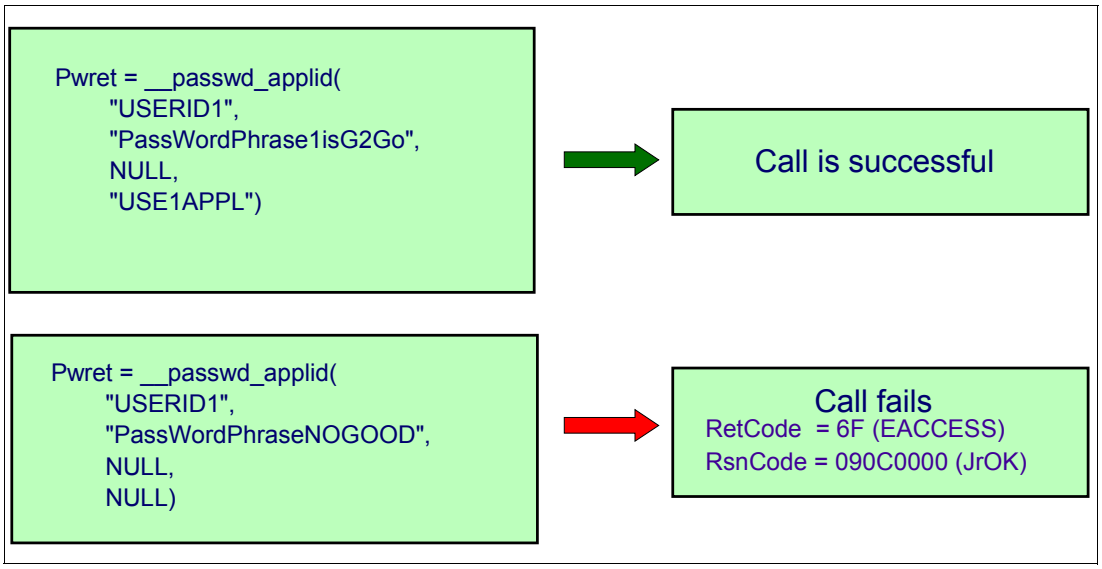


Figure 18-15 Example 2 with two password APPLID calls

The first call has a password phrase specified for the pass value. An APPLID value is also specified. The APPLID value is ignored when authenticating the password phrase. However, if the APPL class is active, then the user ID must have access to the APPLID specified, as shown in Figure 18-16.

```
PERMIT USE1APPL CLASS(APPL) ID(USERID1) ACCESS(READ)
```

Figure 18-16 Authorizing a user to an application ID

The second call has a password phrase that is not defined for the user. The call fails with a message written to the syslog:

```
ICH408I USER(USERID1  ) GROUP(SYS1    ) NAME(####################) LOGON/JOB
INITIATION - PASS PHRASE IS NOT VALID
```

Example 3

Figure 18-17 shows an example of a password APPLID call that intermixes a password phrase with a password specification.

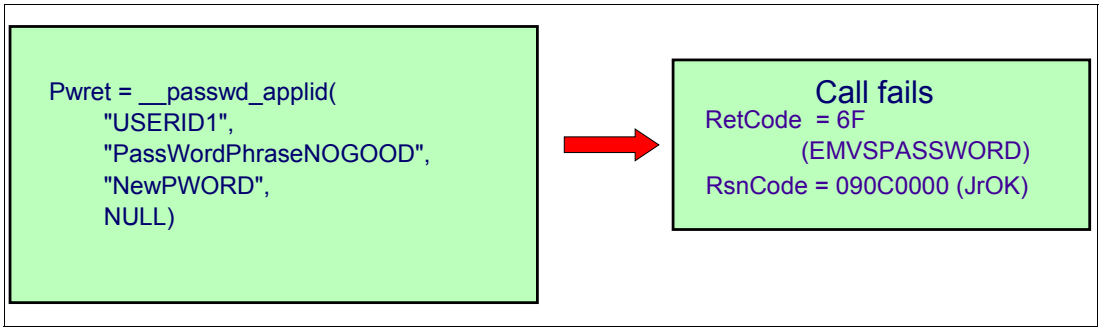


Figure 18-17 Example with a password APPLID call with a password phrase and a password

On this call a password phrase is passed as the Pass parameter and a password is passed for the New_Pass parameter.

The security product considers this an error with the specification of the new password.

18.5 Support loading from USS files into common storage

Here we provide an overview of a new z/OS UNIX System Services function that gives you the ability to load file system resident programs into common storage.

Currently, some multi-platform applications need to install executables into MVS libraries in order to be able to have the executables loaded into common storage. This complicates their install logic for z/OS over what is required for other platforms that these applications run on where the executables are all installed into a UNIX style directory.

A new extended loadhfs syscall allows authorized programs to load USS modules into common storage. This simplifies installation processing.

18.5.1 Usage of the loadhfs extended syscall

The new loadhfs extended (BPX1LDX,BPX4LDX) syscall does the following:

- Supports all functions of loadhfs (BPX1LOD,BPX4LOD).
- Adds a new option, Lod_Directed, that allows authorized callers to load a USS program into common.
- The Flags field parameter is validity checked such that anything other than the supported flag settings must be set to zero, otherwise this results in an EINVAL error return code.
- The parameters are identical to loadhfs (BPX1LOD/BPX4LOD).

Attention: There is no C/C++ run-time library interface available.

18.5.2 Using the Lod_Directed option

There are several considerations when using the Lod_Directed option.

- The caller must be APF-authorized or have PSW key 0-7 or Supervisor State.
- When this option flag is specified, the storage subpool is supplied as the last byte of the FLAGS parameter.

  Only subpool 241 is currently supported; any other subpool specified will result in an EINVAL error return code.
- The storage obtained for the target program is in key 0 storage.
- The shared library program attribute, st_Sharelib, is ignored.
- The target program must have an APF-authorized extended attribute set.

  The sticky bit for a file is ignored.
- The target program must be linked with the AC=1 attribute.

Note: Use the **extattr** command to set the APF-authorized attribute (+a option) for the target file system program. To be able to use the **extattr** command for the +a option, you must have at least read access to the BPX.FILEATTR.APF FACILITY class profile.

If the file is an external link, the request will fail with return code of EPERM. This means that the operation is not permitted. The reason code is JrExternalLink.

The returned parameter is a pointer to a 24-byte structure that contains the length of the loaded program storage, followed by the start address of the loaded program, followed by the entry point address of the loaded program. The returned structure is mapped in the BPXYCONS macro.

For a-mode 31 callers, the address of the 24-byte structure is returned in the Return_value operand. For a-mode 64 callers, the address of the 24-byte structure is returned in the Entry_point operand.

18.5.3 Responsibilities of the calling program

The calling program must save the returned program information after each call. The returned data structure can be reused for each syscall by a given task.

The caller needs to free the storage associated with the loaded program when the program is no longer needed. The storage should be freed using the returned storage length and program start address.

It is the responsibility of the caller to use the CSVDYLPA ADD BYADDR(YES) service to create a contents directory entry (CDE) in order to provide serviceability information for the loaded program. Without this, serviceability functions, such as SLIP LPAMOD and IPCS WHERE, will not be available for the loaded program.

When done with a directed load file system program and the program storage is to be released, this serviceability information should be removed.

Note: A program loaded with the Lod_Directed flag cannot be debugged using dbx.

Notes about CSVDYLPA

BYADDR=YES indicates that the modules have already been fetched and are linkedited as authorized (with AC=1). In this case, the MODINFO area must be present not only with the module names but also with the following LPMEA fields set for each module:

LpmeaEntryPointAddr This is the entry point address (with bit 0 of that address being on if the entry is to receive control in AMODE 31).

LpmeaLoadPointAddr This is the load point address (the start) of the primary segment.

LpmeaModlen This is the length of that load segment.

LpmeaLoadPointAddr2 If the load module represents a split RMODE load module, this is the load point address of the secondary segment; otherwise it is 0.

LpmeaModlen2 If the load module represents a split RMODE load module, this is the length of that secondary load segment; otherwise it is 0. All addresses must be in common storage, and the entry point

address must be within the primary segment. It is up to the caller to page fix and page protect the storage as needed.

The CDE can be removed with:

```
CSVDYLPA REQUEST=DELETE, TYPE=BYTOKEN, ...
```

Note: The token is returned from the ADD request.

For details, see the following documentation:

z/OS MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC), SA22-7607

For examples about BPX1LDX and BPX4LDX and the invocation of CSVDYNLPA to add and delete the CDE, see:

z/OS UNIX System Services Programming: Assembler Callable Services Reference, SA22-7803

18.6 Amblist utility UNIX interface

The MVS AMBLIST service aid prints formatted listings of modules to aid in problem diagnosis. AMBLIST can be used to provide listings showing:

- The attributes of program modules
- The contents of the various classes of data contained in a program module, including SYM records, IDR records, external symbols (ESD entries), text, relocation entries (RLD entries), and ADATA
- A module map or cross reference for a program module
- The aliases of a program module, including the attributes of the aliases

With z/OS v1R10, the amblist utility provides a UNIX interface to the AMBLIST program. With AMBLIST, you can obtain information about object modules and executable modules that reside in the file system, and diagnose problems with them. Output is written to stdout and errors to stderr.

Note: Output written to stdout goes to the terminal, and this kind of output is too much to be displayed on a terminal. Output should be directed to a file system file.

The amblist utility reads control statements from standard input (stdin). One or more control statements that identify the processing to be performed must be specified. Each control statement line must begin with one or more blanks. Keywords are case-sensitive and must be uppercase. Each control statement line can be up to 80 bytes long, but only the first 70 bytes can contain control information. Control statement lines longer than 70 bytes might be ignored or might cause an error to be reported. The OMVS z/OS UNIX command is:

```
amblist file....
```

Where:

file The file argument can be either a path name or a data set name. You cannot specify both a path name and a data set name at the same time. If you do, **amblist** ends with an error message and a nonzero return code.

If a path name is specified, it can be either a UNIX file or a UNIX directory. You can use only one path name at a time. If a UNIX directory is used for the path name, MEMBER must be specified on the amblist control statement to specify the file name.

If a data set name is specified, more data sets can be listed to indicate a concatenation of data sets to be searched. If a member name cannot be specified on the data set name (such as for LISTLOAD), it can either be specified on a control statement or omitted completely. If the member name is omitted, then all members are processed.

amblist examples

A control statement from a pipe, and the output redirected to a file, as follows:

```
echo " LISTLOAD" | amblist a.out > a.amblist
```

Control statement read interactively, output sent to terminal:

```
amblist hello.o
```

The user must then type " LISTOBJ" (leading blank, no quotes), then press Ctrl-d to end the amblist processing.

Control statement from a file, with output sent to terminal:

```
amblist hello.o < hello.ambctl
```

where the file hello.ambctl contains a single line " LISTOBJ" (leading blank, no quotes).

Control statement from pipe, process an object data set, output redirected to a file:

```
echo " LISTOBJ" | amblist "//binder.obj(hello)" > hello.amblist
```

contains the single line " LISTOBJ" (leading blank, no quotes).

18.7 Minor zFS updates

18.7.1 zFS man pages support

Previously zfsadm only had online help support via the option **-help** (or with **zfsadm apropos**) as shown in Figure 18-18.

```
$> zfsadm quiesce -help
IOEZ00243I Usage: zfsadm quiesce [{-aggregate <aggregate name> | -all}][-level]
[-help]
```

Figure 18-18 Using the zfsadm "-help" option

Now zfsadm supports the man command when you specify zfsadm and the subcommand as one word. This is shown in Figure 18-19 on page 407.

```
$> man zfsadmquiesce
zfsadm quiesce

  Purpose

  Specifies that an aggregate and all the file systems contained in it
  should be quiesced.


  Format

    zfsadm quiesce {-all | -aggregate name} [-level] [-help]


  Options
...
```

Figure 18-19 Using the new zfsadm man support

In addition, the zFS mount options and the utilities are supported:

```
man zfsmount
man ioeagfmt
man ioeagslv
```

18.7.2 zFS support of EAV volumes

As a result of DFSMS EAV support, zFS aggregates are supported in an EAV environment:

- zFS aggregates (and file systems) can reside on an extended address volume (EAV).
- zFS aggregates (and file systems) can reside in track-managed space or cylinder-managed space (and are subject to any limitations that DFSMS may have).
- zFS still has an architected limit of 4 TB for the maximum size of a zFS aggregate.

```
Old VSAM data set size limits:
65520 cylinders/volume x 90 blocks/cylinder x 8K bytes/block x 59 volumes =
2,850,088,550,400 bytes = 2654 GB = 2.59 TB

New VSAM data set size limits:
262668 cylinders/volume x 90 blocks/cylinder x 8k bytes/block x 59 volumes =
11,425,931,919,360 bytes = 10641 GB = 10.39 TB

New limits per volume:
193,659,863,040 bytes/volume = 180.359 GB/volume
```

Figure 18-20 VSAM data set limits based on 90 8K blocks per cylinder

19

z/OS UNIX-related applications

Network File System (NFS) is a distributed file system that enables users to access UNIX files and directories that are located on remote computers as if they were local. NFS is independent of machine types, operating systems, and network architectures.

The Distributed File Service Server Message Block (SMB) support provides a server that makes Hierarchical File System (HFS) files and data sets available to SMB clients. The data sets supported include sequential data sets (on DASD), partitioned data sets (PDS), partitioned data sets extended (PDSE), and Virtual Storage Access Method (VSAM) data sets.

This chapter explains the new information and functions available with z/OS V1R10 regarding the following z/OS UNIX System Services-related applications:

- Network Filesystem Services
- z/OS Distributed File Service SMB (Server Message Block)

19.1 Network File System

With z/OS V1R10, the following enhancements have been made to the Network File System (NFS):

- zLinux has been added to the list of tested and supported platforms with NFS. This support includes 64-bit addressing support for the NFS client utilities mvslogin, mvslogout, and showattr for non-z/OS platform usage. The keyword **-norpcbind** has been added to the **mvslogin**, **mvslogout**, and **showattr** commands.
- Japanese language National Language Support (NLS) for NFS Server console messages has been added.
- Displaying and modifying remote file system access control lists has been added.
- z/OS NFS file locking and access control has been added. The description of the secure attribute in the client attribute syntax is updated to include the krb5, krb5i, and krb5p values.
- Setting up reserved (privileged) ports describes how to reserve ports for use by the z/OS client.
- A **sockhang** operand is added to tell the server to create a dump when it detects a potential socket hang condition.
- Filtered NFS ctrace records in IPCS are updated to describe a new NFS Server Filtering Criteria panel a new section on processing traces in batch mode.
- Configuring a secure z/OS NFS client.
- NFS client hang problem analysis has been added to describe how to resolve NFS client hang situations.

The following function has been changed with z/OS V1R10 and NFS V4:

- The system server sample attribute table is updated to show the latest sample file that includes changes to NFS name mapping.

19.2 NFS support for zLinux on System z

Previously, z/OS NFS did not support NFS on Linux on System z. With z/OS V1R10, client utility updates now support the zLinux on System z with this new NFS implementation with the z/OS NFS Server. This allows interoperability between the NFS implementations of z/OS and zLinux. Previously, only the following foreign platforms were supported:

- Sun
- AIX
- Linux on Intel
- Windows

NFS protocol is an industry standard protocol. Therefore, the only changes needed for the support of z/OS NFS with NFS on Linux on System z was supporting the client enabling utilities on the Linux on System z clients, as explained here:

- mvslogin

  The **mvslogin** command is used to log in to z/OS from your workstation. The **mvslogin** command can be issued multiple times, and the last one command overrides the previous one. The **mvslogin** command is only required when accessing data on systems where the z/OS NFS server site security attribute is set to saf or safexp.

- mvslogout

  The **mvslogout** command is used to disconnect from the remote z/OS NFS server host. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

- showattr

  The **showattr** command is used to display the default attributes or the attributes that have been set for a specific mount point. If you specify a mount point, **showattr** shows the attributes for the mount point, including the overriding values.

Note: The z/OS NFS client utilities, including mvslogin, mvslogout, and showattr, are installed when the z/OS NFS client and TCP/IP are installed. The target library NFSCUTIL is a DDDEF to an existing z/OS UNIX directory (/usr/lpp/NFS/IBM). It will contain the client commands for the z/OS NFS client after installation. There is no need to port the z/OS NFS client utilities as you would for the remote NFS clients that use the z/OS NFS server.

Installation of the support

To create the executable commands **mvslogin**, **mvslogout**, and **showattr**, issue the following command for the environment that IBM supports to create the executables:

```
make zlinux
```

The NFS client enabling utilities need to be built with the following new keywords added to the makefile:

zLinux64 This specifies that the utilities are to be built for the Linux on IBM System z platform, exploiting 64-bit addressing.

zLinux This specifies that the utilities are to be built for the Linux on IBM System z platform using 32-bit addressing.

Note: To build zLinux, you *must* have a 32-bit build environment installed on your zLinux system. If a 32-bit environment is not available, see the NFS installation publication references in *z/OS Network File System (NFS) Guide and Reference*, SC26-7417, for information about building 64-bit versions of the client enabling commands.

19.2.1 Potential new usage

The new possibility of having a connection established between the NFS Client of z/OS V1R10 and the NFS Server running in zLinux is opening potentially interesting configurations such as the one depicted in Figure 19-1 on page 412.

- The NFS Client and NFS Server are connected through a HiperSocket between two LPARs (LPARz and LPARv) running in the same system (depicted in the top right corner of the figure).
- The NFS Server is running in a zLinux virtual machine VM1 provided by zVM running in LPARv.
- Through an FCP channel provided by the hardware to LPARv, the zLinux running in VM1 can have access to a SAN and its SCSI LUNs as controlled by N_Port ID Virtualization (NPIV), zoning, or LUN masking.
- As a consequence, user USS1 of LPARz (as depicted on the left side of the figure) can have access to the z/OS File System as provided by z/OS, complemented by a file system provided by the NFS client mounted on one of the directories that open up the access from z/OS to OPEN files present on SCSI disks of the SAN.

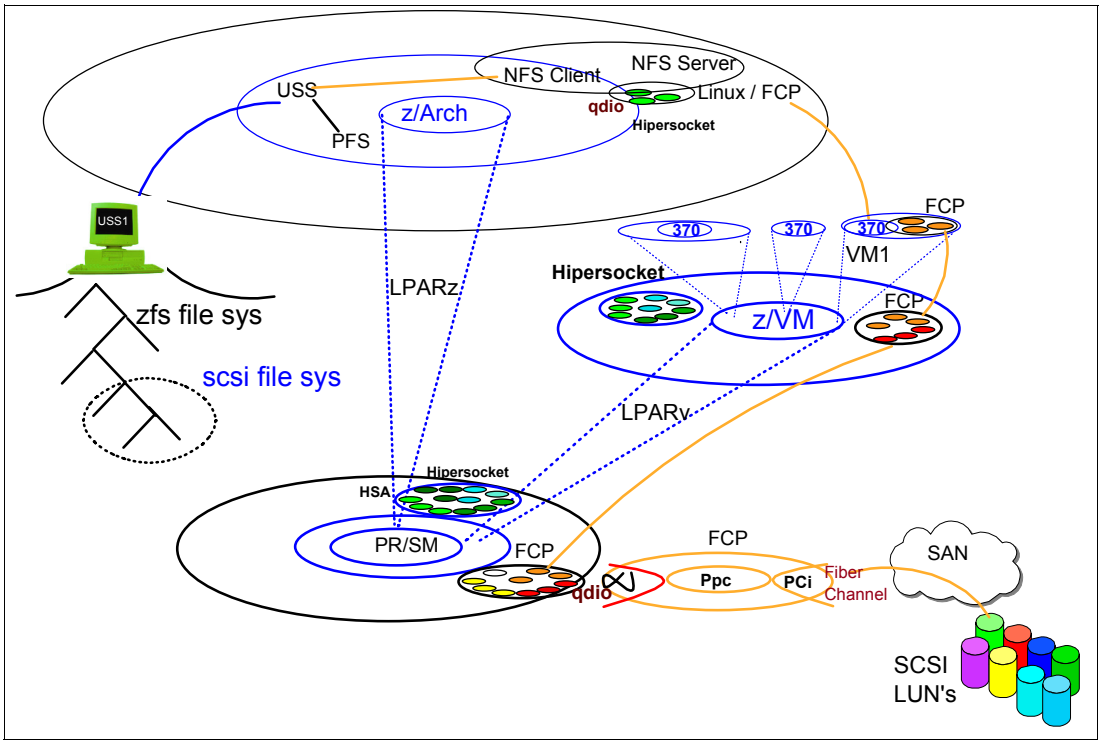


Figure 19-1 SCSI access from UNIX System Services

19.2.2 NFS client 64-bit support

Previously, the z/OS NFS client enabling utilities **mvslogin**, **mvslogout**, and **showattr** only supported 32-bit addressing mode. They did not support 64-bit addressing. Now, however, they are also 64-bit compatible. NFS client users and applications on 64-bit platforms, communicating with the z/OS NFS server, can now exploit the benefits of the 64-bit addressing architecture.

32-bit mode and 64-bit mode

IBM supports the compilation of the client enabling commands in 32-bit mode on all supported platforms, and in 64-bit mode on AIX, Sun, Linux, and z/Linux.

IBM has tested the client enabling commands in 32-bit mode and 64-bit mode on both 32-bit and 64-bit capable systems using:

- The standard gcc, or GNU Compiler collection, 4.0.x compiler command for Linux.
- The gcc 3.4.x compiler command for Sun.
- The gcc 4.0.x compiler command for AIX.
- Compiler support has been expanded to include Sun Studio 11 for Solaris™ 10 and XLC v8 for AIX 5.3.x.x.

makefile utility

The IBM-supplied makefile for the NFS client enabling commands provides the following keywords to enable the use of 64-bit addressing on various platforms:

AIX64 For the AIX platform

LINUX64 For the Linux platform

SUN64 For the Sun platform

ZLINUX64 For the z/Linux platform

The IBM-supplied makefile for the NFS client enabling commands also provides the following keywords to override some of the default values:

CC This allows the default compiler that is used by the makefile for the target platform to be overridden with a different compiler name. For example, “make cc=gcc sun” builds the utilities for the Sun Solaris platform using the gcc compiler.

CFLAGS This allows the default compiler options that are to be used for the target platform to be overridden with a different set of compiler options. The options string must be placed in double quotes, because the string can include any characters, including blanks.

Note: 64-bit mode is not currently supported for the z/OS client. IBM cannot test all possible compiler or option combinations. Any compilation or execution failures experienced when the default compiler or option values are overridden are the customer’s responsibility to resolve. The required system run-time libraries for 64-bit support must be available on the platform.

19.3 NFS server message globalization

Previously, the z/OS NFS server only supported the generation of console messages in English. With z/OS V1R10, new support provides the National Language Support for Japanese translation for its console messages.

19.4 NFS V4 access control list support

Some UNIX platforms have introduced additional security granularity by adding access control list (ACL) support to provide security specification on an individual user and group basis. An ACL is simply a list that specifies which users and groups get access to a file with what type of permission. The precise characteristics of this ACL support are platform-specific.

The NFS Version 4 protocol provides the ability to remotely manage ACLs by providing the ability to display and modify ACL values with the ACL attribute. The NFS v4 protocol has provided a very rich ACL definition with granularity beyond that provided by many platform ACL implementations. Therefore, it is necessary to map between the NFS ACL definition and the platform definition. The key is to ensure that in this mapping process, the mapping should err in the direction of more restricted access, not less. The new support provides:

- When the NFS server sets an ACL, it must be set at least as secure as specified by the NFS request.
- When an NFS server sends an ACL to an NFS client, the client must not perceive the file as more secure than it really is.

NFS version 4 ACLs have two types of ACLs: “Allow” and “Deny”, as follows:

Allow “Allow” indicates that the target user or group is being given the specified permission to access the file.

Deny "Deny" indicates that the target user or group is explicitly being denied the specified permission to access the file.

NFS can display and modify remote file system access control lists, if that the function is supported by the remote NFS server. This support is limited to z/OS UNIX access control lists. Access control list checking is controlled by the underlying file systems on the server systems, not by the z/OS NFS server or client.

19.5 NFS V4 file locking

With previous versions of NFS, the locking of a file on the z/OS NFS server is managed by Network Lock Manager (NLM) and Network Status Monitor (NSM). NLM and NSM are integrated into the z/OS NFS server to facilitate the expanded locking and serialization functions. Separate procedures for starting and stopping NLM and NSM were replaced in z/OS V1R7 by the server site attribute nlm|nonlm, which specifies their startup along with the NFS server. This integration also coordinates the locking function with stale file handle processing; when a file handle becomes stale, not only will the code clean up the file-related data as it does in prior releases, but it will also release any locks that remain held for that file.

NFS V4 enhancements

With the NFS version 4 protocol, the support for file locking is part of the NFS protocol. The file locking support is structured so that an RPC callback mechanism is not required. This is a departure from the previous versions of the NFS file locking protocol, Network Lock Manager. The state associated with file locks is maintained at the server under a lease-based model. The server defines a single lease period for all states held by an NFS client. If the client does not renew its lease within the defined period, all state associated with the client's lease may be released by the server. The client may renew its lease with use of the RENEW operation or implicitly by use of other operations (primarily READ).

Note: With the NFS version 4 protocol, a client user can choose to lock the entire file, or a byte range within a file. Byte-range locking is used to serialize activity to a range of bytes within a file. Byte-range locking is an advisory locking mechanism; that is, it does not prevent access to any application, but provides a mechanism for applications to communicate cooperatively through obtaining locks and querying if a lock is held.

Advisory locking

z/OS NFS supports only advisory locking. Advisory locking is when the operating system keeps track of which files have been locked by which process, but does not prevent a process from writing to a file that is locked by another process. This means that a process can ignore an advisory lock if the process has adequate permission.

Share reservations

Share reservations are a new concept in the NFS V4 protocol and are different than byte-range locking. Share reservations provide a method for controlling access to a file. When an OPEN request is sent for a file, the requester can indicate the type of access that should be denied to other requesters attempting to access the same file, which is NONE, READ, WRITE, or BOTH. Share reservations have an advantage over byte-range locking in that they provide a mandatory locking interface; any application that attempts to OPEN a file that is already opened and locked by another application is denied access. However, in these cases, access to files (and application processing) may be slowed if files are not shared.

Listing and releasing locks

To diagnose possible problems with conflicting locks, z/OS operators can issue a **listlock** command that displays all client programs and users that hold a lock on a file. The output messages include client and user ID, the lock ranges held, and lock status. The **listlock** command can be used for MVS data sets, PDS or PDSE members, or z/OS UNIX files.

You can use the **listlock** command to find locking information in cases where a lock is unavailable and the blocker is managed by another NFS server address space running on the z/OS system. To determine the identity of the blocker in this case, the **listlock** command should be issued on the system which owns the lock.

```
F mvsnfs,listlock='file_name'
```

New messages with NFS V4 for an MVS data set, PDS member, or PDSE member are issued as follows:

- If file_name references a z/OS conventional MVS data set for the first time, message GFSA921I is issued.
- If file_name references a PDS or a PDSE data set (that does not include the member name), GFSA918I is issued.
- If file_name references an invalid member name of a PDS or a PDSE data set, GFSA919I is issued.
- If file_name is found and there are no locks to list, message GFSA793I is issued.
- For a z/OS UNIX file, if there are no locks to list, message GFSA793I is issued.

To release all locks for a file, z/OS operators can issue a **release** command that releases locks for z/OS UNIX files and MVS data sets or members. The command also forces the NFS server to release the file, and if the file is active, to close and deallocate it.

Note: NFS V4 incorporates locking operation into the NFS protocol itself. This differs from NFS V2 and V3, which relied on Network Lock Manager (NLM) protocol.

Using NFS V4 Server V4 Locking

Using NFS V4 Server V4 Locking, you can serialize accesses to z/OS UNIX Files and MVS data sets for NFS Version 4 protocol.

Note: NFS uses UNIX System Services interfaces to provide the locking support for both z/OS UNIX files and MVS data sets. File accesses will only be serialized for cooperating applications. In a multi-server environment, the byte range locks are not coordinated for MVS data sets across different servers and sysplexes.

19.6 Reserving privileged ports

With z/OS V1R10, the z/OS NFS client uses a reserved (privileged) port to prevent the NFS server from rejecting a client request. The z/OS client attempts to use reserved port 1023. If that port is not available, the z/OS client will subtract 1 from 1023 until a reserve port is available. If no reserve ports are available, the z/OS client will issue error message GFSC724E.

The number of reserved ports the z/OS client can use is based on the client attribute biod. The number of reserved ports can be calculated from the following formula:

```
reserved ports = 8 + ( #biod * 4 )
```

TCPIP.PROFILE data set

The privileged ports should be reserved in the tcpip.profile file using the PORTRANGE statement. The default biod(6) and 8 additional ports correspond to 32 privileged ports that can be used by the z/OS client. For biod(6), the tcpip.profile file should include the following PORTRANGE statement:

```
PORTRANGE 991 32 UDP MVSNFSC
PORTRANGE 991 32 TCP MVSNFSC
```

This allows ports 991 through 1023 to be used by the z/OS client.

Note: MVSNFSC is the default z/OS NFS client startup procedure. Specify the correct z/OS NFS client startup procedure if it is different than the default.

Privileged UDP ports

When specifying secure(udp) or proto(udp), the z/OS client uses the privileged UDP ports to communicate with the NFS servers. When specifying proto(tcp) the z/OS client uses the privileged TCP ports to communicate the MOUNT RPC or UNMOUNT RPC with the NFS server. However, the z/OS client uses the ephemeral TCP ports to communicate NFS RPC with the NFS server. As a result, the z/OS client does not work with NFS servers that require all source TCP ports to be privileged.

19.7 Sockhang operand

With z/OS V1R10, when the z/OS NFS server starts up, a new operand, sockhang, has an initial setting of off. Set sockhang to on if you want the server to create a dump if it detects a potential socket hang condition. This optimizes the ability to detect the situation and capture the necessary diagnostic trace data before the trace data is lost.

Use this operand as follows:

- When sockhang is set to on, tracing levels Network, MVS (DFP_request and DFP_return), and Task_Flow (Dispatch, Resume, Schedule, and Suspend) are set on to ensure that the necessary diagnostic data is collected when a socket hang is first detected.
- When sockhang is subsequently set to off, tracing levels Network, MVS, and Task_Flow are reset to off; any previous settings of Network, MVS, and Task_Flow before sockhang was turned on must be reset as needed.

Detail level tracing provides the ideal setting to further isolate the source of a socket hang condition, but with a greater effect on system performance. Enter the sockhang operand as follows:

```
f mvsnsf, sockhang= on | off
```

Where:

```
sockhang=off - Will not create a dump if a socket hang is detected.
sockhang=on - Will create a dump if a socket hang is detected.
```

19.8 NFS ctrace filters

CTRACE (component trace) diagnostic information is difficult to analyze and in the specification of filtering, typing errors can easily occur. Furthermore, ctrace record processing ties up the TSO session.

With z/OS V1R10, filtering NFS trace records in IPCS is enhanced, as follows:

- New parameters have been introduced in ctrace filtering.

  The increased filtering granularity facilitates data reduction of displayed data. The data analysis has been made user-friendly and efficient by the panel interface and the batch processing support.
- A panel interface has been provided to specify the filtering criteria, as shown in Figure 19-2 on page 418.
- Batch processing support has been provided for ctrace filtering.

The z/OS NFS ctrace function allows you to restrict the trace records to be processed by IPCS and displayed on the trace screen. Use the CTRACE DISPLAY PARAMETERS panel, shown in Figure 19-2 on page 418, to do so.

Use the Component field to specify whether you want trace records for the NFS server or NFS client. Specify the name of the server or client as it appears in your start procedure. The example shown in Figure 36 on page 278 uses MVSNFSC as the name of the NFS client.

Use the Options field in Figure 19-2 on page 418 to specify any of the following:

- Nothing, in which case all records will be processed and displayed.
- A specific list of record types to be processed and displayed. In this case, you can use shorthand record specifications (for example, CALL, TASK_FLOW), as well as the full-length record type names. All other record types will be filtered out and not displayed. Any invalid option values are ignored. If no valid options are specified, no trace records are displayed. Figure 19-2 on page 418 shows an example of selecting only the BUFFER and USS (that is, USS_REQUEST and USS_RETURN) record types.
- The value "PANEL" to display a filtering criteria panel, which provides another method of specifying options. If the Component field specifies the name of the NFS server, then the NFS Server Filtering Criteria panel (shown in Figure 19-3 on page 419) is displayed. If the Component field specifies the name of the NFS client, then the NFS Client Filtering Criteria panel (shown in Figure 19-4 on page 420) is displayed.

```
-------------------- CTRACE DISPLAY PARAMETERS -------------------------------
COMMAND ===>
  System     ===>            (System name or blank)
  Component  ===> MVSNFSC    (Component name (required))
  Subnames   ===>

  GMT/LOCAL ===> G                          (G or L, GMT is default)
  Start time ===>                           (mm/dd/yy,hh:mm:ss.dddddd or
  Stop time ===>                             mm/dd/yy,hh.mm.ss.dddddd)
  Limit ===> 0               Exception ===>
  Report type ===> FULL      (SHort, SUmmary, Full, Tally
  User exit ===>             (Exit program name)
  Override source ===>
  Options         ===> USS,BUFFER

  To enter/verify required values, type any character
  Entry IDs ===>   Jobnames ===>   ASIDs ===> OPTIONS ===>   SUBS ===>

  CTRACE COMP(MVSNFSC) FULL OPTIONS((REQ,BUF))


  ENTER = update CTRACE definition. END/PF3 = return to previous panel.
  S = start CTRACE. R = reset all fields.
```

Figure 19-2 CTRACE Display Parameters panel

PANEL value

The existing z/OS NFS Server Ctrace function supports record filtering based on the NFS Server Ctrace record types. These values must be specified in a comma-separated list format on the CTrace Options menu under the Options parameter. A simple typographical error in the specification of the values can invalidate the filtering process.To avoid this problem in the future, the panel interface has been added to select the desired filtering criteria. This panel is selected by specifying **PANEL** as the options parameter. It is shown in Figure 19-3 on page 419.

Any combination of record types can be selected on the **OPTIONS** and **SUPEROPTIONS** sections of the menu, as shown in Figure 19-3 on page 419. Within those record types, further selection filtering can be performed, based on the following criteria. If multiple criteria are specified, records matching any of the record type options and all of the other specified criteria are considered to be a match and will be selected.

Note: You can invoke Help panels by pressing F1.

Any of the secondary options listed can be specified on the IPCS Ctrace Display Parameters panel **OPTIONS** list if the panel is not used:

MODNAME 1– 8 character name of the server load module.

TASKNAME 1– 8 character name of the server task.

FUNCNAME 1 – 16 character name of a specific server function.

PREQBLK 1 – 8 hexadecimal address of a specific request block.

REQID 1 – 8 hexadecimal digit representing a specific request.

OPERATION 1 – 22 character name of the NFS V2/V3 request, or V4 operation.

```
----------------------   NFS Server Filtering Criteria   ---- ENTER NEW VALUES
 COMMAND ===>
-------------------------------------------------------------------------------
              O P T I O N S                |      S U P E R O P T I O N S
-------------------------------------------------------------------------------
 _ FFDC         | _ DFP_RETURN   | _ INFO    | _ CALL (ENTRY, EXIT)
 _ ENTRY        | _ CB_MGMT      | _ WARNING | _ TASK_FLOW (SUSPEND, RESUME,
 _ EXIT         | _ NETWORK      | _ ERROR   |              SCHEDULE, DISPATCH)
 _ SUSPEND      | _ GENERAL      | _ DEBUG1  | _ USS  (USS_REQUEST, USS_RETURN)
 _ RESUME       | _ DETAIL       | _ DEBUG2  | _ MVS  (DFP_REQUEST, DFP_RETURN)
 _ SCHEDULE     | _ TRAP         | _ DEBUG3  | _ LOCK (LOCK_REQUEST, LOCK_RESUME,
 _ DISPATCH     | _ BUFFER       | _ DEBUG4  |         LOCK_RELEASE)
 _ USS_REQUEST  | _ LOCK_REQUEST | _ DEBUG5  | _ DEBUG7 (DEBUG 1, 5, 6)
 _ USS_RETURN   | _ LOCK_RESUME  | _ DEBUG6  | _ DEBUG9 (ALL OF DEBUG RECORDS)
 _ DFP_REQUEST  | _ LOCK_RELEASE | _ DEBUG8  |
-------------------------------------------------------------------------------
 MODNAME  => ________      TASKNAME  => _________  FUNCNAME  => _________________
 PREQBLK  => ________      REQID     => _________  OPERATION => ________
-------------------------------------------------------------------------------
 Batch processing  => N (Y/N)
-------------------------------------------------------------------------------
   ENTER = update Filtering Criteria definition.
   END/PF3 = return to previous panel.  S = start.   R = reset all fields.
```

Figure 19-3 NFS server filtering criteria panel

Specifying NFS client filtering options using the panel interface

The existing z/OS NFS Client Ctrace function supports record filtering based on the NFS Client Ctrace record types. These values must be specified in a comma-separated list format on the Ctrace Options menu under the Options parameter. A simple typographical error in the specification of the values can invalidate the filtering process.

To avoid this problem in the future, select the desired filtering criteria on the panel shown in Figure 19-4 on page 420.

Any of the secondary options listed in Figure 19-4 on page 420 can be specified on the IPCS Ctrace Display Parameters panel **OPTIONS** list if the panel is not used.

MODNAME 1 – 8 character name of the client load module.

FUNCNAME 1 – 16 character name of a specific client function.

VNOP 1 – 8 character name of the vnode/vfs operation, like vn_rdwr.

REQID 1 – 8 hexadecimal digit number representing a specific request.

PREQBLK 1 – 8 hexadecimal digit number representing the address of a specific request block.

PID 1 – 8 hexadecimal digit number representing the process id of the Client process.

DIRRNODE 1 – 8 hexadecimal digit number representing the address of a specific parent directory rnode.

MNTINFO 1 – 8 hexadecimal digit number representing the address of a specific mount info block.

RNODE 1 – 8 hexadecimal digit number representing the address of a specific rnode associated.

For **BUFFER** record types, several additional selection criteria are available. These criteria are ignored if no buffer records are selected:

BNNUM 1 – 16 hexadecimal digit number representing a specific buffer.

BNFLAG 3-bit flag field associated with the BUFFER trace records.

BNPTR 1 – 8 hexadecimal digit number representing the address of a specific buffer.

```
----------------------   NFS Client Filtering Criteria   ---- ENTER NEW VALUES
 COMMAND ===>
 ------------------------------------------------------------------------------
                 O P T I O N S                |    S U P E R O P T I O N S
 ------------------------------------------------------------------------------
 _ FFDC         | _ NFS_REQUEST | _ CB_MGMT    | _ CALL (ENTRY, EXIT)
 _ ENTRY        | _ NFS_RETURN  | _ BUFFER     | _ TASK_FLOW (SUSPEND, RESUME,
 _ EXIT         | _ USS_REQUEST | _ NETWORK    |            SCHEDULE, DISPATCH)
 _ SUSPEND      | _ USS_RETURN  | _ GENERAL    | _ USS  (USS_REQUEST,USS_RETURN)
 _ RESUME       | _ LOCK_REQUEST| _ DETAIL     | _ NFS  (NFS_REQUEST,NFS_RETURN)
 _ SCHEDULE     | _ LOCK_RESUME | _ TRAP       | _ LOCK (LOCK_REQUEST,
 _ DISPATCH     | _ LOCK_RELEASE| _ MSG        |       LOCK_RESUME,LOCK_RELEASE)
                |               |              |
------------------------------------------------------------------------------
 MODNAME  => ________     PID       => _________  FUNCNAME  => ________________
 PREQBLK   => ________    REQID     => _________  VNOP      => ________
 RNODE    => ________     DIRRNODE  => _________  MNTINFO   => ________
------------------------------------------------------------------------------
                      Buffer Filtering Criteria
 BNPTR    => ________     BNFLAG    => ___       BNNUM     => ________________
------------------------------------------------------------------------------
 Batch processing  => N (Y/N)
------------------------------------------------------------------------------

   ENTER = update Filtering Criteria definition.
   END/PF3 = return to previous panel.  S = start.   R = reset all fields.
```

Figure 19-4 NFS client filtering criteria panel

Note: You can invoke Help panels by pressing F1.

Processing traces in batch mode

You can process traces in batch mode by running IPCS under the TSO/E terminal monitor program (TMP) in a batch job. Because it can take a long time to process all of the trace records in a large trace file, using batch mode to filter the records and produce a readable output file can be helpful to avoid tying up the debugger's TSO session the entire time.

This support can be invoked by using one of the two ways:

- Submit a batch job.
  - A sample batch job file GFSNBSMP is available in SYS1.NFSSAMP.

- Specify `Y` in the batch processing field on the NFS Filtering Criteria panel in Figure 19-4 on page 420 to build the JCL. Then issue the **`SUBMIT`** command on the editor command line.

New messages

The new messages shown in Table 19-1 may be generated.

Table 19-1 New messages

ID	Message	Accompanying text
GFSNM000	Enter New Options	Enter new values of parameters for NFS Filtering Criteria
GFSNM001	No Options Selected	No NFS Filtering Criteria options were selected
GFSNM002	Invalid Command	Enter "R" or "S" or press PF3
GFSNM003	Empty Command Line	Enter "R" or "S" or press PF3
GFSNM004	Return Code 0	Filtering Options String Formed

19.9 NFS client RPCSEC_GSS support

To enable the z/OS NFS client to support the RPCSEC_GSS authentication flavor using the Kerberos V5 Security Mechanism, perform these tasks:

- Add the client principal, MVSNFSC, to the Kerberos database on the KDC, with a defined password. This principal should *not* use randkey as the password. For example, for the z/OS NDBM type or Sun KDC, issue the following command in the kadmin interface:

  ```
  addprinc mvsnfsc
  ```

  Then enter the desired password in the prompt. For the z/OS SAF type KDC, the password can be defined in the PASSWORD field of the RACF ADDUSER or ALTUSER commands.

- Map the principal, MVSNFSC, to a RACF user. For example:

  ```
  AU PZNFSC OWNER(IBMUSER) OMVS(UID(500))
  ALTUSER PZNFSC PASSWORD(password) NOEXPIRED KERB(KERBNAME(mvsnfsc))
  ```

- From the omvs shell, the system administrator must add the principal, MVSNFSC, into the keytab. If /etc/skrb/krb5.keytab does not exist, create a new one. For example:

  ```
  IBMUSER:/SYSTEM/etc/skrb(127):>
  keytab add mvsnfsc -p password -k /etc/skrb/krb5.keytab
  ```

RPCSEC_GSS security

RPCSEC_GSS security is based on a security mechanism-specific principal name, and it employs Generic Security Services (GSS) APIs. The z/OS NFS client did not support RPCSEC_GSS authentication. Only the System Authentication was supported. This did not protect NFS data across network transmissions.

The z/OS NFS client now supports the RPCSEC_GSS authentication using the Kerberos Security mechanism. The benefit is that NFS data is cryptographically protected across network transmissions.

Using NFS client RPCSEC_GSS support

Using NFS Client RPCSEC_GSS support, you can have NFS mount points be securely mounted with the following pseudo authentication types:

krb5 This is the Kerberos V5 integrity protection on RPC credentials.

krb5i This the Kerberos V5 protection on RPC credentials and data.

krb5p This the Kerberos V5 Integrity protection on RPC credentials and privacy protection on RPC data.

Note: Data is protected across network transmission because client-generated checksums are verified (for krb5 and krb5i) at the server's end by using GSS APIs. Additionally, for krb5p, the data is transmitted in encrypted form.

Cryptographic verification of the RPC credentials helps to validate the identity being claimed by the client user, and therefore facilitates stronger user authentication.

The support is invoked during mount by specifying krb5 or krb5i or krb5p in the secure keyword of the mount command, as shown in Figure 19-5 on page 422.

```
mount filesystem(nfs01) type(nfs) mountpoint('/u/nfsdir1')
parm('aix6000:/home/user,secure(krb5)')
```

Figure 19-5 Example of a mount using the secure keyword

In this example, `aix6000` is the name of the host AIX NFS server and `/home/user` is the name of the remote mount point.

19.10 NFS client hang problem analysis

NFS client hang situations can arise from many different causes. Detailed analysis of the situation is required to determine the source. Some of the possible causes could be:

- Slow server or underlying file system response
- Server failure
- Network failure
- Socket hang

When a client hang occurs, the general z/OS UNIX **`netstat command`**, TSO **`netstat`** command, or z/OS shell **`onetstat`** command can be used to determine whether this could be a socket hang problem.

If the **`netstat`** command is issued on the failing client system and it shows that a very large number of TCP sockets are in CLOSEWAIT state with the destination IP address of the z/OS NFS server, it indicates that sockets are probably hung (z/OS NFS server may not accept new TCPIP connections). In this case, it is very probable that the diagnostic trace data recorded by the z/OS NFS Server is already lost by the time that the hang is recognized at the client. The only thing that can be done to resolve the immediate situation is to restart the z/OS NFS server.

After the z/OS NFS server is restarted, we recommend using the **`sockhan`**g operand to help capture the necessary diagnostic data before it is lost the next time that this situation occurs. This command tells the server to monitor the server's sockets for potential hang conditions

and to create a dump when a hang is suspected so that the diagnostic trace data can be captured before it is lost.

For detailed information about the **sockhang** operand, see “Sockhang operand” on page 416. For other possible causes of a client hang, standard problem analysis techniques should be used.

19.11 NFS V4 name mapping

The z/OS NFS server did not support mapping between owner and group names in a DNS domain and POSIX uid/gid values. The POSIX model of uid/gid was used for user identification. This worked well in closed environments, where UIDs and GIDs are globally managed. In a large networked environment, however, the POSIX model presented a maintenance problem for keeping UIDs and GIDs globally unique.

The NFS server now supports mapping between owner and group names for a DNS domain and the z/OS UNIX System Services uid and gid values.

Note: This allows the owner and group names to be administered on a DNS domain basis and not globally, which facilitates local systems being able to use their own, local owner and group identification.

Using NFS V4 name mapping

Using NFS V4 name mapping, you can:

- Map owner and group names on a DNS domain to z/OS UNIX uid and gid values.
- Override the server’s default local domain with the site attribute NFSV4DOMAIN(*domain*).

Note: This **NFSV4DOMAIN** attribute has also been added to the z/OS NFS client.

Owner and group names are used for file ownership attribute specification and security ACL definitions. Note the following rules:

- Any mixed case names are converted to upper case.
- Only RACF-supported names are allowed.
- The names cannot be longer than 8 characters.
- The names are case-insensitive.

Installing NFS V4 name mapping

The owner/group names must be defined to RACF with the appropriate uid and gid values.

The NFSV4DOMAIN site attribute should be appropriately set if the default DNS domain is to be overridden.

Note: Name resolution is not supported via any global name server such as LDAP.

19.12 z/OS Distributed File Service SMB

With z/OS V1R10, the following enhancements are available with DFS SMB support:

- SMB now performs automatic syntax checking of /opt/dfslocal/home/dfskern/envar.
- The DFSKERN level of dispatching priority has a very small change: during dfskern initialization, a small file (opt/dfslocal/var/dfs/rfsfile) is created (or updated) to keep track of RFS file identifiers. This file should not be changed or erased.
- SMB supports hang detection with the following environment variables:

```
_IOE_HANG_DETECT_DUMP_LIMIT – _IOE_HANG_DETECT_INTERVAL –
_IOE_HANG_DETECT_OENODE_TIMEOUT –
_IOE_HANG_DETECT_THREAD_TIMEOUT
```

- New information is provided about resolving Insufficient space errors.
- There is changed information for Windows SMB clients that make SMB calls directly over TCP using port 445.

19.12.1 Environment variable syntax checking

DFS/SMB has very limited syntax checking of environment variables. If a customer mistypes the name of the environment variable, it is ignored. If the value is wrong or out of range, a message may not be written to the operator console. If the envar file has other syntax errors, SMB may abend.

A new z/OS UNIX **dfssyntax** command has been added to allow checking the syntax of the environment variables before starting DFS/SMB.

Benefit: You can now check the environment variable file before starting SMB and can remove any potential environment variable syntax errors.

Using the new command

Using envar syntax checking, it is possible to:

- Use the new OMVS command, **dfssyntax,** to perform syntax checking on the environment variable file.
- The messages are written to stdout and will include the line number where the syntax mistake was found and what the problem was; for example:
  - A blank line, wrong envar name, wrong value for envar, trailing blanks

When DFS/SMB is started, syntax checking is automatically run and it indicates if there were syntax errors.

New messages

There are 20 new messages dealing with the various syntax errors and they are documented in *z/OS DFS/SMB/zFS Messages and Codes*, SC24-5917. Following are two examples:

```
IOEW16149E Line line of environment variable file file is incorrect. The line is
too long. The maximum line length is maxlen.

IOEW16158I The value value for environment variable envar found in file file
should be in upper case.
```

19.12.2 Hang detection environment variables

The following hang detection environment variables for DFSKERN have these definitions:

- _IOE_HANG_DETECT_DUMP_LIMIT

This specifies the maximum number of dumps for the hang detector to take.

Default value: 5

Expected value: A number greater than or equal to 5.

Example: _IOE_HANG_DETECT_DUMP_LIMIT=5

- _IOE_HANG_DETECT_INTERVAL

  This specifies the interval, in seconds, for which hangs are checked.

  Default value: 10

  Expected value: A number greater than or equal to zero (0).

  Example: _IOE_HANG_DETECT_INTERVAL=10

- _IOE_HANG_DETECT_OENODE_TIMEOUT

  This specifies the period of time, in seconds, that an oenode must be hung before it is detected. When zero (0) is specified, the hang detect function is off.

  Default value: 0

  Expected value: A number greater than or equal to zero (0).

  Example: IOE_HANG_DETECT_OENODE_TIMEOUT=30

- _IOE_HANG_DETECT_THREAD_TIMEOUT

  This specifies the period of time, in seconds,that a thread must wait before it is detected as a hung thread. When zero (0) is specified, the hang detect function is off.

  Default value: 0

  Expected value: A number greater than or equal to zero (0).

  Example: _IOE_HANG_DETECT_THREAD_TIMEOUT=50

19.12.3 Support for using port 445

DFS/SMB listens on TCP port 139. This requires enabling NETBT (NetBios over TCP/IP) in the workstations. If you disable NETBT, then the Windows client will try to contact port 445. This port is not used today by the z/OS DFS/SMB server.

The supported Windows SMB clients make SMB calls directly over TCP or through NETBIOS over TCP/IP. When connecting directly over TCP, the client makes calls using a TCP/IP connection to server port 445. When using NETBIOS over TCP/IP, the client makes calls to ports 137, 138, and 139. When connecting to the server, these clients will attempt to connect to ports 445 and 139 to establish a session.

In z/OS V1R10, support for TCP direct mode (SMB command flow without NETBIOS layer) has been added.

Note: This support implements anx and fulfills the marketing requirement MR0324044821, DFS use of CIFS port 445.

Using the new support, you can:

- Configure the SMB server to respond on port 139 (Netbios over TCP/IP)
- On port 445 (SMB directly over TCP/IP)
- Use both methods when attempting to establish a session with the client.

Note: This allows you to configure the SMB clients *not* to use NETBT (NetBios over TCP/IP) and run with NETBT disabled.

Using the support

To use the support, configure the **_IOE_SMB_TRANSPORTS** environment variable. This envar specifies the enable modes in three different ways:

BOTH This is the default and means SMB is listening on ports 445 and 139.

DIRECT This means SMB is listening on port 445.

NETBIOS This means SMB is listening on port 139.

Many SMB clients have configuration options to enable NETBIOS over TCP/IP, SMB directly over TCP/IP, or both. The server responds on the enabled ports; the client software can chose to attempt one protocol prior to the other, or both in parallel.

20

JES2 enhancements

This chapter discusses the JES2 enhancements in z/OS V1R10. These enhancements in focus mainly on improving the availability of JES2. The following topics are discussed:

- Support for dynamic load and delete of installation exits
- Automatic recovery of NJE device and connection errors

The enhancements are meant to reduce the number of IPLs and JES2 hot starts required to recover from problems with installation exits and NJE devices.

In addition, JES2 in z/OS V1R10 supports new subsystem interface (SSI) function requests for:

- SSI 70 - Scheduler JCL facility
- SSI 80 - Extended status

The new request types are intended to simplify installation and maintenance of applications that manage jobs and SYSOUT data sets, such as SDSF. By using the new SSI requests, applications are provided with a standard mechanism to access more JES2 data, instead of depending on JES2 data areas. This simplifies migration to a new JES2 release and reduces the work needed to support future JES2 releases. For more details on the new SSI requests see *z/OS MVS Using the Subsystem Interface*, SA22-7642.

Note: z/OS V1R10 JES2 can run in the same MAS environment with the z/OS V1R9, z/OS V1R8 and z/OS V1R7 versions of JES2. The coexistence APAR is OA20935.

20.1 Dynamic installation exits

Historically, when you needed to change a JES2 installation exit, you could do so and then hot start JES2 to pick up the new installation exit. Starting in z/OS V1R7, JES2 introduced new installation exits that execute in the user environment. These installation exits are loaded into common storage, such as LPA or CSA. For example, JES2 installation exits number 52, 53, and 54 execute in the user environment and have to be loaded into common storage. The **STORAGE=** parameter of the **LOADMOD** JES2 initialization statement controls the type of storage that the installation exit is loaded into.

However, the **LOADMOD** initialization statement is unable to reload an installation exit that is already loaded into common storage. This limitation makes it impossible to make changes to some JES2 installation exits by performing a hot start. Thus, performing an IPL became the only way to reload an installation exit into common storage.

z/OS V1R10 support

In z/OS V1R10, JES2 introduces the concept of *dynamic installation exits*. The new support allows the installation to dynamically add, delete, and refresh copies of JES2 installation exits using JES2 operator commands. The dynamic exits support is provided both for exits that are loaded into JES2 private storage and for exits loaded into common storage. In addition, JES2 operator commands are provided to dynamically add or remove routines associated with an installation exit.

The dynamic exits support is rolled back to z/OS V1R9 and z/OS V1R8 using APAR OA21346. However, the APAR support for this function is slightly different than the support in z/OS V1R10. Throughout this chapter, whenever there are differences between the support in z/OS V1R10 and z/OS V1R9 and z/OS V1R8, the support is provided by the APARs in z/OS V1R9 and z/OS V1R8. .

20.1.1 $ADD LOADMOD command

The **$ADD LOADMOD** operator command loads a new copy of the specified load module into storage. Storage type can be JES2 private storage, LPA, or CSA. The full syntax of the **$ADD LOADMOD** command is shown in Figure 20-1 on page 428.

```
$ADD LOADMOD(modname),STORAGE=PVT|LPA|CSA
```

Figure 20-1 $ADD LOADMOD command syntax

Load modules loaded using this operator command must be compiled using z/OS V1R10 JES2 macros. Otherwise, the module is not loaded and an error message is issued. For example, the message displayed in Figure 20-2 on page 428 was issued after we tried to dynamically load module HASPX44A, which was compiled on a z/OS V1R9 system using z/OS V1R9 JES2 macros.

```
$ADD LOADMOD(HASPX44A),STORAGE=PVT
$HASP003 RC=(34),ADD 308
$HASP003 RC=(34),ADD LOADMOD(HASPX44A)  - HASPX44A -
$HASP003             MODULE VERSION IS INVALID,
$HASP003             EXPECTED-z/OS1.10 ACTUAL-z/OS 1.9
```

Figure 20-2 $ADD LOADMOD RC=34 error message

We then recompiled our HASPX44A exit under a z/OS V1R10 system and using z/OS V1R10 JES2 macros. The **$ADD LOADMOD** command was then successful, as shown in Figure 20-3 on page 429.

```
$ADD LOADMOD(HASPX44A),STORAGE=PVT
$HASP819 LOADMOD(HASPX44A) 330
$HASP819 LOADMOD(HASPX44A)  ADDRESS=23D47D00,
$HASP819                    LENGTH=000200,
$HASP819                    LOADTIME=(2008.119,
$HASP819                    16:48:57),RMODE=ANY,
$HASP819                    SPLEVEL=CHECK,STORAGE=PVT
```

Figure 20-3 $ADD LOADMOD success message

Note that the **$ADD LOADMOD** command has no effect on exit routines. JES2 does not attempt to associate routines in the load module with any routines associated on an **EXIT** initialization statement or a previous **$T EXIT** command. This allows for a function to be split over multiple load modules, which can then be activated in its entirety after all the modules are loaded.

20.1.2 $DEL LOADMOD command

Use the **$DEL LOADMOD** command to logically remove a load module from JES2 storage. The command supports the removal of load modules from JES2 private and from common storage. After the module is logically deleted, any exit routines associated with it are nullified. The load module is physically deleted from storage only after JES2 determines that there are no users of the load module.

Note: The APAR version of the **$DEL LOADMOD** command does not allow removing load modules from JES2 private storage.

Figure 20-4 on page 429 shows an example of the **$DEL LOADMOD** command. It first displays the routines associated with exit number 44. The exit load module is then removed from storage. Finally, another **$D EXIT** command is issued to display the nullified exit routines.

```
$D EXIT(44)
$HASP823 EXIT(44) 448
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINES=(HASX44SA,HASX44SB),
$HASP823            SPLEVEL=CHECK,TRACE=YES,USECOUNT=0
$DEL LOADMOD(HASPX44A)
$HASP819 LOADMOD(HASPX44A) 411
$HASP819 LOADMOD(HASPX44A)  ADDRESS=23D47D00,
$HASP819                    LOADTIME=(2008.120,
$HASP819                    10:00:34),STORAGE=PVT -
$HASP819                    ELEMENT DELETED
$D EXIT(44)
$HASP823 EXIT(44) 413
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINES=(HASX44SA(NULL),
$HASP823            HASX44SB(NULL)),SPLEVEL=CHECK,
$HASP823            TRACE=YES,USECOUNT=0
```

Figure 20-4 $DEL LOADMOD command example

Note: There may be cases where JES2 cannot determine whether anyone is still using the load module. In these cases, the load module is not physically deleted from storage.

In general, the **$DEL LOADMOD** command is not intended to be regularly used in a production environment. Instead, it is used as a mechanism to allow continued operations in case of an error with an installation exit.

20.1.3 $T LOADMOD REFRESH command

Use the **$T LOADMOD REFRESH** command to bring a new copy of a load module into storage and remove the old copy. This command provides a major benefit over performing a **$DEL LOADMOD** and then a **$ADD LOADMOD** in that it preserves exit routines associated with the load module. The old exit routine addresses are replaced with matching addresses from the new load module location. Addresses of dynamic tables in the load module are also preserved.

The refresh process is performed in such a way that routine addresses are never nullified. If a routine does not exist in the new copy of the load module, its address is nullified. If the new copy of the load module contains new routines that are were not associated with an exit, the routines are not automatically associated with any exit. After the refresh is complete, the removal of the old load module proceeds in the same manner as a **$DEL LOADMOD** command.

The example in Figure 20-5 on page 430 shows two routines in load module HASPX44A, associated with exit 44. The load module is then refreshed and the new copy of the load module is located at a different address in storage. The routines associated with the exit remain active and point to the correct addresses.

```
$D EXIT(44),ROUTINES
$HASP823 EXIT(44)   ROUTINES=(HASX44SA,HASX44SB)
$D LOADMOD(HASPX44A)
$HASP819 LOADMOD(HASPX44A) 507
$HASP819 LOADMOD(HASPX44A)  ADDRESS=23D47D00,
$HASP819                    LENGTH=000200,
$HASP819                    LOADTIME=(2008.120,
$HASP819                    10:08:04),RMODE=ANY,
$HASP819                    SPLEVEL=CHECK,STORAGE=PVT
$T LOADMOD(HASPX44A),REFRESH
$HASP819 LOADMOD(HASPX44A) 509
$HASP819 LOADMOD(HASPX44A)  ADDRESS=23D7DB00,
$HASP819                    LENGTH=000200,
$HASP819                    LOADTIME=(2008.120,
$HASP819                    10:35:27),RMODE=ANY,
$HASP819                    SPLEVEL=CHECK,STORAGE=PVT
$D EXIT(44),ROUTINES
$HASP823 EXIT(44)   ROUTINES=(HASX44SA,HASX44SB)
```

Figure 20-5 $T LOADMOD REFRESH command example

20.1.4 $D LOADMOD and $D EXIT commands

The **$D LOADMOD** command is enhanced to include the date and time that the load module was loaded into storage. An example of the new command output is shown in Figure 20-5 on page 430.

The **$D EXIT** command is enhanced to include a **LONG** parameter. When specified with the new parameter, JES2 lists all the routines associated with an exit in a long format, which also includes the address of each routing and the name of the load module it is located in. An example is shown in Figure 20-6 on page 431.

```
$D EXIT(44),LONG
$HASP823 EXIT(44)
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINE(1)=(NAME=HASX44SA,
$HASP823            USECOUNT=0,ADDRESS=23D47D50,
$HASP823            LOADMOD=HASPX44A),
$HASP823            ROUTINE(2)=(NAME=HASX44SB,
$HASP823            USECOUNT=0,ADDRESS=23D47DE8,
$HASP823            LOADMOD=HASPX44A),SPLEVEL=CHECK,
$HASP823            TRACE=YES,USECOUNT=0
```

Figure 20-6 $D EXIT(xxx),LONG command example

A **(NULL)** suffix is added to a routine name if there is no routine address associated with the routine. JES2 skips these routines during exit processing and does not cause an abend.

20.1.5 $T EXIT command

The **$T EXIT** command is enhanced to support a new **ROUTINES=** parameter. Using the new parameter, it is possible to dynamically change the routines associated with an exit. The **ROUTINES=** parameter accepts a list of routines to associate with the exit, but it also supports simply adding or removing a list of specific routines from an exit routines list. This avoids having to specify the entire list of routines when only some of the routines must be added or removed.

To add a routine to an existing list of routines, issue the **$T EXIT(xxx),ROUTINES=** command and specify a plus (+) sign in front of the routine name. To remove a routine from an existing list of routines, specify a minus (-) sign in front of the routine name. For example, Figure 20-7 illustrates how to use the command to add routine HASX44SB to exit number 44.

```
$T EXIT(44),ROUTINES=(+HASX44SB)
```

Figure 20-7 Dynamically add a routine using the $T EXIT command

Figure 20-8 illustrates how to use the command to remove routine HASX44SB from exit 44.

```
$T EXIT(44),ROUTINES=(-HASX44SB)
```

Figure 20-8 Dynamically remove a routine using the $T EXIT command

Note that the plus (+) and minus (-) signs apply also to subsequent routines in the list that are not prefixed with a plus or minus sign. Requests to add or remove a routine are processed from left to right. Examples of plus and minus sign usage are listed in Table 20-1 on page 432.

Table 20-1 $T EXIT(xxx),ROUTINES= command processing

$T EXIT(xxx),ROUTINES= command	Routines after command is issued
$T EXIT(xxx),ROUTINES=(A,B,C)	A,B,C
$T EXIT(xxx),ROUTINES=(+D,E)	A,B,C,D,E
$T EXIT(xxx),ROUTINES=(-C,D)	A,B,E
$T EXIT(xxx),ROUTINES=(+A)	A,B,E,A
$T EXIT(xxx),ROUTINES=(-A)	B,E,A
$T EXIT(xxx),ROUTINES=(-A,B,E)	null

When **$T EXIT** is used with a plus (+) sign to add routines to an exit, only the routines specified on the command go through address resolution processing. No other routines associated with the exit are affected by the command.

20.1.6 Special behavior when loading routines into LPA

The dynamic exits support does not change the **LOADMOD** initialization statements processing of JES2. In particular, if a module is not located in the link list concatenation or in a STEPLIB data set in the JES2 procedure, then the LPA is searched for the module. If the module is loaded into LPA, then it is presumed that the module lives in LPA and a subsequent **$T LOADMOD REFRESH** command only searches LPA for that module.

In the same way, when a **$ADD LOADMOD** command is issued with the **STORAGE=LPA** parameter, JES2 only searches the LPA for the module. If the module is loaded from the LPA, any subsequent calls to the **$T LOADMOD REFRESH** command only searches for the module in the LPA. In addition, as long as the load module is located in the LPA, JES2 ignores **$ADD LOADMOD** commands with the **STORAGE=PVT** parameter for the load module.

JES2 does not dynamically load a load module into the LPA and does not physically delete a load module from the LPA. You must use the **SETPROG LPA** command to accomplish that. Figure 20-9 on page 433 shows the process of dynamically loading a JES2 load module to the LPA and then removing it. As illustrated in the figure, the following steps are performed:

- Try to load the load module from a STEPLIB data set to the LPA. This request is rejected by JES2.
- Dynamically add the load module to the LPA using a **SETPROG LPA** command.
- The **$ADD LOADMOD** request is then honored by JES2.
- Associate two of its routines with exit 44 and enable the exit.
- Refresh the load module to show that it remains located at the same address in the LPA.
- Dynamically remove the load module from the LPA.
- Issue a **$D EXIT** and **$D LOADMOD** to show that JES2 also removed the load module.
- Clean up the null exit routines from exit 44 using a **$T EXIT(xxx),ROUTINES=** command.

Note: JES2 uses an LPA exit to listen for dynamic deletes of load modules in the LPA. The JES2 LPA exit receives control *after* the load module has been deleted from storage. Therefore, there could be a short period of time where a load module has been deleted from LPA, but not logically deleted from JES2 yet.

If the load module is accessed by JES2 during that period of time, an abend occurs. This processing allows you to recover from such an erroneous situation without an IPL.

```
$ADD LOADMOD(HASPX44A),STORAGE=LPA
$HASP003 RC=(31),ADD
$HASP003 RC=(31),ADD LOADMOD(HASPX44A)  - HASPX44A -
$HASP003            MODULE COULD NOT BE LOADED
SETPROG LPA ADD MODNAME=HASPX44A DSN=SYS1.SHASLNKE
CSV551I 10.48.49 LPA ADD 541
SUCCESSFUL: 1  UNSUCCESSFUL: 0  NOT PROCESSED: 0
MODULE    RESULT
HASPX44A  SUCCESSFUL
$ADD LOADMOD(HASPX44A),STORAGE=LPA
$HASP819 LOADMOD(HASPX44A)
$HASP819 LOADMOD(HASPX44A)  ADDRESS=217E3000,
$HASP819                    LENGTH=000200,
$HASP819                    LOADTIME=(2008.120,
$HASP819                    10:49:04),RMODE=ANY,
$HASP819                    SPLEVEL=CHECK,STORAGE=LPA
$T EXIT(44),ROUTINES=(HASX44SA,HASX44SB),ENABLE
$HASP823 EXIT(44)
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINES=(HASX44SA,HASX44SB),
$HASP823            SPLEVEL=CHECK,TRACE=YES,USECOUNT=0
$T LOADMOD(HASPX44A),REFRESH
$HASP819 LOADMOD(HASPX44A)
$HASP819 LOADMOD(HASPX44A)  ADDRESS=217E3000,
$HASP819                    LENGTH=000200,
$HASP819                    LOADTIME=(2008.120,
$HASP819                    10:52:06),RMODE=ANY,
$HASP819                    SPLEVEL=CHECK,STORAGE=LPA
SETPROG LPA DELETE MODNAME=HASPX44A FORCE=YES
CSV551I 10.57.56 LPA DELETE 576
SUCCESSFUL: 1  UNSUCCESSFUL: 0  NOT PROCESSED: 0
MODULE    RESULT
HASPX44A  SUCCESSFUL
$D EXIT(44)
$HASP823 EXIT(44)
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINES=(HASX44SA(NULL),
$HASP823            HASX44SB(NULL)),SPLEVEL=CHECK,
$HASP823            TRACE=YES,USECOUNT=0
$D LOADMOD(HASPX44A)
$HASP003 RC=(52),D
$HASP003 RC=(52),D LOADMOD(HASPX44A)  - NO SELECTABLE
$HASP003           ENTRIES FOUND MATCHING SPECIFICATION
$T EXIT(44),ROUTINES=(-HASX44SA,HASX44SB)
$HASP823 EXIT(44)
$HASP823 EXIT(44)   STATUS=ENABLED,ENVIRON=JES2,
$HASP823            ROUTINES=(),SPLEVEL=CHECK,TRACE=YES,
$HASP823            USECOUNT=0
```

Figure 20-9 $ADD LOADMOD STORAGE=LPA command example

20.1.7 Programming dynamic exits

To enable an exit for dynamic exits support, add a DYNAMIC=YES parameter on the $MODULE macro of the exit. The default for this parameter is DYNAMIC=NO. The reason for this is because changes may be required for an existing exit to support dynamic loading and removal. Therefore, we recommend that you review your exits before changing to DYNAMIC=YES. Pay attention to the following issues:

- Structures or code in your installation exit that is pointed by JES2 data areas

  JES2 can handle dynamic tables and routine addresses. The rest should be handled by the exit.

- Data structures that JES2 processes during initialization, but not again when the load module is removed or refreshed

  These include PCEs and DTEs that are created during initialization time, $WSTABs that JES2 processes once and then saves the address of the tables in $DCTs, and installation $BERTTABs. If you have enabled exit 24 in your installation, this may be a good indication for such data structures.

- Other code that performs initialization and may not handle correctly a situation where it is refreshed or loaded at a different storage address

$$$$LOAD and $$$$DEL routines

The $$$$LOAD and $$$$DEL routines are regular $ENTRY routines in the exit load module. They are provided to allow the installation exit writer with a way to customize what happens when the module is loaded or removed from storage. The $$$$LOAD routine is called after JES2 completes its processing to set up the load module. Dynamic tables and routine addresses are already processed at that point. The $$$$DEL routine is called before JES2 starts the actual delete process of a load module. Dynamic tables and routine addresses are not yet updated at that point.

The parameter list passed to $$$$LOAD and $$$$DEL is the $CSVLIST. It contains the reason for the call, such as operator command or initialization statement, as well as the $LMT address for the load module. Both routines receive control in the JES2 environment, as part of the JES2 process that loads or deletes load modules. Because the routines may receive control under initialization processing and not only for dynamic actions processing, it is your responsibility to code them using the appropriate services only. For example, do not use the $WAIT service, because it is prohibited during initialization processing.

For a **`$T LOADMOD REFRESH`** command, JES2 processing is as follows:

1. Load the new copy of the load module, but do not process dynamic tables or routine addresses yet.
2. Call the $$$$DEL routine for the old copy of the load module.
3. Perform processing to replace the old module with the new.
4. Call the $$$$LOAD routine for the new module.

During refresh processing, both the old and the new $LMTs are passed to the $$$$LOAD and $$$$DEL routines.

$$$$LOAD and $$$$DEL are not intended to block the dynamic loading or removal of a load module. $$$$LOAD is not required to return a return code. It is recommended to return zero (0) in R15, because this requirement may change in the future.

However, $$$$DEL may return the following return codes:

0 This code indicates to proceed normally with the removal of the load module from storage. $$$$DEL will not be called again.

4 This code indicates that additional processing is needed. JES2 logically deletes the load module but continues to check if there are any active users of the load module. After there are no active users, JES2 calls the $$$$DEL routine again, checking for permission to physically delete the module.

8 This code indicates to JES2 to not physically delete this load module. $$$$DEL is not called again. (Keep in mind that preventing module deletion increases storage usage.)

There are three exceptions for return code 8, regarding modules that reside in CSA or LPA, where JES2 will call $$$$DEL again:

- During JES2 normal termination - in this case all JES2 modules are being deleted, so $$$$DEL is called again.
- During a JES2 hot start - this is to inform $$$$DEL that JES2 is restarting in case it is now okay to delete the load module.
- For an LPA load module - when that module is deleted from LPA.

Blocking dynamic actions for an exit

As described, there may be many issues that can prevent you from enabling dynamic actions for a certain installation exit. To block dynamic actions for an exit, use these methods:

- Code the DYNAMIC=NO parameter on the $MODULE macro, or take the default.
- Code a $$$$LOAD routine that sets the LMT2NDYN bit in the $LMT passed to it. This prevents future dynamic actions and only allows the module to be dynamically loaded once using the **$ADD LOADMOD** command.

20.1.8 Migration considerations

This section discusses the issues to keep in mind when you migrate to JES2 on z/OS V1R10.

New RACF profiles

The new JES2 operator commands in support of dynamic exits support are protected by RACF profiles in the OPERCMDS class. Table 20-2 on page 435 lists the profile names and access level required for each operator command.

Table 20-2 RACF profiles for new JES2 operator commands

JES2 operator command	RACF profile name	Required access level
$ADD LOADMOD	JES2.ADD.LOADMOD	CONTROL
$DEL LOADMOD	JES2.DEL.LOADMOD	CONTROL
$T LOADMOD REFERSH	JES2.MODIFY.LOADMOD	CONTROL
$T EXIT(xxx),ROUTINES=	JES2.MODIFY.EXIT	CONTROL

Secondary command processor

A second COMM PCE is added with the dynamic exits support. The secondary command processor is used for all LOADMOD, EXIT and MODULE commands. The main command processor acts as a router that routes these commands to the secondary processor.

The reason for a secondary command processor is that module load actions require I/O operations, and this usually involves a wait. Such a wait can impact processing of other commands. By moving the load actions to a separate command processor, the impact on other JES2 command processing is reduced.

Exit number 5, the JES2 command processor exit, is called twice for commands processed by the secondary command processor. The first time is under the main command processor and the second time is under the secondary command processor.

Note: In the APAR version of the dynamic exits support, exit 5 is only called under the main command processor.

$XIT and $XRT changes

$XIT is a 256-element array that represents all possible JES2 exits. $XRTs are pointed to by the $XIT. Each $XRT is an array in which each entry represents a routine associated with a JES2 exit.

In z/OS V1R10 and on prior releases with APAR OA21346 applied, the $XRT begins with a header that describes the list, followed by an array of XRTEs, each representing an exit routine entry point. The following $XIT fields are moved to the $XRT header in z/OS V1R10:

- XITBSPL, the SPLEVEL check bypassed flag, is now XRTBSPL in XRTFLAGS.
- XIT#RTNS, the number of routines associated with this exit, is now XRTCOUNT.
- XITUSCNT, the exit use count, is now XRTUSCNT.

There can be a chain of $XRTs pointed by a $XIT. In that case, the second and later $XRTs represent logically deleted exit routines. However, if all the routine names associated with an exit are deleted, then the first $XRT on the chain may be logically deleted. Logically deleted $XRTs are physically deleted when the use count goes to zero (0).

$LMT changes

The $LMT is used to track modules loaded by the JES2 $MODLOAD service, which includes modules loaded via the LOADMOD initialization statement and the **$ADD LOADMOD** command. Code that searches for $LMTs must consider the state of the $LMT before examining fields in the $LMT. In prior releases, the only state that needed to be examined was invalid/free $LMT, represented by bit LMT1INVD.

In z/OS V1R10, JES2 introduces two new $LMT states that should be considered when searching $LMTs:

- Logically deleted $LMT, represented by bit LMT2DELT
- Force freed $LMT, represented by bit LMT3FREE

20.2 NJE network monitor

Another availability enhancement in z/OS V1R10 JES2 is the NJE network monitor. The network monitor provides the ability to automatically restart JES2 Network Job Entry (NJE) devices and connections. This ability may simplify the automation required by the installation in order to start NJE devices and connections after an IPL, JES2 hot start, or a network problem, and increase the availability of the JES2 NJE network. The network monitor support includes updates to JES2 initialization statements and operator commands, which are discussed in detail in this chapter.

The network monitor support provides a means to define a restart interval on every type of NJE device and connection. If a device or a connection fails, JES2 automatically restarts it on the next interval. In addition, new operator commands are available to display, reset, start, purge, or halt all network devices using a single operator command. The new operator commands can prove very useful when you must stop JES2 or recycle VTAM.

20.2.1 Setting restart interval defaults for connections

The NJEDEF initialization statement is used to define the NJE characteristics of the current JES2 node. The NODE initialization statement is used to define the NJE characteristics of other nodes in the NJE network. Both initialization statements are updated to support a new CONNECT= parameter.

The purpose of the CONNECT= parameter is to define whether JES2 should automatically restart connections with the specified node if they are disconnected longer than a specified interval. The CONNECT= parameter is specified as follows:

CONNECT=(YES,interval) This indicates connections with this node are automatically restarted if disconnected longer than the specified interval. The interval is specified in minutes. The maximum interval is 1440 minutes (1 day).

CONNECT=NO This globally turns off the network monitor for the node. It can be overridden on the APPL, SOCKET, and LINE initialization statements.

Figure 20-10 shows an example of coding the NJEDEF and NODE initialization statements with the CONNECT= parameter, specifying a 30-minute restart interval.

```
NJEDEF   CONNECT=(YES,30),
         DELAY=300,
         HDRBUF=(LIMIT=100,WARN=80),
         JRNUM=1,          JTNUM=1,
         SRNUM=7,          STNUM=7,
         LINENUM=40,       MAILMSG=YES,
         MAXHOP=0,         NODENUM=999,
         OWNNODE=1,        PATH=1,
         RESTMAX=0,
         RESTNODE=100,     RESTTOL=0,
         TIMETOL=0

N2       NAME=WTSCMXA,PATHMGR=YES,SUBNET=WTSCNET,CONNECT=(YES,30)
```

Figure 20-10 Using the CONNECT= parameter on the NJEDEF and NODE initialization statements

The **$D NJEDEF**, **$T NJEDEF**, **$D NODE** and **$T NODE** operator commands are enhanced to display and update the CONNECT= parameter. See 20.3.1, “Operator commands” on page 441 for more details.

It is possible for an automatic restart to fail for different reasons. In such cases, JES2 issues message $HASP569. The message also indicates when will JES2 automatically try to restart again. For example:

```
$HASP569 NODE=N2 – NO IDLE SNA LINE IS AVAILABLE, RC=05 NEXT ATTEMPT: 19 MAR 2008
AT 11:45
```

20.2.2 Automatic restarting of connections

The APPL initialization statement is used to specify the characteristics of a VTAM application to JES2. The SOCKET initialization statement specifies a TCP/IP address and port number that is to be listened on by NETSRV. The LINE initialization statement specifies the characteristics of one teleprocessing line to be used during remote or network job entry. These initialization statements define a connection to the JES2 node, using different communication protocols.

These statements are updated to support a CONNECT= parameter. Using the CONNECT= parameter on the APPL, SOCKET, and LINE initialization statements allows you to override the default specified on the node level. The CONNECT= parameter is specified as follows:

CONNECT=(YES, interval) This indicates connections with this node are automatically restarted if disconnected longer than the specified interval, regardless of the definition on the NJEDEF or NODE initialization statement. The interval is specified in minutes. The maximum interval is 1440 minutes (1 day). An interval of 0 indicates to use the value from NJEDEF.

CONNECT=NO This indicates no restarts are performed for this connection, regardless of the definition on the NJEDEF or NODE initialization statement.

CONNECT=DEFAULT This indicates restart is the same as the definition on the NODE initialization statement. This is the default.

Note: When CONNECT=(YES,0) is specified or when the default CONNECT= value is taken on a LINE initialization statement, the NODE= parameter must also be specified. Otherwise, JES2 does not know which NODE definition specifies the default for the LINE.

Figure 20-11 shows an example of the APPL, SOCKET, and LINE initialization statements with the CONNECT= parameter.

```
APPL(SC64NJE)    NODE=1,CONNECT=NO
SOCKET(WTSCPLX1) NODE=6,IPADDR=9.12.6.84,CONNECT=(YES,5)
LINE12           UNIT=TCP,CONNECT=(YES,15),NODE=7
```

Figure 20-11 Using the CONNECT= parameter on the APPL SOCKET and LINE initialization statements

The **$D APPL**, **$T APPL**, **$D SOCKET**, **$T SOCKET**, **$D LINE** and **$T LINE** operator commands are enhanced to display and update the CONNECT= parameter. See 20.3.1, “Operator commands” on page 441 for more details.

20.2.3 Automatic restarting of devices

The LOGON initialization statement identifies JES2 as an application program to VTAM. The NETSERV initialization statement defines a network server that is to be used for NJE TCP/IP communication. The LINE initialization statement specifies the characteristics of one teleprocessing line to be used during remote or network job entry. These initialization statements are updated with two new parameters, START= and RESTART=.

The START= parameter is specified as follows:

START=YES This indicates the device is to be automatically started at initialization. This parameter already existed for the NETSERV initialization statement prior to z/OS V1R10. This is especially useful after a JES2 hot start.

START=NO This indicates the device is not to be automatically started at initialization. This is the default.

The RESTART= parameter is specified as follows:

RESTART=(YES,interval) This indicates the device should be restarted if drained for longer than the specified interval. The interval is specified in minutes. The maximum interval is 1440 minutes (1 day). An interval of 0 indicates to use the value from NJEDEF.

RESTART=NO This indicates the device is not automatically restarted.

Figure 20-12 on page 439 shows an example of the LOGON, NETSERV, and LINE initialization statements with START= and RESTART= parameters.

```
LOGON(1)   APPLID=&SYSNAME.NJE,RESTART=(YES,30)
NETSERV(1) SOCKET=WTSCPLX2,START=YES,RESTART=(YES,10)
LINE12     UNIT=TCP,CONNECT=(YES,15),NODE=7,START=YES
```

Figure 20-12 Using the START= and RESTART= parameters on the LOGON, NETSERV, and LINE initialization statements

As a result of adding the NODE= parameter to the LINE initialization statement, it is now possible to use the **$S N,LNExxxxx** command on SNA and TCP/IP lines.

NETSERV automatic restart example

Figure 20-13 displays an example of NETSERV automatic restart, as follows:

- Using the **$T NETSRV** command to define a 5-minute restart interval for NETSRV1.
- At 16:03:20, NETSRV1 was drained for some reason.
- At 16:08:20, 5 minutes later, JES2 automatically restarts NETSRV1.

```
16:00:38.80 PELEG    00000210  $T NETSRV1,RESTART=(YES,5)
...
16:03:20.67          00000010  $HASP097 NETSRV1    IS DRAINED
...
16:08:20.66          00000010  $HASP568 NETSRV1 - AUTOMATIC RESTART
16:08:20.70          80000010  IEF403I IEESYSAS - STARTED - TIME=16.08.20 - ASID=0074 - SC70
16:08:20.71          00000010  IAZ0542I NETSRV1 IAZNJTCP for HBB7750 compiled Mar 22 2008 16:07:33
16:08:20.73          00000010  IAZ0511I NETSRV1 Server Port number could not be resolved, DEFAULT
                               assumed: 175
16:08:20.73          00000010  $HASP5091 NETSRV1 IS ACTIVE
16:08:20.73          00000010  IAZ0537I NETSRV1 NJETCP SERVER WAITING FOR WORK
```

Figure 20-13 Automatic restart of NETSERV example

20.3 Collision avoidance

Two nodes may specify the same restart interval. It is then possible for the two nodes to initiate a sign-on on the other node simultaneously, which would result in a sign-on failure due to the collision of sign-on bits or the fact that the number of connections is exceeded. The two nodes would then automatically retry again at the same time, and fail again. From this point, the collision might repeat endlessly. The situation of repeated collisions due to automatic restarts is illustrated in Figure 20-14 on page 440.

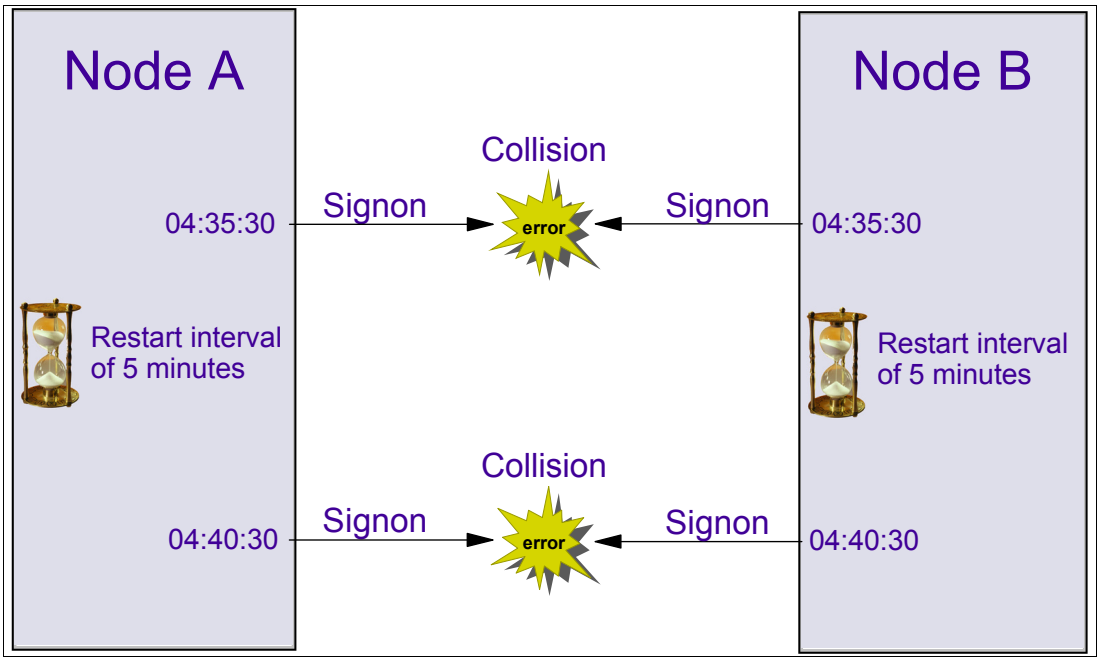


Figure 20-14 Repeated collisions due to automatic restart

To avoid this situation, JES2 adds a *random factor* of up to 45 seconds to each interval. Even with the random factor, collisions may still occur, but are unlikely to occur repeatedly. Collision avoidance using a random factor is illustrated in Figure 20-15 on page 441.

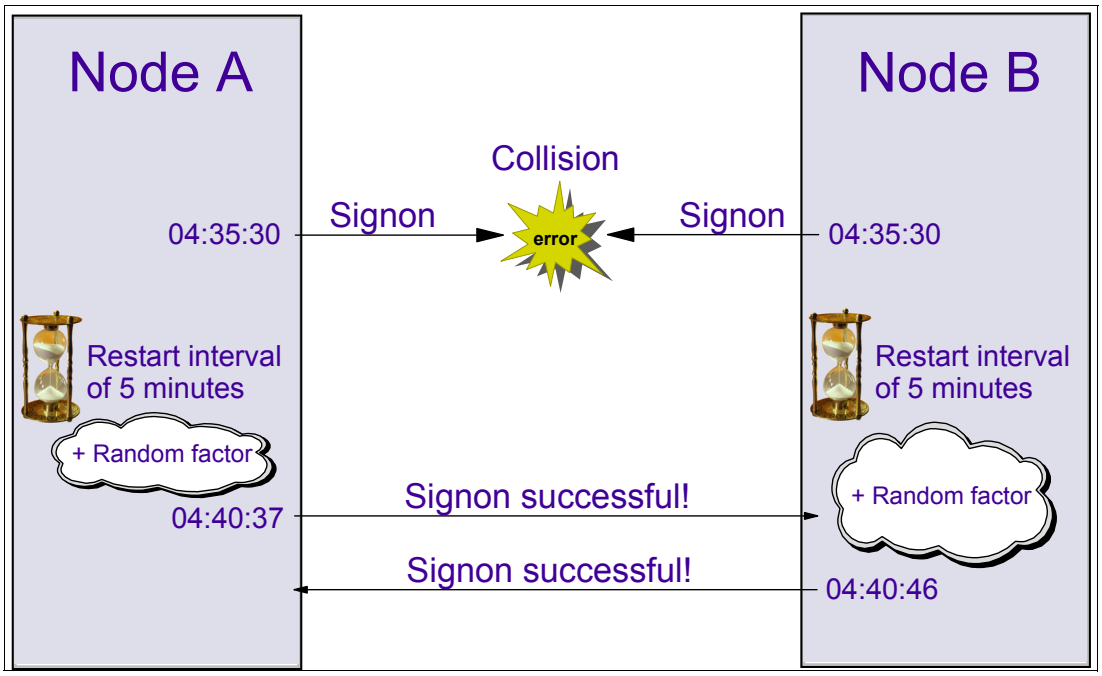


Figure 20-15 Collision avoidance with automatic restart

20.3.1 Operator commands

The JES2 network monitor support includes updates to existing operator commands, and it introduces several new operator commands.

Updated operator commands

Table 20-3 on page 441 lists the updates to existing JES2 operator commands in support of the NJE network monitor.

Table 20-3 Updated JES2 operator commands for NJE connections and devices

Operator command	Update
$D NJEDEF	Displays the value of the CONNECT parameter.
$T NJEDEF	Allows modifying the value of the CONNECT parameter.
$D NODE	Changed to display only basic node characteristics. For a detailed display, use **`$D NODE(xx),LONG`**.
$T NODE	Allows modifying the value of the CONNECT parameter.
$D APPL	Displays the value of the CONNECT parameter.
$T APPL	Allows modifying the value of the CONNECT parameter.
$D SOCKET	Displays the value of the CONNECT parameter.
$T SOCKET	Allows modifying the value of the CONNECT parameter.
$D LINE	Displays the value of the CONNECT and RESTART parameters.
$P LINE	Allows modifying the value of the RESTART parameter.
$T LINE	Allows modifying the value of the CONNECT and RESTART parameters.

Operator command	Update
$D LOGON	Displays the value of the CONNECT and RESTART parameters.
$P LOGON	Allows modifying the value of the RESTART parameter.
$T LOGON	Allows modifying the value of the CONNECT and RESTART parameters.
$D NETSRV	Displays the value of the CONNECT and RESTART parameters.
$P NETSRV	Allows modifying the value of the RESTART parameter.
$T NETSRV	Allows modifying the value of the CONNECT and RESTART parameters.
$SN LNExxxxx	Supports starting network connection for SNA and TCP/IP lines.

Figure 20-16 on page 443 shows an example of using some of the updated commands. (Note that not all the commands listed in Table 20-3 are shown in the figure.)

```
$D NJEDEF
$HASP831 NJEDEF
$HASP831 NJEDEF  OWNNAME=WTSCPLX2,OWNNODE=1,
$HASP831         CONNECT=(YES,10),DELAY=300,
$HASP831         HDRBUF=(LIMIT=100,WARN=80,FREE=100),
$HASP831         JRNUM=1,JTNUM=1,SRNUM=7,STNUM=7,
$HASP831         LINENUM=40,MAILMSG=YES,MAXHOP=0,
$HASP831         NODENUM=999,PATH=1,RESTMAX=0,
$HASP831         RESTNODE=100,RESTTOL=0,TIMETOL=0
$D NODE(2)
$HASP826 NODE(2)
$HASP826 NODE(2)    NAME=WTSCMXA,STATUS=(VIA/SC63),
$HASP826            TRANSMIT=BOTH,RECEIVE=BOTH,HOLD=NONE
$D NODE(2),LONG
$HASP826 NODE(2)
$HASP826 NODE(2)    NAME=WTSCMXA,STATUS=(VIA/SC63),
$HASP826            AUTH=(DEVICE=YES,JOB=YES,NET=NO,
$HASP826            SYSTEM=YES),TRANSMIT=BOTH,
$HASP826            RECEIVE=BOTH,HOLD=NONE,PENCRYPT=NO,
$HASP826            SIGNON=COMPAT,ADJACENT=NO,
$HASP826            CONNECT=(NO),DIRECT=NO,ENDNODE=NO,
$HASP826            REST=0,SENTREST=ACCEPT,COMPACT=0,
$HASP826            LINE=0,LOGMODE=,LOGON=0,NETSRV=0,
$HASP826            OWNNODE=NO,
$HASP826            PASSWORD=(VERIFY=(NOTSET),
$HASP826            SEND=(NOTSET)),PATHMGR=YES,
$HASP826            PRIVATE=NO,SUBNET=WTSCNET,TRACE=NO
$D NETSRV1
$HASP898 NETSRV1 918
$HASP898 NETSRV1    STATUS=ACTIVE,RESTART=(YES,5),
$HASP898            ASID=0074,NAME=JES2S001,
$HASP898            SOCKET=WTSCPLX2,STACK=,
$HASP898            TRACEIO=(JES=NO,COMMON=NO,
$HASP898            VERBOSE=NO)
$P NETSRV1,RESTART=NO
IAZ0536I NETSRV1 NJETCP SERVER RECEIVED A PURGE REQUEST
$HASP898 NETSRV1 920
$HASP898 NETSRV1    STATUS=DRAINING,ASID=0074,
$HASP898            NAME=JES2S001,SOCKET=WTSCPLX2,STACK=
$HASP097 NETSRV1    IS DRAINED
```

Figure 20-16 Updated JES2 operator commands example

New operator commands

Table 20-4 on page 443 lists the new JES2 operator commands added with the NJE network monitor support. These commands simplify the NJE network operations and allow for easy shutdown of JES2 or a recycle of VTAM.

Table 20-4 New JES2 operator commands for NJE network simplification

Operator command	Action
$D NETWORK	Displays network activity. This is the same display as the network section of the **$D JES2** operator command.

Operator command	Action
$E NETWORK	Resets current network devices.
$P NETWORK	Purges current network devices after all activity on the completes.
$S NETWORK	Starts automatic device restart and NJE connection processing that might have been stopped with a **$P NETWORK** command.
$Z NETWORK	Halts automatic network connection and device processing and drains NJE devices. This is similar to issuing **$E NETWORK** followed by **$P NETWORK**.

Figure 20-17 on page 444 shows an example of using the new operator commands listed in Table 20-4 on page 443.

```
$D NETWORK
$HASP899 $DNETWORK 000
$HASP899 ACTIVE NETWORKING DEVICES
$HASP899 NAME                            STATUS
$HASP899 --------------------------------
$HASP899 -----------------
$HASP899 NETSRV1                         ACTIVE
$E NETWORK
$HASP899 $ENETWORK 002
$HASP899 ACTIVE NETWORKING DEVICES
$HASP899 NAME                            STATUS
$HASP899 --------------------------------
$HASP899 -----------------
$HASP899 NETSRV1                         ACTIVE
IAZ0510I NETSRV1 NJETCP server bringing down all TCP/IP socket
connections
IAZ0511I NETSRV1 Server Port number could not be resolved, DEFAULT
assumed: 175
IAZ0540I NETSRV1 NJETCP Server accepting inbound requests
IAZ0537I NETSRV1 NJETCP SERVER WAITING FOR WORK
$P NETWORK
IAZ0536I NETSRV1 NJETCP SERVER RECEIVED A PURGE REQUEST
$HASP899 $PNETWORK 008
$HASP899 ACTIVE NETWORKING DEVICES
$HASP899 NAME                            STATUS
$HASP899 --------------------------------
$HASP899 -----------------
$HASP899 NETSRV1                         DRAINING
$D NETWORK
$HASP899 NO ACTIVE NETWORKING DEVICES
```

Figure 20-17 New JES2 operator commands for NJE networking example

21

HiperDispatch

z/OS workload management and dispatching have been enhanced to take advantage of the System z10 hardware design. The IBM z10 processor supports a new mode of dispatching called HiperDispatch (HD) which increases the system capacity by up to 10%. The amount of improvement varies according to the system configuration and workload. This chapter discusses the following:

- HiperDispatch overview
- Activating HiperDispatch
- Monitoring HiperDispatch

21.1 HiperDispatch overview

HiperDispatch is a combination of hardware, Hypervisor, and z/OS that increases system capacity. HiperDispatch increases system capacity by increasing the probability of cache hits when executing z/OS instructions. Each CPU has its own level 1 (L1) cache. This is the best place to find data as it requires the fewest machine cycles to access the data. CPUs are grouped at the hardware level in books.

All CPUs in the same book share a common level 2 (L2) cache. This is the second best place to access data. A CPU can also access the L2 cache of other books but this requires more machine cycles. The difference in machine cycles required to access a piece of data found in the L1 cache versus the same book L2 cache is relatively small. However, there is a significant difference in the number of machine cycles to access a piece of data in the same book L2 cache versus a different book L2 cache. To optimize for same book L2 cache hits, a unit of work must run on a subset of the available processors in the same book.

21.1.1 Without HiperDispatch

To describe the interaction between z/OS and Hypervisor we use a hypothetical example of a z10 processor with eight physical processors. The z10 is running two LPARs with the same weight and eight logical processors. Each LPAR will receive four physical CPUs of execution time which will be distributed across the eight logical processors defined to each LPAR, as shown in Figure 21-1.

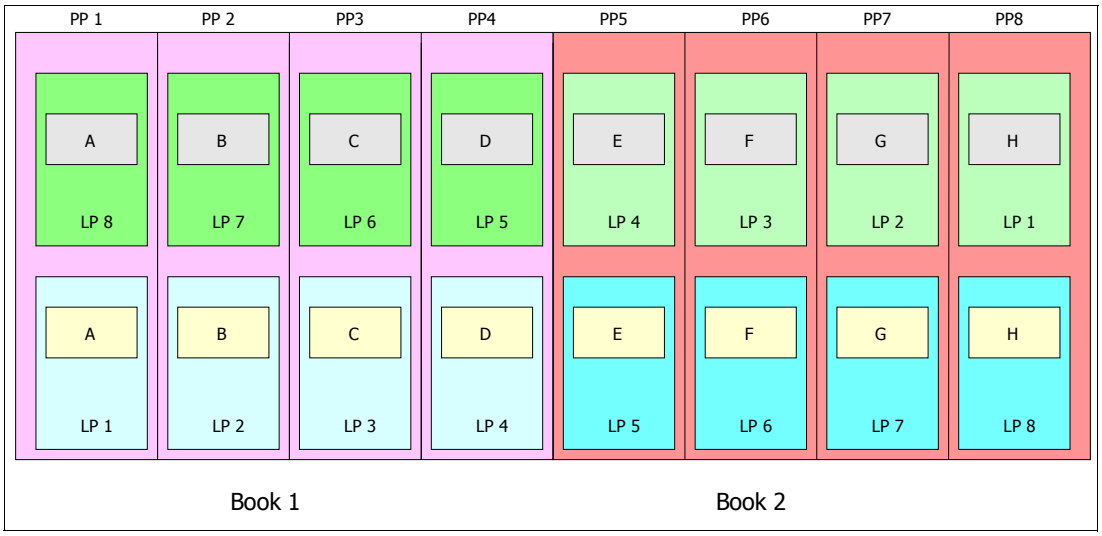


Figure 21-1 z10 without HiperDispatch

Without HiperDispatch, the Hypervisor can dispatch any logical processor on any physical processor and z/OS can dispatch any thread on any logical processor. This results in threads being dispatched randomly across physical processors. This has the effect of randomizing which physical processor is chosen reducing L1 and L2 cache hits which reduces the throughput of the system.

In HD=NO, there are three work unit queues (WUQs), one for CP work, one for zAAP work, and one for zIIP work. Each WUQ contains a queue of work element blocks (WEBs) to be dispatched by the CPs, zAAPs, or zIIPs. Since every CP, zAAP, and zIIP is dispatching work from its respective WUQ, any WEB can be dispatched by any processor of the same type.

Since WEBs can be dispatched on any physical processor of the same type in HD=NO, level 2 cache (book level) and level 1 cache (CPU level) are not optimized.

21.1.2 With HiperDispatch

HiperDispatch optimizes for same book L2 cache hits by dispatching threads intelligently on physical processors. First, Hipervisor needs to dispatch a given logical processor on the same physical processor. Then z/OS needs to dispatch a thread among a relatively static collection of processors in the same book to increase the probability of a same book L2 cache hit. Since a thread is going to be dispatched amongst a small group of processors, the probability of a L1 cache hit also increases.

Using our hypothetical example in Figure 21-2, if both LPARs consume their full LPAR weight, each LPAR will still receive four physical CPUs worth of execution time. With HiperDispatch=YES, the four physical CPUs worth of execution time is distributed amongst a subset of the logical processors defined to each LPAR. Normally, each LPAR uses four of its logical processors to process its work, as shown in Figure 21-2.

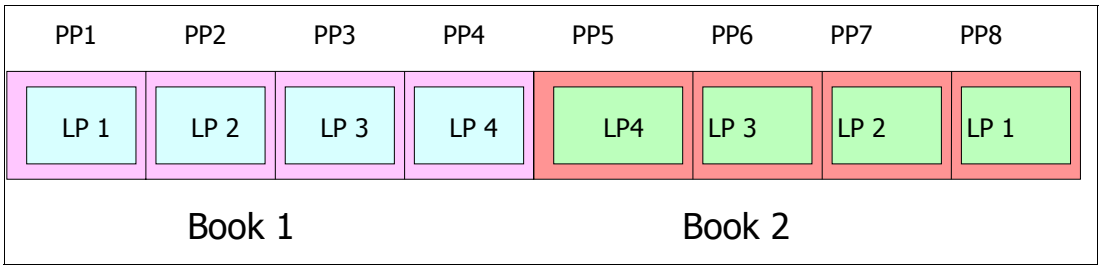


Figure 21-2 z10 with HiperDispatch and two LPARs of equal weight

Since HiperDispatch optimizes the same book L2 cache, Hipervisor assigns four logical processors 1, 2, 3, and 4 from the blue LPAR to physical processors 1, 2, 3, and 4 in book 1 and the four logical processors 1, 2, 3, and 4 from the green LPAR to physical processors 8, 7, 6, and 5 in book 2. These logical processors are referred to as vertical highs because one logical processor is mapped to exactly one physical processor. When more than one logical processor is mapped to the same physical processor, that logical processor is referred to as a vertical medium. Multiple vertical mediums share the processing power provided by one physical processor.

There are four logical processors in each LPAR that are unaccounted for. Those remaining four logical processors are not mapped to any physical processors. These logical processors are referred to as vertical lows or discretionary processors and are normally in a parked state. When the z10 is busy and starts to approach 100% utilization the vertical lows and then vertical mediums are parked. A parked CPU is online but will not dispatch work or process I/O interrupts. An LPAR's vertical lows float on top of a different LPAR's vertical highs/mediums so that an LPAR that needs more capacity can receive additional capacity from a different LPAR which is not consuming its full weight. An LPAR will only expand into its vertical lows when both of the following conditions are met:

- An LPAR is consuming its full LPAR weight.
- A different LPAR is not consuming its full weight.

When both conditions are met, the LPAR consuming its full weight can unpark one or more of its vertical lows to use another LPAR's unused capacity. Once a vertical low is unparked it will dispatch work on behalf of that LPAR. The vertical low will be parked once the overworked LPAR no longer needs the extra capacity or the other LPAR starts consuming its full weight.

Grouping CPUs into affinity nodes

z/OS groups logical processors into entities called affinity nodes. Processors assigned to an affinity node tend to be in the same book to increase the probability of an L2 cache hit. All CPUs in an affinity node have the same CPU type (CP, zAAP, or zIIP). Each affinity node has its own WUQ and all CPUs assigned to that affinity node dispatch work from that WUQ. HiperDispatch optimizes cache hits by ensuring that threads are redispatched on the same affinity node.

To ensure that the responsiveness of high priority work is not impacted with HiperDispatch, high priority work is assigned to a new WUQ called the high performance WUQ (HPWUQ). All standard CPs dispatch work from the HPWUQ before dispatching work on their affinity WUQ. The HPWUQ contains work element blocks (WEBs) for high priority work that includes:

- WEBs with dispatching priority 255. Address spaces with service class SYSTEM are assigned dispatching priority 255.
- SRB WEBs with dispatching priority 254. Address spaces with service class SYSSTC are assigned dispatching priority 254.
- Lock promotion WEBs. These have dispatching priority 255.

HiperDispatch responds to utilization spikes within an affinity node by assigning processors from another affinity node to perform work from the busy affinity node's WUQ. In addition, WLM gathers statistics every two seconds to distribute the workload evenly across the affinity nodes.

There is a significant improvement in the capacity of the system (0%-10% depending on system configuration and the workload) by optimizing for level 2 cache hits with HiperDispatch. The more physical processors a z10 has or the larger the overcommit ratio between logical processors and physical processors, the larger the capacity gain from HiperDispatch.

21.2 Activating HiperDispatch

HiperDispatch is only supported by the IBM z10 processor, device type 2097. It is part of the z/OS V1R10 base code and available for HBB772S, HBB7730, and HBB7740 via APARs:

- OA20633 and OA23333 for supervisor
- OA20418 for WLM
- OA12774 for RMF

IEAOPTxx parmlib member

To activate HiperDispatch a new keyword in IEAOPTxx is used:

- HiperDispatch = YES
- The default is HiperDispatch = NO

HiperDispatch can be activated at IPL by specifying HiperDispatch = YES in the IEAOPTxx used during IPL. It can also be activated and de-activated dynamically by updating IEAOPTxx or creating a new IEAOPTxx specifying HiperDispatch = YES or NO and issuing the following MVS command:

```
SET OPT=xx
```

HiperDispatch messages

When HiperDispatch is activated, the following message is issued:

```
IRA860I HIPERDISPATCH MODE IS NOW ACTIVE
```

When HiperDispatch is deactivated, the following message is issued:

```
IRA861I HIPERDISPATCH MODE IS NOW INACTIVE
```

HiperDispatch example

When HiperDispatch is activated the supervisor vector table (SVT) flag SVTAFFON is turned on, the system WUQ (SWUQ) becomes the HPWUQ, and the affinity WUQs are created for CPs, zAAPs, and zIIPs.

For example, suppose an LPAR on a z10 consisted of eight CPs, four zAAPs, and four zIIPs.

In HiperDispatch = NO there would be one WUQ for the CPs (SWUQ), one WUQ for the zAAPs (ASWUQ) and one WUQ for the zIIPs (SSWUQ). This would be true regardless of how many CPs, zAAPS, or zIIPs were available.

When the system transitions to HiperDispatch = YES affinity nodes are assigned for CPs, zAAPs, and zIIPs. In our example, two affinity nodes would be created for the eight CPs, one for the four zAAPs and one for the four zIIPs and a WUQ would be assigned to each affinity node. In addition, a HPWUQ would be assigned from which all CPs can dispatch work.

21.3 Monitoring HiperDispatch

HiperDispatch has implications for workload management (WLM) and help processing. There are also changes in RMF reports to display HiperDispatch information.

21.3.1 WLM considerations

With HiperDispatch the prioritization of workloads via WLM policy definitions becomes more important because access to processors changes. To optimize cache hits a work unit has access to a smaller number of processors, which increases the potential for queuing delays.

As HiperDispatch changes how work is dispatched among the processors, additional attention and review of the WLM policy may be needed to ensure proper workflow through the system. With HiperDispatch it is important that critical work, highly interactive work and CPU intensive work is prioritized appropriately.

Prior to z/OS V1R10, only the Master address space and WLM are automatically assigned with service class SYSTEM and cannot be assigned a different service class. In z/OS V1R10 there are extra system addresses that are classified into SYSTEM and cannot be changed. They are XCFAS, GRS, CONSOLE, IEFSCHAS, IXGLOGR, SMF, CATALOG, SMSPDSE, and SMSPDSE1.

With HiperDispatch, work for address spaces with service class SYSTEM runs on the High Performance WUQ (HPWUQ), ensuring high access to CPU.

21.3.2 RMF reports

With HiperDispatch=Yes different processor utilizations are seen in the RMF CPU Activity report depending on whether a CPU is a vertical high, vertical medium, or vertical low.

With the RMF HiperDispatch APAR OA12774, the RMF CPU Activity report has a new column for PARKED time % and is enhanced to indicate the high, medium or low share via the LOGICAL PROCESSOR SHARE % column. In addition changes are introduced with APAR OA24074, which adds an indication if HiperDispatch is active on the CPU type and model information line. OA24074 also changes the calculation of the MVS view of CPU utilization to take into account that logical processors can be parked.

With APAR OA24074, the calculation is:

$$\text{MVS UTIL(\%)} = \frac{\text{Online Time - (Wait + Parked Time)}}{\text{Online Time - Parked Time}} * 100$$

This also affects the AVG MVS UTIL(%) for all logical processors because the MVS UTIL(%) for each logical processor is weighted by the time being online and unparked. This effect becomes obvious with processors being parked partially during the interval. For example, an MVS UTIL of 100% for a processor parked 80% means the processor was unparked for 20% of the interval and busy the whole time it was unparked. The interval for this processor adds less to the overall average than an MVS UTIL of 100% for a processor that was not parked.

Figure 21-3 shows an example of a CPU activity for a system with HiperDispatch=Yes and OA24074 installed running on a z10. The logical processor share for the partition of 640.0% was allocated across five logical processors with a high share of 100%, two logical processors with a medium share of 70%, and one discretionary logical processor, CP 7, with a low share of 0% which was parked 85.78% and thus unparked 14.22% of the online interval time. During this same interval, CP 7 was busy processing 13.84% of the time.

With HIPERDISPATCH=NO, the logical processor share would be 80% for each of the 8 logical processors. There are 12 vertical highs, 7 vertical mediums, and 14 vertical lows.

```
                          C P U  A C T I V I T Y

                    z/OS V1R10               SYSTEM ID S59              DATE 11/28/200
                    RPT VERSION V1R10 RMF      TIME 16.45.00               CYCLE 1.000 SECONDS
CPU 2097  MODEL 732  H/W MODEL  E40  SEQUENCE CODE 000000000000DC6CE   HIPERDISPATCH=YES
---CPU---   ---------------- TIME % ----------------   LOG PROC    --I/O INTERRUPTS--
NUM  TYPE   ONLINE   LPAR BUSY   MVS BUSY   PARKED    SHARE %     RATE    % VIA TPI
 0    CP    100.00   96.33       97.34       0.00    100.0        5.80   48.75
 1    CP    100.00   95.96       97.07       0.00    100.0        4.59   55.30
 2    CP    100.00   95.79       96.84       0.00    100.0        5.10   55.18
 3    CP    100.00   95.46       96.68       0.00    100.0        2.40   53.75
 4    CP    100.00   95.08       96.41       0.00    100.0        8435   10.05
 5    CP    100.00   73.92       96.86       0.00     70.0       20.74    4.95
 6    CP    100.00   74.33       97.13       0.00     70.0       14.15   19.39
 7    CP    100.00   13.84       98.89      85.78      0.0        0.00    0.00
TOTAL/AVERAGE       80.09       96.94                640.0        8488   10.14
```

Figure 21-3 RMF CPU Activity report

21.4 Help processing

Help processing occurs when an affinity node is overcommitted and the dispatcher determines that it needs help. This is done by assigning the WUQ for the overcommitted

affinity node to another less busy CPU. In HiperDispatch=NO all CPUs are candidates to give help while in HiperDispatch=YES preference is given to CPUs in the same affinity node.

When a CP affinity node needs help:

- In HiperDispatch=NO, all CPs are candidates to give help.
- In HiperDispatch=YES, waiting CPs in the same affinity node are the first processors chosen for help. If there are no waiting CPs in the same affinity node, a good candidate CP with the same book is chosen. If there are no good candidates in the same book and the affinity node needs help badly enough, a CP in a different book can be chosen for help.

When a zAAP affinity node needs help:

- In HiperDispatch=NO, help from another zAAP is preferred. If all other zAAP processors are busy, a CP is chosen if IFAHONORPRIORITY=YES.
- In HiperDispatch=YES, waiting zAAPs in the same affinity node are the first processors chosen for help. If all the other zAAP processors in the same affinity node are busy, help can be provided by a zAAP in a different affinity node or a CP if IFAHONORPRIORITY=YES.

When a zIIP affinity node needs help:

- In HiperDispatch=NO, help from another zIIP is preferred. If all other zIIP processors are busy, a CP is chosen if IFAHONORPRIORITY=YES.
- In HiperDispatch=YES, waiting zIIPs in the same affinity node are the first processors chosen for help. If all the other zIIP processors in the same affinity node are busy, help can be provided by a zIIP in a different affinity node or a CP if IFAHONORPRIORITY=YES.

21.4.1 Alternate wait management (AWMT)

The alternate wait management (AWMT) and honor priority values specified via IEAOPTxx affect help processing for both HiperDispatch=YES and HiperDispatch =NO.

The keywords in IEAOPTxx are:

- CCCAWMT
  - Alternate Wait Management (AWM) value for normal CPs
  - For HiperDispatch = NO the valid range is 1-1000000 microseconds.

    Specifying CCCAWMT >= 500000 disables AWM.
  - Default for HiperDispatch = NO is 12000 microseconds.
  - For HiperDispatch =YES the valid range is 1600-3200 microseconds. Specifying a value outside that range will result in 3200 microseconds being assigned.
  - Default for HiperDispatch = YES is 3200 microseconds.

Note: For a dedicated LPAR, AWM is always inactive. For a shared LPAR, AWM is always active with HiperDispatch=Yes. For HiperDispatch=No, AWM can be disabled by specifying CCCAWMT >= 500000.

- ZAAPAWMT
  - Alternate Wait Management (AWMT) value for zAAP processors

  - For HiperDispatch = NO the valid range is 1-499999 microseconds.
  - Default for HiperDispatch = NO is 12000 microseconds.
  - For HiperDispatch = YES the valid range is 1600-3200 microseconds. Specifying a value outside that range will result in 3200 microseconds being assigned.
  - Default for HiperDispatch =YES is 3200 microseconds.
- ZIIPAWMT
  - Alternate Wait Management (AWMT) value for zIIP processors
  - For HiperDispatch = NO the valid range is 1-499999 microseconds.
  - Default for HiperDispatch = NO is 12000 microseconds.
  - For HiperDispatch = YES the valid range is 1600-3200 microseconds. Specifying a value outside that range will result in 3200 microseconds being assigned.
  - Default for HiperDispatch =YES is 3200 microseconds.
- IFAHONORPRIORITY=YES|NO
  - Specifying IFAHONORPRIORITY=YES means that normal CPs will help when zAAPs need help.
  - Specifying IFAHONORPRIORITY=NO means that normal CPs will not be eligible to help zAAPs.
- IIPHONORPRIORITY=YES|NO
  - Specifying IFAHONORPRIORITY=YES means that normal CPs will help when zIIPs need help.
  - Specifying IFAHONORPRIORITY=NO means that normal CPs will not be eligible to help zIIPs.

An affinity node is deemed to need help if either:

- A CPU assigned to that affinity node's WUQ has been executing work continuously for at least an AWMT interval - CCCAWMT for CPs, ZAAPAWMT for zAAPs or ZIIPAWMT for zIIPs.
- The queue length of the WUQ exceeds a threshold value. This effectively means there is too much work queued to the WUQ and it needs help.

The interval used to check whether a processor needs help is also based on the AWMT values. This means the AWMT values affect how responsive help processing is to spikes in workload and also the criteria to determine whether a processor needs help.

In HiperDispatch = YES, help is given for a certain number of dispatches by the chosen CPU while in HiperDispatch = NO the CPU chosen to give help will continue to dispatch work from the WUQ needing help until the WUQ is empty.

When a system is IPLed specifying HiperDispatch = NO the default value for CCCAWMT, ZAAPAWMT and ZIIPAWMT of 12000 micro-seconds is used. These can be seen via the SVT fields:

- SVTTODDL at SVT+X'278' for CCCAWMT = X'02EE0000'
- SVT_SupAWMT_Elapsed_Timer at X'39C' for ZIIPAWMT = X'02EE0000'
- SVT_IFAAWMT_Elapsed_Timer at X'3AC' for ZAAPAWMT = X'02EE0000'

This value is in 64-bit TOD format where bit 51 corresponds to 1 micro-second. This means that these values can be converted to microseconds by dropping the last three digits (nibbles). X'02EE0000' = X'02EE0' micro-seconds = 12000 micro-seconds.

When the system transitions to HiperDispatch = YES, in addition to using affinity node WUQs these values change to:

- SVTTODDL at SVT+X'278' for CCCAWMT = X'00C80000'
- SVT_SupAWMT_Elapsed_Timer at X'39C' for ZIIPAWMT = X'00C80000'
- SVT_IFAAWMT_Elapsed_Timer at X'3AC' for ZAAPAWMT = X'00C80000'

  `X'00C80000' = X'00C80' micro-seconds = 3200 micro-seconds.`

22

IOS enhancements

This chapter describes the following IOS enhancements and new functions:

- Single point of failure detection - SPOF
- z/OS Basic HyperSwap
- Exploitation of subchannel sets for PPRC secondary devices

22.1 Single point of failure detection

A failure of a single hardware component can cause a system to stop functioning. With the complexity of most I/O configurations, it becomes very complicated to determine whether there are any single hardware components that could cause an outage when they fail.

A new IOS service, IOSSPOF, has been added to z/OS V1R10 to detect single points of failure in an I/O configuration. There are 16 single points of failure conditions that are currently detected by IOSSPOF. The main objective of this new service is to reduce the number of unplanned outages.

IOSSPOF macro

The IOSSPOF macro is used to check for I/O configuration redundancy of DASD devices or pairs of DASD devices. To do this, IOSSPOF verifies that there are redundant hardware components such that failure of a hardware component would not affect the availability of the device. IOSSPOF detects whether devices containing critical data sets have common hardware components through all paths. It will also detect if primary and secondary copies of a data set have shared hardware components.

The IOSSPOF service is intended to be used as a way of determining single points of failure without issuing I/O. It can be invoked in any addressing mode or storage key and has been integrated with Health Checker. This ensures that the messages are in consistent format for each type of failure.

XCF is exploiting this service in z/OS V1R10 to check for single points of failure for devices that contain sysplex CDS data sets.

22.1.1 Examples of single points of failure detected by IOSSPOF

The following diagrams illustrate single points of failure that can be detected by IOSSPOF.

Figure 22-1 on page 457 illustrates single points of failure detected by IOSSPOF for a single device. These include ensuring that:

- Not all paths to a device use host interfaces on the same I/O cage.
- Not all paths to a device share a common switch.
- Not all paths to a device share a common control unit interface.

Cascaded switch

A “cascaded switch connection” means that two switches, shown in Figure 22-1 on page 457, are interconnected to provide the complete serial link connection between the host, another switch, and the control unit.

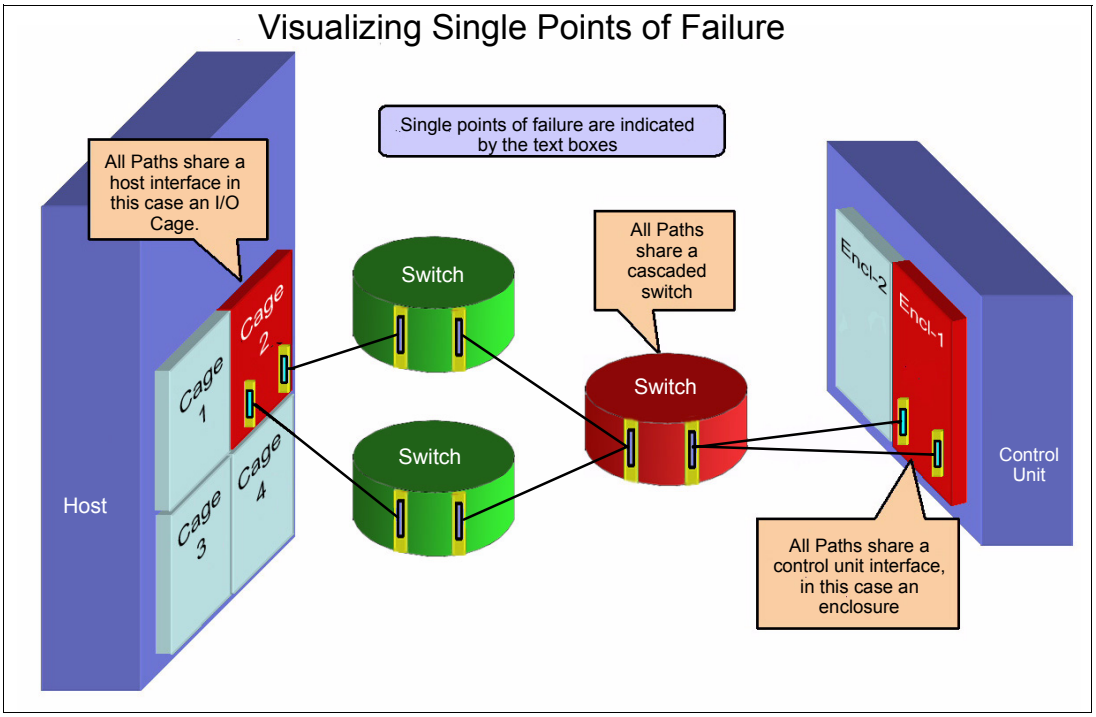


Figure 22-1 Single points of failure - 1

Figure 22-2 on page 458 and Figure 22-3 on page 458 illustrate single points of failure detected for devices that contain a primary and alternate data set. An example of this would be the primary and secondary device containing the primary and alternate sysplex CDS data sets. These single points of failure checks include ensuring that:

- Not all paths to a primary and secondary device are sharing the same switch.
- Not all paths to a primary and secondary device are sharing the same control unit interface.
- All paths are not using a non-preferred path. This would be a path via a control unit interface on the other side of the disk server.
- The primary and secondary devices share the same control unit.

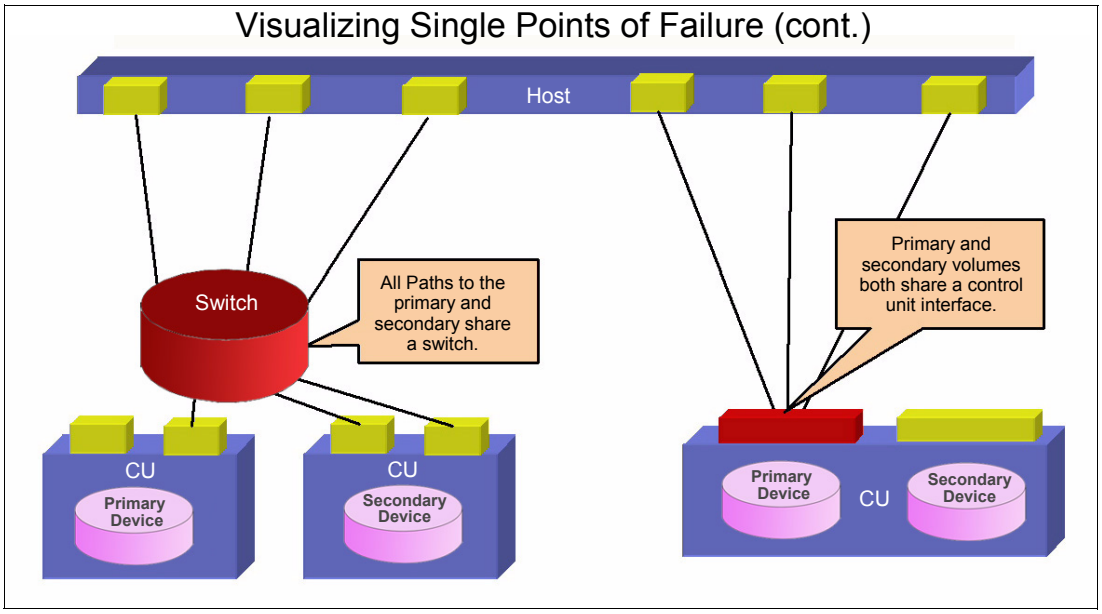


Figure 22-2 Single points of failure - 2

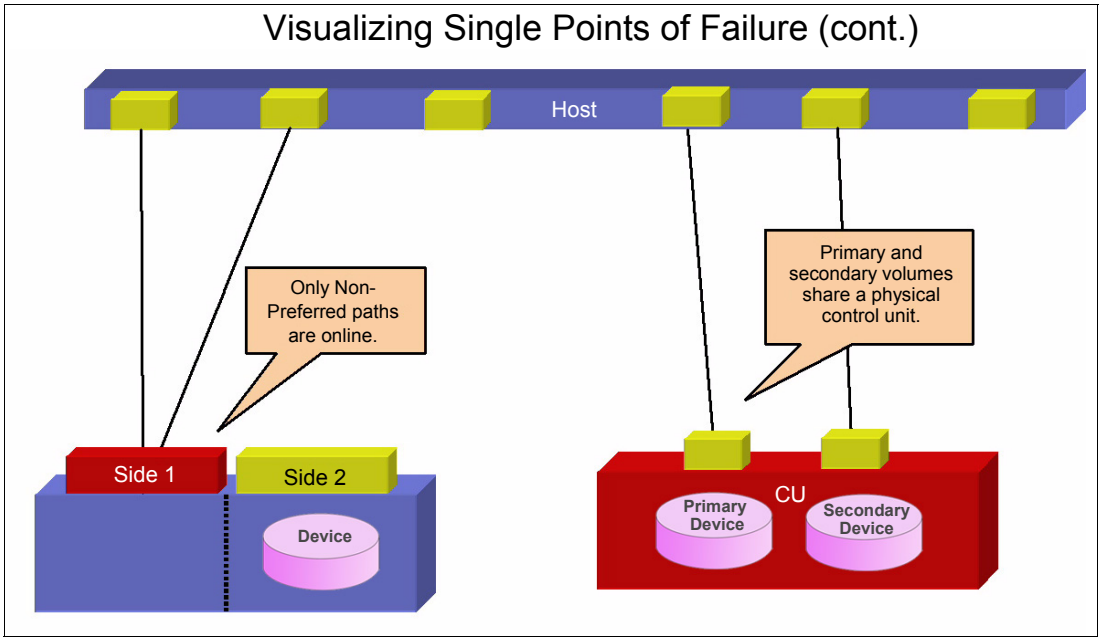


Figure 22-3 Single points of failure - 3

22.1.2 Using the IOSSPOF service

The IOSSPOF service is available for both vendors and customers to write Health Check to warn of single points of failure in I/O configurations. SPoF determination can be used with Health Checker using a local check for long-running jobs that use a disk resource. It can also be invoked with Health Checker using a remote check for relatively shorter running jobs.

The interface allows WTO rtecode(11) messages to be issued and a control block to automate some tests that require the knowledge of single points of failure. The IOSSPOF macro in Figure 22-4 on page 459 is documented in *z/OS MVS Programming: Authorized*

Assembler Services Reference, Volume 2 (EDTINFO-IXGWRITE), SA22-7610. Use the macro as follows:

- A pair of devices, DEVN1 and DEVN2 or volumes VOLSER1 and VOLSER2, can be specified to test for single points of failure for a primary and secondary pair. DEVLIST and VOLLIST cannot be used for primary and secondary device checking.
- The values supplied for DSN1 are for message purposes only.
- Specifying HCMSG=YES results in the IOSSPOF macro automatically issuing the Health Checker message in a check environment. HANDLE= is for remote checks.
- Specifying SPOFAREA results in a control block mapped by IOSDSPOF to be returned. This can be interpreted programmatically to report single points of failure.
- Offline devices are not currently supported.

The IOSSPOF is exploited by XCF to detect single points of failure for primary and alternate CDS data sets. This is described in 9.5, “New and updated XCF Health Checks” on page 232.

```
[xlabel]  IOSSPOF     PERFORM_CHECK
                       + ,DEVN1=xdevn1
                       |    [,SCHSET1=xschset1]
                       |    [,DSN1=xdsn1]
                       |    [,DEVN2=xdevn2
                       |        [,SCHSET2=xschset2]
                       |        [,DSN2=xdsn2]]
                       + ,VOLSER1=xvolser1
                       |    [,DSN1=xdsn1]
                       |    [,VOLSER2=xvolser2
                       |        [,DSN2=xdsn2]]
                       + ,DEVLIST=xdevlist
                       |    ,DEVCOUNT=xdevcount
                       |   [,DSNLIST=xdsnlist]
                       + ,VOLLIST=xvollist
                            ,VOLCOUNT=xvolcount
                           [,DSNLIST=xdsnlist]
                        [,SPOFAREA=xspofarea]
                        [,HCMSG=NO]
                        [,HCMSG=YES
                            [,HANDLE=xhandle]]
                        [,WTO=NO]
                        [,WTO=YES]
                        [,IND_CHECKS=YES]
                        [,IND_CHECKS=NO]
                        [,IND_CHECKS=ONLY]
                        [,SWITCH_CHECKS=YES]
                        [,SWITCH_CHECKS=NO]
                        [,CU_CHECKS=YES]
                        [,CU_CHECKS=NO]
                     [,RETCODE=xretcode]
                     [,RSNCODE=xrsncode]
```

Figure 22-4 IOSSPOF macro

22.2 z/OS Basic HyperSwap

The Basic HyperSwap function provided with z/OS V1R10 ensures continuous availability of storage subsystems by eliminating the single point of failure for primary device I/O errors. Basic HyperSwap is a single site-only solution and is not intended to be a disaster recovery solution. It is designed to react to primary device I/O errors, not to PPRC link failures that can occur in a cross-site configuration.

Having data mirrored by PPRC and automatically swapping to the secondary disk when a permanent I/O error occurs on a primary disk will help to eliminate application failures and allow planned swaps to reduce system downtime during disk subsystem maintenance.

The interface to z/OS Basic HyperSwap is TotalStorage Productivity Center for Replication (TPC-R) V3.4 and higher.

22.2.1 HyperSwap overview

Basic HyperSwap provides a set of functions that allow a HyperSwap management application to interact with the I/O Supervisor to perform DDR swaps of UCBs on a mass scale. HyperSwap functions are performed in parallel for each Logical Subsystem (LSS) in the PPRC configuration.

Basic HyperSwap does not provide an interface to directly manage and configure the PPRC configuration. It provides functionality to perform a HyperSwap on a PPRC configuration file via the authorized PC interface (IOSHXSWP). The interface to z/OS basic HyperSwap is TotalStorage Productivity Center for Replication (TPC-R) V3.4 and higher.

A HyperSwap can only be performed on a PPRC configuration that is in DUPLEX status. The primary volumes are continuously monitored for events that may trigger a HyperSwap, such as:

- A permanent I/O error on a primary volume will trigger a HyperSwap for all devices in the configuration file.
- A PPRC status change will cause Basic HyperSwap to be disabled. Basic HyperSwap will be re-enabled when the PPRC devices are back in DUPLEX status.

Unplanned HyperSwap

An unplanned HyperSwap is triggered by disk I/O failure and is signalled via Event Notification Facility (ENF) 63. The following disk I/O errors will trigger an unplanned HyperSwap with the following messages:

- PPRC primary disk problem (IEA491E with reason “PRIMARY-DEVICE-WRITE-FAILURE”)
- Permanent I/O errors (IEA497I)
- No paths available (IOS002A)
- Boxed devices (IOS107I)
- I/O Timing timeout if enabled as a HyperSwap trigger (IOS078I)

Planned HyperSwap

A planned HyperSwap is triggered by a request via TPC-R. Figure 22-5 on page 461 shows a PPRC configuration that is enabled for HyperSwap. Host applications are performing I/O to the primary PPRC volumes which are being synchronously mirrored. The PPRC paths were

established with CGROUP(YES) so that the PPRC configuration is part of a consistency group.

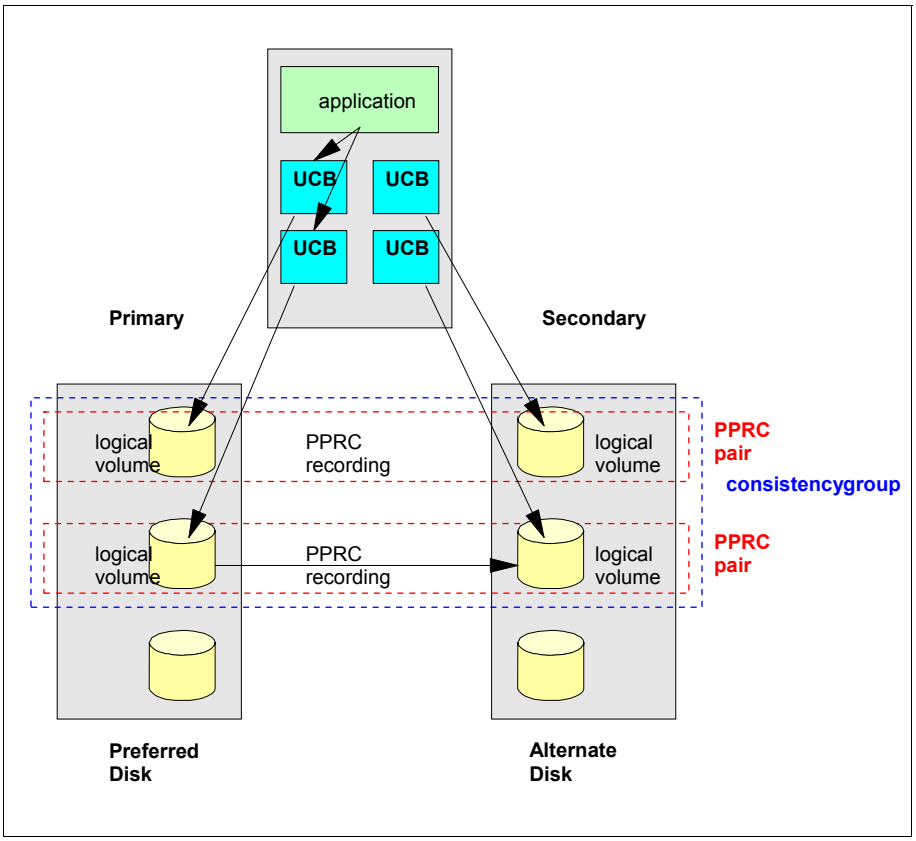


Figure 22-5 PPRC configuration before HyperSwap

HyperSwap activation

When a HyperSwap is triggered, the following steps are performed:

1. A CGROUP FREEZE is issued, which deletes the PPRC paths and places the primary volumes in an Extended Long Busy (ELB) state.
2. Primary disk I/O is quiesced.
3. A PPRC FAILOVER is performed, which terminates PPRC and places both primary and secondary volumes in PRI-SUSP status. This allows change recording to continue so that PPRC DUPLEX status can be re-established by copying only changed tracks.
4. The contents of the UCBs for the primary and secondary volumes are swapped. This means that the primary UCB now points to the secondary volume and has the secondary device number.
5. Primary disk I/O is resumed and ALIAS BIND processing is performed.

At this point PPRC is suspended and will remain in this state for an Unplanned HyperSwap. The new primary devices are in PRI-SUSP status so that change recording is being done for subsequent write I/O.

For a planned HyperSwap the PPRC status can be requested to remain SUSPENDed or to be returned to DUPLEX status. Figure 22-6 on page 462 shows the PPRC configuration after a planned HyperSwap where the PPRC configuration remains in DUPLEX status. This ensures that there is no data loss as a result of the HyperSwap if there is a subsequent disk

failure on the new primary volumes. In this case an extra step is performed before primary disk I/O is resumed: Establish PPRC in the reverse direction.

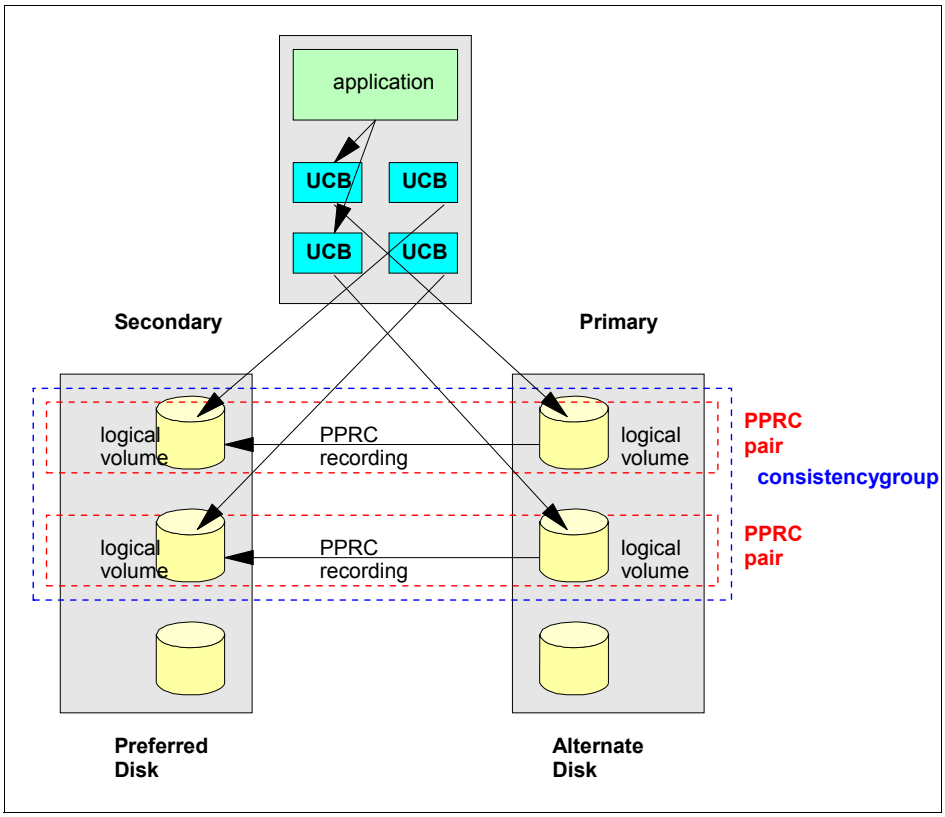


Figure 22-6 PPRC configuration after planned HyperSwap

22.2.2 Basic HyperSwap software components

Basic HyperSwap requires the following software components:

- z/OS Basic HyperSwap function provided in z.OS V1R10 or z/OS V1R9 with APAR OA20658.
- TotalStorage Productivity Center for Replication (TPC-R) V3.4 and higher. This is the interface to z/OS basic HyperSwap.
- XCF communication provided via a Parallel Sysplex, base sysplex, or monoplex.

Figure 22-7 on page 463 shows a two-way sysplex with TPC-R active on one of the members and the control flow through Basic HyperSwap for a request initiated by TPC-R.

Basic HyperSwap API services are invoked by TPC-R via a Program Call (PC) instruction and use XCF for cross-system communication.

Basic HyperSwap automatically assigns one system in the sysplex to be the master system that controls HyperSwap processing. When the master system becomes unavailable during a HyperSwap, a new master will be selected via sysplex group exit processing. A request sent to the master system during a master takeover will be queued until a new master system is determined and delivered to the new master system.

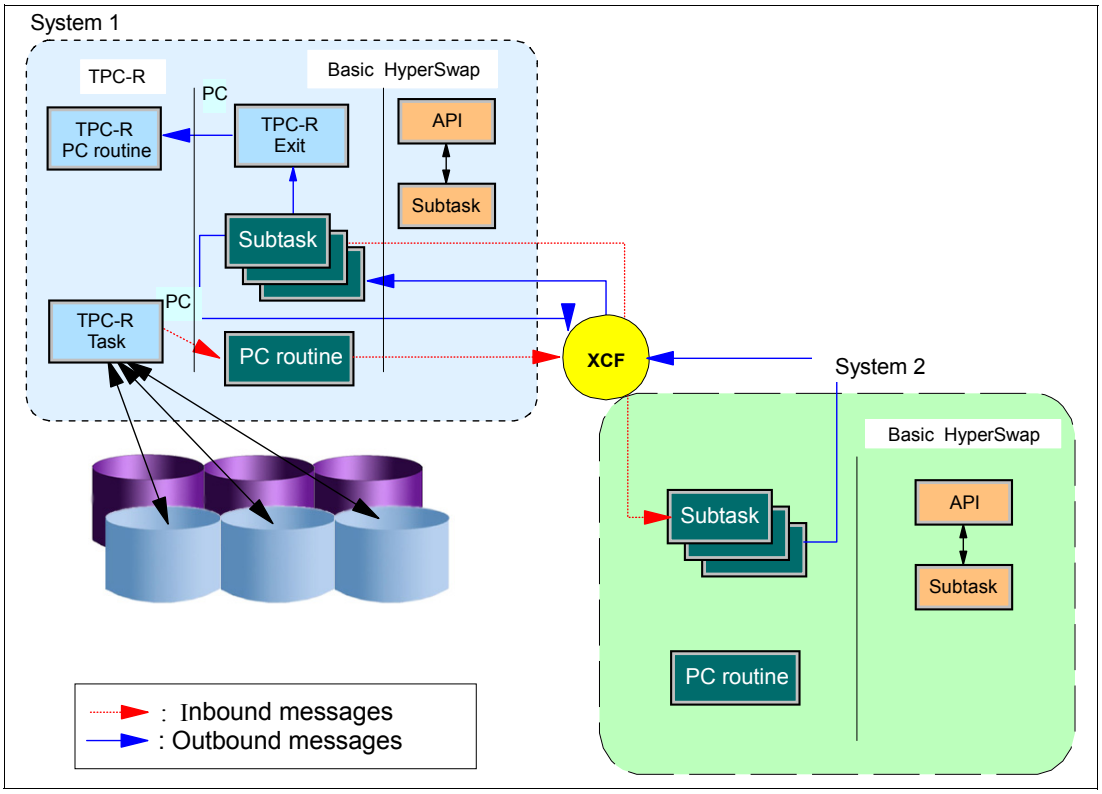


Figure 22-7 Basic HyperSwap software components

22.2.3 TotalStorage Productivity Center for Replication (TPC-R) overview

TPC-R is a comprehensive system storage management product that can operate on multiple platforms. It is a Web-based solution for storage management including capacity monitoring and automation, and backup and recovery. Communication to disk is via CCWs through System Data Mover (SDM).

There is only one instance of TPC-R in the SYSPLEX. TPC-R performs the following disk-related functions:

- Creates and maintains DASD PPRC pairs.
- Provides PPRC configuration to Basic HyperSwap.
- Requests Basic HyperSwap to take actions on the configuration (Swap, Purge).

Note: An entitlement version of TPC-R 3.4 (only HyperSwap function) will be made available to z/OS customers to use with Basic HyperSwap.

22.2.4 Basic HyperSwap commands

Basic HyperSwap accepts commands from TPC-R or operator commands.

TPC-R commands

The following commands can be issued via TPC-R:

LoadTest Validate a configuration file

Load	Validate and activate a configuration file
Purge	Deactivate the currently active configuration file
Swap	Perform HyperSwap for devices in the current configuration file
Enable	Allows HyperSwap to be performed (if no other disablement condition exists)
Disable	Prevents HyperSwap to be performed (even though no other disablement condition exists)
Status	Query the current Basic HyperSwap status

Operator commands

The Basic HyperSwap status can be displayed and controlled via new operator commands.

- DISPLAY HS,STATUS

  Displays the status of HyperSwap. This command also displays any reasons why basic HyperSwap may be disabled, and the current policies for the Basic HyperSwap Session.

- D HS,CONFIG(DETAIL,ALL)

  Displays the detailed configuration for the current Basic HyperSwap session. This will list the volumes and status of all pairs in the basic HyperSwap configuration.

- SETHS ENABLE

  Enables HyperSwap. This allows a HyperSwap to be performed, either by command or automatically, if HyperSwap is not disabled for other reasons.

- SETHS DISABLE

  Disables HyperSwap. This prevents a HyperSwap from being performed, either by command or automatically.

- SETHS SWAP

  Performs a planned HyperSwap. This can be done instead of issuing the HyperSwap command from TPC-R.

Note: You must have at least UPDATE authority to use the SETHS command.

22.2.5 Basic HyperSwap status

The Basic HyperSwap status is either:

- ENABLED - Ready to perform HyperSwap
- DISABLED - HyperSwap cannot be performed

Each member of the sysplex keeps a bitmap of disablement reasons which are copied to the master system via the XCF group exit so that the master system sees an aggregate of all disablement reasons for all systems.

The disablement reasons can be displayed via

```
DISPLAY HS,STATUS
```

Basic HyperSwap tracks multiple disablement reasons and HyperSwap will be DISABLED if any one disablement reason is set in the bitmap for any system. The following highlighted message is issued when HyperSwap is disabled:

```
IOSHM0803I HyperSwap Disabled
```

Basic HyperSwap returns to ENABLED status when all the disablement reasons are resolved and the disablement bits are off. IOSHM0803I will then be removed.

Disablement reasons

There are two levels of disablement reasons:

- Member level

  These are conditions that reflect individual member status:

  - No configuration loaded
  - Basic HyperSwap or API address space not active
  - Basic HyperSwap Management address space is terminating
  - Failure loading the configuration
  - No connection to PPRC secondary device(s)

- Master level

  These are conditions that affect overall Basic HyperSwap functions:

  - Disable requested by operator via the SETHS command
  - HyperSwap in progress
  - Configuration load or purge is in progress
  - PPRC pairs suspended or not fully established

22.2.6 Basic HyperSwap implementation and customization

This section discusses the implementation and customization requirements of Basic HyperSwap.

Address spaces

Basic HyperSwap requires two address spaces. Started tasks need to be created in SYS1.PROCLIB to run:

- The HyperSwap management address space that executes program IOSMCTL. This address space:
  - Maintains configuration information.
  - Monitors devices in the configuration.
  - Controls and coordinates HyperSwap activities.
- The HyperSwap API Services address space that executes program IOSHSAPI. This address space:
  - Communicates directly with disk control units and devices.
  - Issues commands to disk control units and devices to perform requests from Basic HS Management address space.

Sysplex requirements

The systems sharing the PPRC configuration managed by Basic HyperSwap must be in a sysplex. XCF is used to communicate between tasks on the same system and between systems in the sysplex.

- The sysplex couple data set volume should not be in the HyperSwap-managed PPRC configuration. It is used to store member status information while coordinating swap

processing in the sysplex. If I/O to the sysplex CDS volume hangs, the swap processing cannot complete.

- The other couple data set volumes should also be outside of the PPRC configuration with the exception of LOGGER.
- A single point of failure should be avoided for the couple data set volumes by defining primary and alternate data sets.
- The volumes containing the primary couple data sets should be on the same disk subsystem as the primary PPRC volumes and the volumes containing the alternate couple data sets should be on the same subsystem as the secondary PPRC volumes.

Note: It is recommended that the couple data set volumes are in an LSS that is not in the PPRC configuration although this is not mandatory. They will not be affected by HyperSwap processing because only the devices in the PPRC configuration are swapped and although CGROUP FREEZE is performed at the LSS level, devices in SIMPLEX status are not affected.

Command authorization

The following RACF authority is required to issue Basic HyperSwap operator commands:

- SETHS ENABLE/DISABLE/SWAP requires RACF UPDATE authority to profile MVS.SETHS in the OPERCMDS class.
- DISPLAY HS requires RACF READ authority to profile MVS.DISPLAY.HS in OPERCMDS class.

Hardware reserve processing

Basic HyperSwap allows a hardware reserve to be transferred to the new primary device after the HyperSwap. The reserve will be released from the old primary device and obtained for the new primary device. It is the responsibility of the application that initially obtained the reserve to release it after the HyperSwap.

Attention: If a reserve is used to serialize a resource across multiple sysplexes, a system in a sysplex not participating in the HyperSwap that is holding the reserve will have serialization for the wrong resource after a HyperSwap.

Migration and coexistence

To be enabled for device monitoring and swap ready, Basic HyperSwap must be active on all members of the sysplex. Basic HyperSwap will be disabled if a system with z/OS 1.8 or lower is active in the sysplex.

22.3 Subchannel Sets for PPRC secondary devices

A critical constraint facing large z/OS customers today is the four-digit device number limit. There have been a number of enhancements to address this constraint. A new control unit feature called HyperPAV provided significantly improved utilization of the Parallel Access Volume alias (PAV-alias) devices, thereby reducing the number needed by an order of magnitude.

The z/OS V1R7 operating system release and the System z9 processor family have implemented a feature called Subchannel Sets, adding an additional 64K subchannels to the logical channel subsystem machine implementation. This was exploited by allowing the

customer to define the Parallel Access Volume aliases (PAV-aliases) in a separate number space, thereby allowing the number of logical volumes to grow (to 64K-256).

To allow System z customers to grow their data, additional significant exploitation of the alternate subchannel set is now possible for two special devices types:

- Non-PAV-alias devices for backing up of open systems data from the SAN.
- PPRC secondary devices.

22.3.1 Special devices

Special devices are not visible to applications and cannot be VARYed ONLINE or OFFLINE. They always reside in subchannel set 1 and have a 5-digit device number.

There are two types of special devices that can use subchannel set 1:

- 3370 FBA devices which are defined as device type 3390S to HCD.
- PPRC secondary devices which are defined as device type 3390D to HCD.

The 3370 FBA devices have the following characteristics:

- Never allowed online (no ECKD VTOC or volume label)
- Not allocated via JCL or SVC 99
- Device serialization via UCBNALOC processing (device offline and in-use by system component)
- IBM-provided DASD Error Recovery Procedure (ERP) is bypassed by the application (IOSNERP specified in the IOS block, IOSB, representing the request).

The special PPRC secondary devices have the following characteristics:

- The 4-digit device numbers for the primary and secondary devices must be the same.
- The associated primary device must be defined as a 3390B and have the same attributes as the secondary device. It is best to add and delete these PPRC pairs at the same time, and when you are in a non-swapped configuration.
- The MIH values will not be swapped during a swap.
- The devices are configured to come up offline.

Restriction: IPL and IODF secondary devices cannot be special devices. The current zArchitecture requires that the operating system explicitly enable the alternate subchannel sets before they can be used. An architecture change is required to allow IPL from an alternate subchannel set. Therefore, the customer will be required to use PPRC secondary devices in subchannel set 0 for the system data sets needed for IPL.

22.3.2 Exploitation by HyperSwap

z/OS device support code allows customers to place PPRC secondary devices in the alternate subchannel set and use HyperSwap-in-the-Base to dynamically switch devices concurrent to applications using them.

The PPRC configuration in Figure 22-8 is used to show how special PPRC secondary devices are configured and used by HyperSwap.

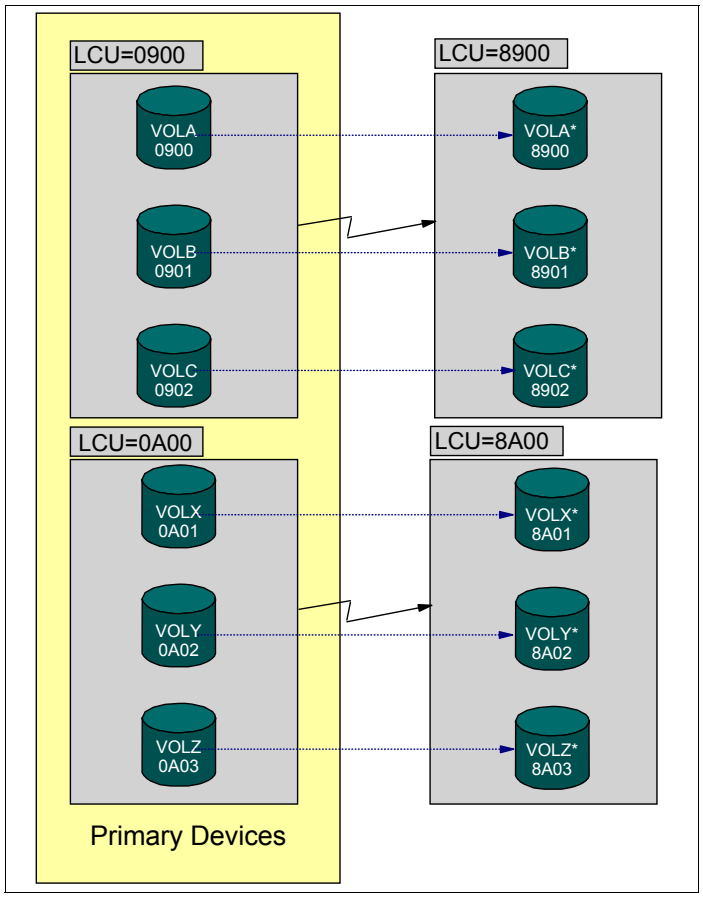


Figure 22-8 Current PPRC configuration

Current PPRC configurations

In current PPRC configurations, both the primary and secondary devices are defined in subchannel set 0. The primary devices are ONLINE while the secondary devices are OFFLINE and cannot be used by applications.

To allow the PPRC secondary devices to be defined in subchannel set 1, they must be defined as device type 3390D via HCD. In addition, the devices need to be renumbered so that the device number of the secondary in subchannel set 1 is the same as the primary in subchannel set 0, as shown in Figure 22-9 on page 469.

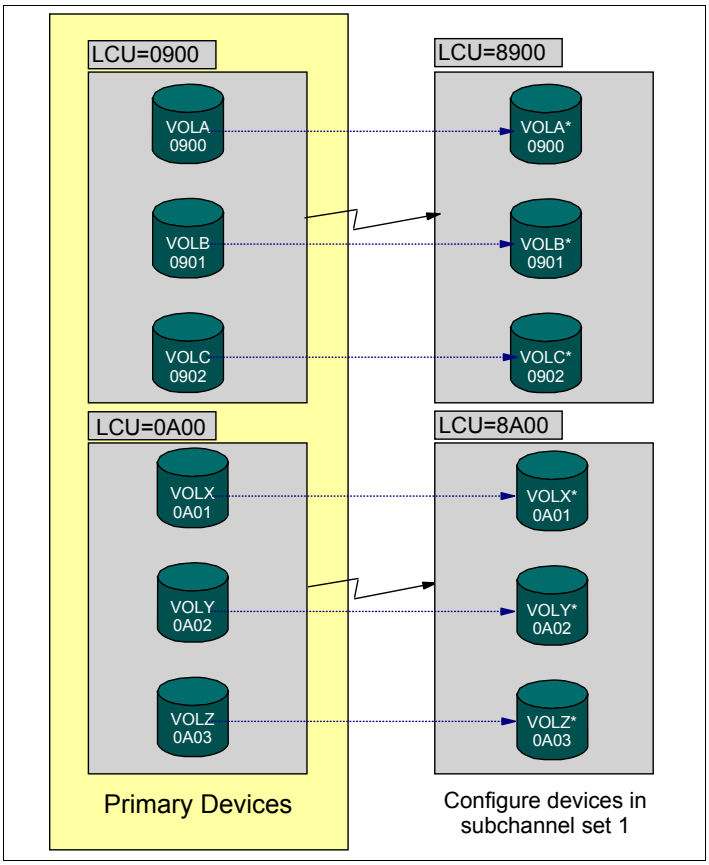


Figure 22-9 Swapping and special devices

Activity after a HyperSwap

After a HyperSwap, applications that were using the subchannel set 0 primary devices will use the new subchannel set 1 primary devices, as shown in Figure 22-10 on page 470.

IOS messages issued for these devices will continue to use the 4-digit device number. The subchannel set ID will not be displayed.

The IOS services UCBLOOK, UCBSCAN, and IOSCDR will find the device currently in use regardless of which subchannel set it resides on. After a swap, subchannel set 0 must be explicitly specified to find a device in subchannel set 0. See 22.3.4, “Updated services” on page 472.

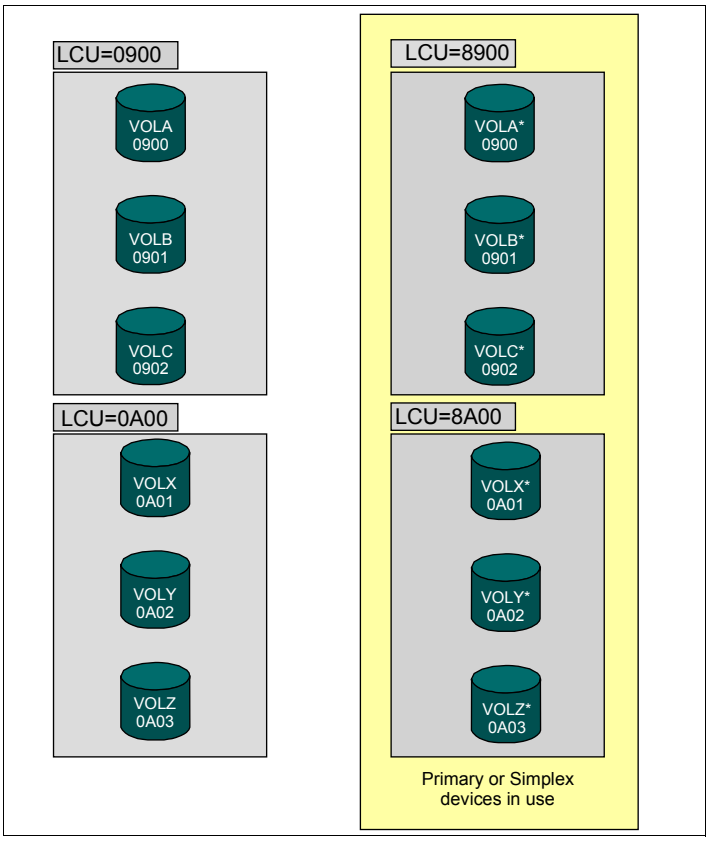


Figure 22-10 IPL after HyperSwap

22.3.3 Installation and activation

The SCHSET IODF statement in LOADxx is used to indicate which subchannel set is to be used for the primary devices. When the SCHSET parameter specifies:

- 0 - z/OS will use the special devices as offline secondary devices.
- 1 - The initial PPRC special secondary devices will be brought online as the new PPRC primary devices and the initial PPRC primary devices will remain offline as the new PPRC special secondary devices.

Message IEA111D

If the SCHSET parameter is not specified in LOADxx and at least one PPRC pair exists with the same device number, the following WTOR is issued to prompt the operator for a choice:

```
IEA111D SPECIFY SUBCHANNEL SET TO BE USED FOR DEVICES THAT ARE ACCESSIBLE FROM
MULTIPLE SUBCHANNEL SETS - REPLY SCHSET=X
```

IPL after a HyperSwap with following LOADxx

To IPL using the new primary devices in subchannel set 1 after a HyperSwap one of the following must be specified:

- SCHSET 1 in the IODF statement in LOADxx
- Subchannel set 1 in response to message IEA111D

An example of the LOADxx member is shown in Figure 22-11. The new statements are SCHSET and DYNCPADD, as follows:

SCHSET Enables you to request whether the special secondary devices in subchannel set 1 are used for the IPL or the normal base devices in subchannel set 0.

Specify 0 or 1 to indicate which subchannel set IOS should use for normal base devices that have a special secondary device with the same address.

DYNCPADD Indicates whether z/OS support for dynamic CPU addition is enabled.

Column Contents:

1-8 DYNCPADD

10-16 {ENABLE | DISABLE}

ENABLE If running on a machine that supports dynamic CPU addition (for example, z10), z/OS support for such addition is enabled, and an IPLed z/OS image can recognize such an addition. If not running on such a machine, this option is not relevant.

DISABLE z/OS does not support dynamic CPU addition. Even if running on a machine that supports dynamic CPU addition, an IPLed z/OS image cannot recognize such an addition. Default: ENABL.

```
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7
HWNAME   h1
LPARNAME l1
VMUSERID v1
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7
ARCHLVL  a
DYNCPAAD {ENABLE | DISABLE}
IEASYM  [xx]
        [(xx,yy,zz,...,L)]
INITSQA  xxxxK yyyyK
         xxxxM yyyyM
IODF     xx hiqualif configid id y
NUCLEUS  n
NUCLST   nn  y
PARMLIB  dsn                                          [volid]
                                                      [******]
                                                      [*MCAT*]
SCHSET   n
SYSCAT   volserxycsdsname                                      hlqtvc
SYSPARM [xx]
        [(xx,yy,zz,...,L)]
SYSPLEX  plexname
```

Figure 22-11 LOADxx changes for HyperSwap

Processing during IPL

Special devices are not included in the checking performed at IPL for duplicate device numbers. This prevents the following message from being issued:

```
IEA213A DUPLICATE VOLUME 'volser' FOUND ON DEVICES XXXX and YYYY
```

Table 22-1 shows which device will be brought online for the different combinations of device status and subchannel set.

Table 22-1 Combinations of device status and subchannel set

Subchannel Set 0 device status	Subchannel Set 1 device status	SCHSET is LOADxx or reply to IEA111D	Result
Primary	Secondary	0	Primary brought online
Primary	Secondary	1	Secondary brought online Primary is boxed
Secondary	Primary	0	Primary is boxed
Secondary	Primary	1	Primary brought online
Simplex	Simplex	0	Device in SS 1 not brought online
Simplex	Simplex	1	Device in SS 0 not brought online

22.3.4 Updated services

A number of IOS services have been updated to allow for special devices in subchannel set 1.

UCBLOOK service

A new keyword, SPECIAL=NO|YES, has been added with NO as the default.

Whenever an application issues UCBLOOK to find a device, the following new processing is performed:

- UCBLOOK will not return a SPECIAL device unless the application specifies the new SPECIAL=YES keyword.
- If the UCBLOOK invocation does not specify a subchannel set number, then the UCB representing the active primary device will be returned even if the actual subchannel connected to the UCB is in subchannel set 1. This will make the IOS UCB Swap process transparent to applications after a swap to the PPRC secondary has occurred.
- If the UCBLOOK invocation does specify a subchannel set number, the UCB connected to that subchannel set will be returned.

UCBSCAN service

A new keyword, SPECIAL=NO|YES|ONLY, has been added with NO as the default. Whenever an application issues UCBSCAN to search for a device, the following new processing is performed:

- UCBSCAN will not return SPECIAL devices unless the application specifies the new SPECIAL=YES keyword.
- UCBSCAN will return only SPECIAL devices if the application specified the SPECIAL=ONLY keyword.
- If the UCBSCAN invocation does not specify a subchannel set number, then the UCB representing the active primary will be returned even if the actual subchannel connected to the UCB is in subchannel set 1. This will make the IOS UCB Swap process transparent to applications after a swap to the PPRC secondary has occurred.

- If the UCBSCAN invocation does specify a subchannel set number, the UCBs connected to that subchannel set will be returned.

Table 22-2 shows which device numbers are returned by UCBSCAN for the different combinations of the SPECIAL keyword and the subchannel set that was used at IPL for 3390S devices and devices in a Special Pair (3390B/3390D) when the pair is NOT swapped.

The first two columns are the input that was specified. The last three columns indicate whether the device will be returned on the scan request. They represent the primary device (with a PPRC secondary), the SPECIAL PPRC secondary device, and a 3390S device.

Table 22-2 Table of UCBSCAN results for a non-swapped configuration

SPECIAL	Subchannel Set	0dddd - PPRC	1dddd - PPRC	3390S
SPECIAL=NO	Default SCHSET	YES	NO	NO
SPECIAL=NO	SCHSET=0	YES	NO	NO
SPECIAL=NO	SCHSET=1	NO	NO	NO
SPECIAL=NO	SUBCHANNELSET=ALL	YES	NO	NO
SPECIAL=YES	Default SCHSET	YES	NO	NO
SPECIAL=YES	SCHSET=0	YES	NO	NO
SPECIAL=YES	SCHSET=1	NO	YES	YES
SPECIAL=YES	SUBCHANNELSET=ALL	YES	YES	YES
SPECIAL=ONLY	Default SCHSET	NO	YES	YES
SPECIAL=ONLY	SCHSET=0	NO	NO	NO
SPECIAL=ONLY	SCHSET=1	NO	YES	YES
SPECIAL=ONLY	SUBCHANNELSET=ALL	NO	YES	YES

Table 22-3 shows which device numbers are returned by UCBSCAN for the different combinations of the SPECIAL keyword and the subchannel set that was used at IPL for 3390S devices and devices in a Special Pair (3390B/3390D) when the pair is swapped.

The first two columns are the input that was specified. The last three columns indicate whether the device will be returned on the scan request. They represent the primary device (with a PPRC secondary), the SPECIAL PPRC secondary device, and a 3390S device.

Table 22-3 Table of UCBSCAN results for a swapped configuration

SPECIAL	Subchannel Set	0dddd - PPRC	1dddd - PPRC	3390S
SPECIAL=NO	Default SCHSET	NO	YES	NO
SPECIAL=NO	SCHSET=0	YES	NO	NO
SPECIAL=NO	SCHSET=1	NO	NO	NO
SPECIAL=NO	SUBCHANNELSET=ALL	YES	NO	NO
SPECIAL=YES	Default SCHSET	NO	YES	NO

SPECIAL	Subchannel Set	0dddd - PPRC	1dddd - PPRC	3390S
SPECIAL=YES	SCHSET=0	YES	NO	NO
SPECIAL=YES	SCHSET=1	NO	YES	YES
SPECIAL=YES	SUBCHANNELSET=ALL	YES	YES	YES
SPECIAL=ONLY	Default SCHSET	YES	NO	YES
SPECIAL=ONLY	SCHSET=0	NO	NO	NO
SPECIAL=ONLY	SCHSET=1	NO	YES	YES
SPECIAL=ONLY	SUBCHANNELSET=ALL	YES	NO	YES

IOSCDR service

IOSCDR is changed to obtain the configuration data records for the active primary of a SPECIAL PPRC pair (3390B/3390D) if the SCHSET keyword is not specified. This will make the IOS UCB swap process transparent to applications after a swap to the PPRC secondary has occurred.

IOSCDR will obtain the configuration data records for the device in the subchannel set (SCHSET), which specifies whether or not a swap has occurred.

Most applications using this service will not need to change. If specific applications (vendor- or customer-written) need to access special devices using this service, they should be recompiled to use the new version of the macro.

IOSODS service

The IOSODS service is changed to support the DEVNCHAR, SCHSET, and LDEVNCHAR keywords.

The DEVNCHAR and LDEVNCHAR keywords are similar to the other UCB services that accept a device number. The input area contains a one-byte length field followed by the number of characters for the device number. For five-digit device numbers the first byte of the device number indicates the subchannel set number.

IOSODS will update the offline device for the active primary of a SPECIAL PPRC pair (3390B/3390D) if the SCHSET and LDEVNCHAR keywords are both not specified. This makes the IOS UCB swap process transparent to applications after a swap to the PPRC secondary has occurred.

The IOSODS service places the target offline device into a pseudo on-line state by initializing device self-description data, dynamic pathing, and other device-dependent functions. The device is marked as offline and in-use by system component (UCBNALOC) so it cannot be reused by another application. For 3390S and 3390D devices (Special), no VARY ENF signal is issued.

22.3.5 Dynamic activate

Until a HyperSwap occurs devices that are part of SPECIAL PPRC pairs can be added, deleted, or changed via separate dynamic activates.

After a HyperSwap occurs:

- If a Special Secondary (3390D) device is being added, a 3390B PPRC primary device with the same four-digit device number must also be added. If one device in a Special PPRC pair (3390B/3390D) is not being added when the other device in the pair is being added, the dynamic activate fails.
- If a Special Secondary (3390D) device is deleted, a 3390B PPRC primary device with the same four-digit device number must also be deleted. If one device in a Special PPRC pair (3390B/3390D) is not deleted when the other device in the pair is deleted, the dynamic activate fails.
- If a Special Secondary (3390D) device is changed, a 3390B PPRC primary device with the same four-digit device number must also be changed. If one device in a Special PPRC pair (3390B/3390D) is changed when the other device in the pair is not changed, the dynamic activate fails.

The following new messages are issued for dynamic activate failure associated with SPECIAL devices:

```
REASON=0148, CAN NOT DELETE SPECIAL PPRC DEVICE sdddd
     DESCTEXT=ASSOCIATED SPECIAL PPRC DEVICE sdddd IS NOT BEING DELETED

REASON=0149, CAN NOT ADD SPECIAL PPRC DEVICE sdddd
     DESCTEXT=ASSOCIATED SPECIAL PPRC DEVICE sdddd IS NOT BEING ADDED
```

If a new Special PPRC pair is added after a HyperSwap has occurred, the UCBs for devices in the pair are swapped. This allows the establishment of primary and secondary roles in the IODF for devices that match Special PPRC pairs already in the IODF. If it turns out that the swap of the added devices was not warranted, then, after the PPRC relationship of the added devices has been established, the VARY DEVICE,ONLINE command can be used to swap the UCBs back.

22.3.6 Migration and coexistence

Listeners for IOS ENF signals that include either device numbers or UCB addresses for PPRC primary and/or secondary devices need to qualify the device numbers for these devices with a subchannel set ID since they can now be in subchannel set 1.

Any listeners of these ENF signals may need to use the new SPECIAL(YES) keyword and the SCHSET keyword on UCBLOOK and/or UCBSCAN invocations in their code.

Changes required by ENF event code 28

Any use of UCBCHAN from the UCB addresses in the parameter list to identify a device will also need to reference UCBSSID to correctly identify the device. The UCBSSID can be referenced using the IOSDUPI mapping macro based on the UCBIEXT pointer.

Changes required by ENF event codes 31 and 32

For event codes 31 and 32, when processing the DCCD parameter list, if the DCCDETYP= DCCDDEV entries are being used, the DCCDETYP = DCCDSDEV should also be used. The DCCDSDEV entries are the same as the DCCDDEV entries but they have a DCCDETYP= DCCDDEVE entry immediately following them. The DCCDDEVN from the DCCDSDEV entry should be used in combination with the DCCDSSID from the DCCDDEVE entry to correctly identify the device.

Figure 22-12 shows the difference between the old DCCD device entries and the new DCCD device entries.

```
Old            New
DCCDDEV        DCCDDEV
DCCDDEV        DCCDSDEV
DCCDDEV        DCCDDEVE
               DCCDDEV

The DCCSDEV entry looks exactly like DCCDDEV but implies that DCCDDEVE
immediately follows.
```

Figure 22-12 DCCD device entries

Figure 22-13 shows sample code using the new DCCD device entries.

```
*
*    Process each array entry
*    Use UCBLOOK to get the UCB address for each added UCB
*
LOOP     DS    0H
         SLR   R10,R10             Clear subchannel set id
         MVC   THEDEVN,DCCDDEVN    Remember device number
         CLI   DCCDETYP,DCCDDEV    Device entry ?
         BE    CHKADD              Yes, check if add
         CLI   DCCDETYP,DCCDSDEV   Device entry in non-zero subchannel set?
         BNZ   NEXT                No, get next entry
LA    R04,DCCDELEN(R04)   Get to the next entry (DEVE entry)
         IC    R10,DCCDSSID        R10 contains subchannel set id
CHKADD   DS  0H                    determine if this is an add request
         CLI   DCCDEREQ,DCCDDADD   Add device entry ?
         BNZ   NEXT                No, get the next entry
         TM    DCCDEFLG,DCCDESFT   Software entry ?
         BNO   NEXT                No, get the next entry
*
*    Use UCBLOOK to find UCB address of added device:
*
         UCBLOOK DEVN=THEDEVN,UCBPTR=ADD_UCB,NOPIN,DYNAMIC=YES,  X
                 NONBASE=YES,RANGE=ALL,MF=(E,LOOKP),SCHSET=(R10) X
                 SPECIAL=YES
*
*    Process the added UCB
*
NEXT     DS    0H
         LA    R04,DCCDELEN(R04)   Get the next entry
         CR    R04,R05             Are we finished ?
         BL    LOOP                No, check out the entry
```

Figure 22-13 Sample code using the new DCCD device entries

Changes required by ENF event code 33

When processing the DACH parameter list, if the DACHQN= DACHPAV entries are being used, the DACHQN= DACHPAV_AS entries should also be used, and, if the DACHQN=DACHIORA entries are being used, the DACHQN= DACHIORA_AS entries

should also be used. The DACH_PAV_DEVN from the DACHPAV_AS entries should be used in combination with DACH_PAV_SSIDBase to correctly identify the device. The DACH_IORA_DEVN from the DACHIORA_AS entries should be used in combination with DACH_IORA_SSID to correctly identify the device; see Table 22-4.

Table 22-4 Table of DACH parameter list changes

DACHTYPE	DACHQN if device is in subchannel set 0	DACHQN if device is in subchannel set 1	Meaning
'LPE'	DACHIORA	DACHIORA_AS	I/O resource available
'PAVS'	DACHPAV	DACHPAV_AS	Change in PAV UCB

22.3.7 Updated commands

The following commands have been updated for special devices:

- D IOS,CONFIG
- D M=DEV
- VARY device
- V path

D IOS,CONFIG updates

If at least one PPRC secondary device is defined in subchannel set 1, the following lines will be displayed and are shown in Figure 22-14.

```
D IOS,CONFIG
  IOS506I 14.46.13 I/O CONFIG DATA 196
  ACTIVE IODF DATA SET = TREXEXP.IODF30
  CONFIGURATION ID = CONFIG00       EDT ID = 00
  TOKEN:  PROCESSOR DATE     TIME     DESCRIPTION
  SOURCE: XMP00    02-10-07 09:34:40
  ACTIVE CSS:  1   SUBCHANNEL SETS IN USE:  0, 1, 2, 3
  CHANNEL PATH MEASUREMENT FACILITY IS CURRENTLY ACTIVE
  CHANNEL MEASUREMENT BLOCK FACILITY IS ACTIVE
 [INITIAL SUBCHANNEL SET FOR PPRC PRIMARY: 0 | 1]
 [HYPERSWAP FAILOVER HAS OCCURRED: NO | YES]
```

Figure 22-14 D IOS,CONFIG display

D M=DEV updates

The system is to display the number of online channel paths to devices (including special devices) or a single channel path to a single device. The system indicates whether HyperSwap is enabled for the device and if the device has been swapped.

The following updates were made to the D M=DEV command:

- If a 4-digit device number is entered in the command, then the device information representing subchannel set 1 is displayed when the actual subchannel connected to the device is in subchannel set 1 (after a swap).
- If a 5-digit device number is entered, the device information representing the specified subchannel set is displayed. If a range of device numbers is found and one of the two numbers is a 5-digit number, the other number in the range must be 5-digit, too. If the

device is being monitored by HyperSwap, HS appears in the FUNCTIONS ENABLED line. If the device has been involved in a failover swap and the device is part of a PPRC pair that has a 3390D device, the DEVICE HAS BEEN SWAPPED line is displayed.

- If a range of device numbers is specified and the first number is 5 digits, the second number in the range must be 5 digits and vice versa.
- A SPECIAL device will have a status of SPECIAL.
- If the device is being monitored by HyperSwap, HS will appear in the FUNCTIONS ENABLED line.
- If the SPECIAL device has been involved in a failover swap, the following line will appear:

```
DEVICE HAS BEEN SWAPPED
```

Figure 22-15 shows the D M=DEV command output for a SPECIAL device.

```
SY1  d m=dev(10180)
SY1  IEE174I 15.51.34 DISPLAY M 986
DEVICE 0180   STATUS=SPECIAL
CHP                   34   39   46   47
DEST LINK ADDRESS     00   00   00   00
PATH ONLINE           Y    Y    Y    Y
CHP PHYSICALLY ONLINE Y    Y    Y    Y
PATH OPERATIONAL      Y    Y    Y    Y
MANAGED               N    N    N    N
CU NUMBER             .... .... .... ....
MAXIMUM MANAGED CHPID(S) ALLOWED:  0
DESTINATION CU LOGICAL ADDRESS = 00
SCP CU ND         = 002105.000.IBM.75.000000029377.0008
SCP TOKEN NED     = 002105.000.IBM.75.000000029377.0000
SCP DEVICE NED    = 002105.000.IBM.75.000000029377.0036
FUNCTIONS ENABLED = MIDAW, HS
DEVICE HAS BEEN SWAPPED
```

Figure 22-15 D M=DEV for a SPECIAL device

VARY device command updates

The VARY device command only allows three- or four-digit device numbers and operates on the perceived active device of a SPECIAL PPRC pair. If the VARY online fails due to this device being a secondary, a swap of the SPECIAL PPRC pair is performed and the following new message is issued:

```
IOS583I CAUSE OF FAILURE OF DEVICE dddd CORRECTED. REISSUE VARY COMMAND
```

A reissue of the VARY command should bring the primary device online.

VARY PATH command updates

The VARY PATH command is changed to allow a 5-digit logical device number: 1-digit subchannel set ID plus 4-digit device number.

- If a 4-digit device number is entered in the command, then the device information representing subchannel set 1 is varied when the actual subchannel connected to the device is in subchannel set 1 (after a swap).
- If a 5-digit device number is entered in the command, then the device information representing the specified subchannel set is varied. A device number is 3 to 5 hexadecimal digits, optionally preceded by a slash (/). You can precede the device

number with a slash to prevent ambiguity between the device number and a Coupling Facility name. Five-digit logical device numbers consist of a one-digit subchannel set ID plus a four-digit device number. If a 3-digit or 4-digit device number is entered in the command, then the device information representing subchannel set 0 is used for the display even if the actual subchannel connected to the device is in subchannel set 1. If a 5-digit device number is entered, the device information representing the specified subchannel set is displayed. If a range of device numbers is found and the first number is 5 digits, the second number in the range must be 5 digits and vice versa.

- If a range of device numbers is specified and the first number is 5 digits, the second number in the range must be 5 digits and vice versa.

22.3.8 Updated VARY PATH messages

The VARY PATH command messages are changed the same way as messages containing PAV-aliases in subchannel set 1 were changed in z/OS V1R7.

- If the device is in subchannel set 0, the message appears as it has in the past.
- If the device is in subchannel set 1, the message appears with the following two differences:
  - The message ID's 3-digit number is preceded by a "1".
  - The device number in the message is preceded by the 1-digit subchannel set ID.

Note: If the SSID of the device is non-zero, then the message ID becomes the IOS1nnnX or IEE1nnnX type.

Figure 22-16 shows how the VARY PATH messages have changed for SPECIAL devices with an additional 1 as in the message number and an s as the first digit of the device number.

```
IEE1169I VARY REJECTED, PATH(sdddd,xx) OFFLINE DUE TO insert
IEE1302I PATH(sdddd,XX) ONLINE
         PATH(sdddd,XX) ONLINE BY ESCM
         sdddd ONLINE
         sdddd ONLINE BY ESCM
IEE1303I PATH(sdddd,XX) OFFLINE
         sdddd OFFLINE
         PATH(sdddd,XX) OFFLINE BY ESCM
         sdddd OFFLINE BY ESCM
IEE1376I VARY REJECTED, PATH(sdddd,xx) LAST PATH TO DEVICE
IEE1378I VARY REJECTED, PATH(sdddd,xx) DOES NOT EXIST
IEE1379I VARY REJECTED, PATH(sdddd,xx) RESERVED
IEE1383I VARY REJECTED, DEVICE sdddd CANNOT BE ACCESSED
IEE1384I VARY REJECTED, UCB FOR DEVICE sdddd NOT CONNECTED
IEE1385I VARY REJECTED, I/O TIMED OUT DURING PATH(sdddd,xx)
IEE1491I PATH(sdddd,xx) NOT VARIED, FUNCTION CANCELLED BY OPERATOR
IEE1714I PATH(sdddd,xx) NOT OPERATIONAL
IEE1717D VARY NOT PROCESSED, sdddd STILL BUSY- REPLY EXTEND OR CANCEL
IOS1165I DEVICE sdddd.  PREFERRED PATHING NOW IN USE
         DEVICE sdddd.  PREFERRED PATHING NO LONGER IN USE
```

Figure 22-16 VARY PATH messages

Figure 22-17 shows other messages that have been updated in a way similar to the VARY PATH messages.

```
IOS1002A sdddd, NO PATHS AVAILABLE
IOS1275I  C.U.I.R. REQUEST TO QUIESCE THE FOLLOWING PATH(S):
             CHPID XX TO DEVICE(S) sxxxx,syyyy-szzzz,...
             CHPID YY TO DEVICE(S) sxxxx,syyyy-szzzz,...
IOS1278I  C.U.I.R. REQUEST TO RESUME THE FOLLOWING PATH(S):
             CHPID XX TO DEVICE(S) sxxxx,syyyy-szzzz,...
             CHPID YY TO DEVICE(S) sxxxx,syyyy-szzzz,...
IOS1279I C.U.I.R. REQUEST TO QUIESCE THE FOLLOWING DEVICE(S):
            sxxxx,syyyy-szzzz,...
IOS1280I C.U.I.R. REQUEST TO RESUME THE FOLLOWING DEVICE(S):
            sxxxx,syyyy-szzzz,...
IOS1282I C.U.I.R. QUIESCE REQUEST FAILED FOR THE FOLLOWING DEVICE(S):
            sxxxx,syyyy-szzzz,...
IOS1283I C.U.I.R. VARY PATH(sdddd,xx) REJECTED,
IOS1426I sdddd,pp, RESET ALLEGIANCE FAILURE
IOS1428I sdddd,pp, HAS BEEN RECOVERED THROUGH CHANNEL PATH zz
IOS1429I sdddd,pp, COULD NOT BE RECOVERED THROUGH AN ALTERNATE CHANNEL PATH
IOS1450E sdddd, cc
IOS1451I sdddd,
IOS1452I sdddd, cc
IOS1582I CHPID xx AUTOMATICALLY RECOVERED FOR DEVICE(S):
            sxxxx,syyyy-szzzz,...
```

Figure 22-17 Other updated messages

23

ISPF enhancements

This chapter describes the enhancements to ISPF in z/OS V1R10.

The following topics are discussed:

- Multiple destinations on the COPY and MOVE editor commands
- Enhanced ISPF logical screen swapping
- Block commands in the Data Set List Utility
- Enhancements to the z/OS UNIX Directory List Utility

23.1 Multiple destinations on COPY/MOVE line commands

Starting with z/OS V1R10, the ISPF editor supports specifying multiple targets on the **COPY** and **MOVE** line commands. This capability allows copying or moving a line or a group of lines in a data set to multiple destinations. The support for multiple targets on the **COPY** and **MOVE** commands is provided by new ISPF editor line commands.

The **A** (after), **B** (before), and **O** (over) line commands now indicate the final target of the **COPY** and **MOVE** line commands. The new line commands, which are **AK**, **BK**, and **OK**, specify the interim target destinations for the **A**, **B**, and **O** line commands, respectively.

New line commands

With ISPF in z/OS V1R10, the new line commands are:

AK When data is to be moved or copied, the **A** (after) line command specifies the line after which the data is to be placed. When data is to be moved or copied to multiple destinations, the **A** (after) line command specifies the final destination line after which the data is to be placed. When data is to be moved or copied to multiple destinations, the **AK** (after, multiple targets) line command specifies each multiple destination line (apart from the final destination line) after which the data is to be placed.

To specify multiple destinations for the data that is to be moved or copied:

Type `AK` in the line command field of each line (apart from the final destination) that the moved or copied data is to follow.

Type `A` in the line command field of the final line that the moved or copied data is to follow.

BK When data is to be moved or copied, the **B** (before) line command specifies the line before which the data is to be placed. When data is to be moved or copied to multiple destinations, the **B** (before) line command specifies the final destination line before which the data is to be placed. When data is to be moved or copied to multiple destinations, the **BK** (before, multiple targets) line command specifies each multiple destination line (apart from the final destination line) before which the data is to be placed.

To specify multiple destinations for the data that is to be moved or copied:

Type `BK` in the line command field of each line (apart from the final destination) that the moved or copied data is to precede.

Type `B` in the line command field of the final line that the moved or copied data is to precede.

OK When data is to be moved or copied and then overlaid on multiple destinations; where each destination is a single line:

The **OK** (overlay, intermediate target) line command specifies each intermediate destination (but not the final destination) for the data. You can type a number after the **OK** line command to specify the number of times that the **M** or **C** line command is to be performed. For example, typing the command `OK3` against a line causes the data to be moved or copied and then overlaid on that line and also the next two lines.

The **O** (overlay) line command specifies the final destination for the data. You can type a number after the **O** line command as previously described.

Where each destination is a block of lines:

The **OOK** (overlay, intermediate multiple-line target) line command specifies the first and last line of each intermediate destination (but not the final destination) for the data.

The **OO** (overlay, multiple-line target) line command specifies the first and last line of the final destination for the data.

Note: If you press Enter before specifying where you want the data to go, the editor displays a MOVE/COPY pending message at the top of the panel. The line does not move until you specify a destination.

The **AK** and **BK** line command indicates that another destination of the form **A**, **B**, or **O** is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

The **OK** and **OOK** line commands indicate that another destination of the form **A**, **B**, or **O** is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

Examples using the new line commands

To copy or move lines to multiple destinations, follow this procedure:

- Use the **COPY** or **MOVE** line commands to select the lines to be copied or moved.

  If you press Enter before specifying where you want the data to go, the editor displays a MOVE/COPY pending message at the top of the panel, as shown in Figure 23-1 on page 483. The line does not move until you specify a destination.

  The **AK** line command indicates that another destination of the form of **A**, **B**, or **O** is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PELEG.SRC(TEST10) - 01.04                    MOVE/COPY is pending
Command ===>                                               Scroll ===> HALF
****** ***************************** Top of Data ******************************
000001 CONTROL LIST CONLIST
000002
000003   LISTC ENT(PELEG.SRC)          ALL -
000004   LISTC ENT(PELEG.LOAD)         ALL -
000005   LISTC ENT(PELEG.LOGON.CLIST) ALL -
000006
000007
000008
M          CAT(UCAT.VSBOX01)
****** **************************** Bottom of Data ****************************
```

Figure 23-1 Multiple COPY/MOVE targets - select lines to copy

- Use the **AK**, **BK**, or **OK** line command, as shown in Figure 23-2 on page 484, to indicate the multiple interim targets to **COPY**/**MOVE** to.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PELEG.SRC(TEST10) - 01.04                    MOVE/COPY is pending
 Command ===>                                                Scroll ===> HALF
 ****** ***************************** Top of Data ******************************
 000001 CONTROL LIST CONLIST
 000002
 AK       LISTC ENT(PELEG.SRC)         ALL -
 AK       LISTC ENT(PELEG.LOAD)        ALL -
 000005   LISTC ENT(PELEG.LOGON.CLIST) ALL -
 000006
 000007
 000008
 M         CAT(UCAT.VSBOX01)
 ****** **************************** Bottom of Data ***************************
```

Figure 23-2 Multiple COPY/MOVE targets - indicate multiple targets

- Use the **A**, **B** or **O** line commands to indicate the final **COPY**/**MOVE** target and perform the copy or move action. We used the **A** line command to indicate the final target in the example in Figure 23-3 on page 484.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 EDIT       PELEG.SRC(TEST10) - 01.04                    MOVE/COPY is pending
 Command ===>                                                Scroll ===> HALF
 ****** ***************************** Top of Data ******************************
 000001 CONTROL LIST CONLIST
 000002
 AK       LISTC ENT(PELEG.SRC)         ALL -
 AK       LISTC ENT(PELEG.LOAD)        ALL -
 A00005   LISTC ENT(PELEG.LOGON.CLIST) ALL -
 000006
 000007
 000008
 M         CAT(UCAT.VSBOX01)
 ****** **************************** Bottom of Data ***************************
```

Figure 23-3 Multiple COPY/MOVE targets - indicate final target

Do the move

The final result is shown in Figure 23-4 on page 485 after your press the Enter key for the line commands shown in Figure 23-3 on page 484.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 EDIT       PELEG.SRC(TEST10) - 01.04                     Columns 00001 00072
 Command ===>                                               Scroll ===> HALF
 ****** ***************************** Top of Data ******************************
 000001 CONTROL LIST CONLIST
 000002
 000003   LISTC ENT(PELEG.SRC)         ALL -
 000004    CAT(UCAT.VSBOX01)
 000005   LISTC ENT(PELEG.LOAD)        ALL -
 000006    CAT(UCAT.VSBOX01)
 000007   LISTC ENT(PELEG.LOGON.CLIST) ALL -
 000008    CAT(UCAT.VSBOX01)
 000009
 000010
 000011
 ****** **************************** Bottom of Data ***************************
```

Figure 23-4 Multiple COPY/MOVE targets - result

The **AK**, **BK**, and **OK** line commands can be entered together to have multiple types of target commands in one **COPY**/**MOVE** operation.

Note: The **COPY**, **MOVE**, and **PASTE** primary commands do not support multiple targets. However, it is still possible to issue the **CUT** primary command and then **PASTE** multiple times to multiple locations.

23.2 Enhanced ISPF logical screen swapping

ISPF supports up to 32 logical screens. With only 2 logical screens, it is easy to change between logical screens by entering the **SWAP** command. With more than 2 logical screens the **SWAP NEXT** and **SWAP PREV** commands can be used to change between logical screens and finding the desired logical screen.

It is possible to simplify the process of changing logical screens by using the **SWAP LIST** command. The **SWAP LIST** command, shown in Figure 23-5 on page 486, opens a dialog displaying all currently open logical screens and allows changing to a logical screen by positioning the cursor on it in the list and then pressing Enter.

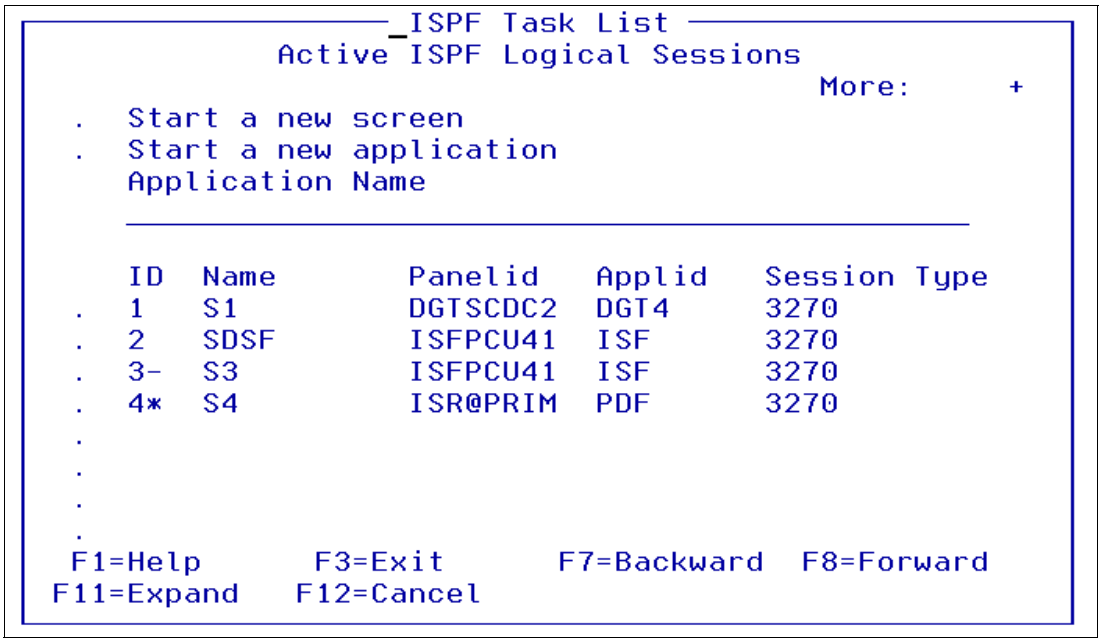


Figure 23-5 Swap List command panel to choose screen

23.2.1 New SWAPBAR command with z/OS V1R10

With z/OS V1R10, ISPF provides a new way to swap between logical screens, the ISPF SWAPBAR. The SWAPBAR is turned on using the **SWAPBAR ON** command. It is turned off using the **SWAPBAR OFF** command. The SWAPBAR appears in the bottom of the screen and provides a point-and-shoot field for every open logical screen. To change to a logical screen, simply place the cursor on the associated point-and-shoot field and press **Enter**.

SWAPBAR for enhanced screen swapping examples

Figure 23-6 on page 487 shows an example of the ISPF SWAPBAR. The SWAPBAR lists all the logical screens currently opened. The SWAPBAR shown is as follows:

- There are three screens shown in the SWAPBAR and the screen names shown are:

  S1 -CMD *SDSF

- An asterisk (*****) marks the current logical screen, as in ***SDSF**.
- A dash (**-**) indicates the opposite logical screen where you can use PF9 to switch to as the opposite logical screen or by using the **SWAP** command. The **-CMD** indicates the ISPF Option 6 panel.
- The **S1** is the third logical screen and is the ISPF Primary Option Menu.

Note: The purpose of the SWAPBAR is to easily switch to another logical screen by placing the cursor under the SWAPBAR entry and pressing **Enter**.

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
HQX7750 -----------------  SDSF PRIMARY OPTION MENU  -------------------------
COMMAND INPUT ===>                                          SCROLL ===> PAGE

DA    Active users                      INIT  Initiators
I     Input queue                       PR    Printers
O     Output queue                      PUN   Punches
H     Held output queue                 RDR   Readers
ST    Status of jobs                    LINE  Lines
                                        NODE  Nodes
LOG   System log                        SO    Spool offload
SR    System requests                   SP    Spool volumes
MAS   Members in the MAS
JC    Job classes                       RM    Resource monitor
SE    Scheduling environments           CK    Health checker
RES   WLM resources
ENC   Enclaves                          ULOG  User session log
PS    Processes

END   Exit SDSF

  S1       -CMD     *SDSF
```

Figure 23-6 ISPF SWAPBAR example

Example

The SWAPBAR shown in Figure 23-7 on page 487 is longer than the physical screen's display width. In such a case, the greater than (>) and less than (<) signs are used to indicate more logical screens exist, as shown circled in Figure 23-7 on page 487.

```
DSLIST   CMD      *SDSF    -OEDIT   READDOC  INFOPRT  S7       VIEW     >
```

Figure 23-7 ISPF SWAPBAR scroll right

Placing the cursor on the > or < signs and pressing Enter allows you to scroll right and left in the SWAPBAR. Figure 23-8 on page 487 shows the same SWAPBAR in Figure 23-7 again after scrolling to the right. The **BATCH**, **FORGRND**, and **DSUTIL** fields are now also visible and the **DSLIST**, **CMD**, and **SDSF** fields are not shown.

```
< OEDIT    READDOC  INFOPRT  S7       VIEW     -BATCH   FOREGRND *DSUTIL
```

Figure 23-8 ISPF SWAPBAR scroll left

The SWAPBAR settings are saved in the system ISPF profile variable ZSWPBR. The SWAPBAR settings are remembered between ISPF sessions.

Note: The **Always show split line** option on the ISPF settings panel is disabled when the SWAPBAR is active.

23.2.2 Changing logical screens using the mouse

Some 3270 emulators, such as IBM Personal Communications, allow you to configure the mouse buttons to perform certain terminal functions. Because the SWAPBAR fields are point-and-shoot fields, it is very convenient to double-click them with the mouse to change logical screen.

To configure IBM Personal Communications to be able to click on point-and-shoot fields, follow this procedure:

- In the menu bar, select **Edit** → **Preferences** → **Hotspots...**

  The **Hostspots Setup** dialog opens.
- Under the **Point-and-Select Commands** frame, select the **ENTER at cursor position** field, as shown in Figure 23-9 on page 488.
- Click **OK**.

  You can now change between ISPF logical screens by double-clicking fields in the ISPF SWAPBAR.

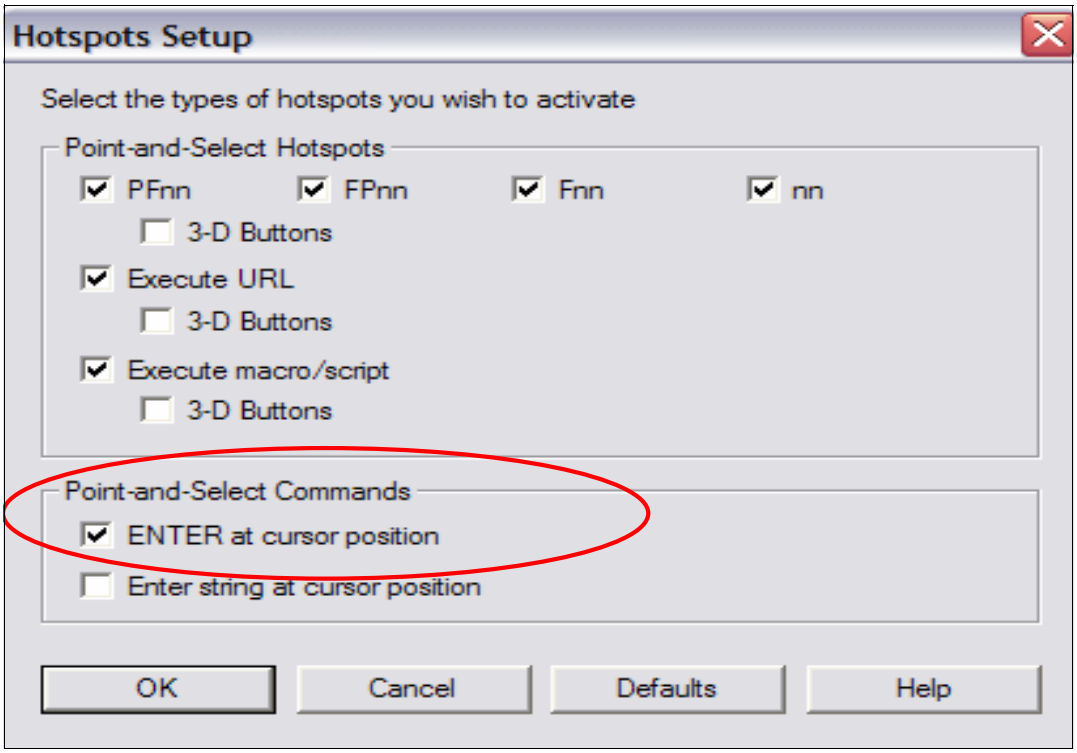


Figure 23-9 IBM Personal Communications Hotspots Setup dialog

Note: In our environment, we used IBM Personal Communications V5.7 20050906.

23.3 Block commands in DSLIST

The ISPF Data Set List Utility is one of the most heavily used utilities in ISPF. The Data Set List Utility is accessible through option 3.4 in the primary ISPF panel. It is also accessible using the ISPF command **`DSLIST`**. For example, to receive a list of all data sets prefixed with

SYS1, enter from any ISPF command line the DSLIST command as shown in Figure 23-10 on page 489.

```
DSLIST 'SYS1'
```

Figure 23-10 Using the DSLIST ISPF command to list data sets prefixed with SYS1

If **DSLIST** is entered with no additional parameters, a table listing all personal lists is displayed, allowing you to chose a personal list of data sets, as shown in Figure 23-11 on page 489. When browsing the list of data sets, **DSLIST** also allows you to issue commands against each data set in the list, for example to edit, view, or delete the data set.

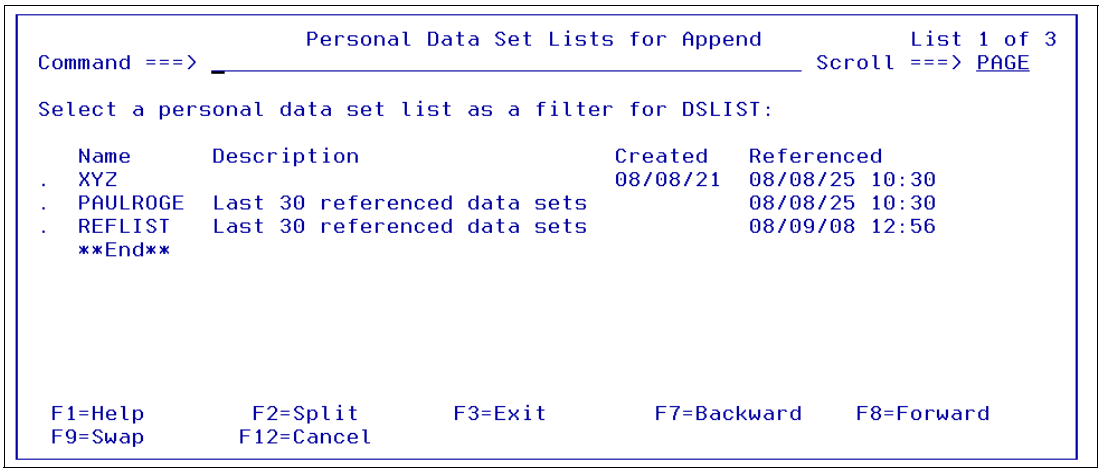


Figure 23-11 Personal Data Set Lists panel using the DSLIST command

Using line commands in a data set list

Starting with z/OS V1R10, ISPF **DSLIST** now allows you to issue commands against a block of data sets grouped together in the list. A block of line commands is marked by entering two forward slash characters (//) at the start and end of the block.

You must type the line command either immediately after the // on the first row of the block, or immediately after the // on the last row of the block. Figure 23-12 on page 490 shows an example of entering a delete command against four data sets in the list.

You can enter several blocks of commands at the same time, but you cannot nest them. Single line commands are not allowed within a block command. You can execute all line commands, including TSO commands, Clists and REXX execs as block commands. If you have selected the DSLIST settings option Execute Block Commands for excluded data sets, all applicable excluded rows are unexcluded before the block commands are executed.

```
 Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
 DSLIST - Data Sets Matching RC70                        Cursor not on choice
 Command ===>                                               Scroll ===> PAGE

 Command - Enter "/" to select action                Message          Volume
 ------------------------------------------------------------------------------
          RC70                                                        *ALIAS
          RC70.BRODCAST                                               SBOXE0
          RC70.LOG.MISC                                               SBOXE4
          RC70.LOGON.CLIST                                            SBOXE3
 //d      RC70.PTF.UA03287                                            SBOX20
          RC70.PTF.UA03288                                            SBOX20
          RC70.PTF.UA03338                                            SBOXFF
          RC70.PTF.UA03339                                            SBOXFH
          RC70.PTF.UK16029                                            SBOXBF
          RC70.PTF.UK16031                                            SBOXFF
          RC70.PTF.UQ79132                                            SBOXA8
 //       RC70.PTF.UW82817                                            SBOXB6
          RC70.SC70.ISPF42.ISPPROF                                    SBOXE2
***************************** End of Data Set list ***************************
```

Figure 23-12 DSLIST block command example - delete command

The result of the delete command is shown in Figure 23-13 on page 490.

```
 Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
 DSLIST - Data Sets Matching RC70                        Data set deleted
 Command ===>                                               Scroll ===> PAGE

 Command - Enter "/" to select action                Message          Volume
 ------------------------------------------------------------------------------
          RC70                                                        *ALIAS
          RC70.BRODCAST                                               SBOXE0
          RC70.LOG.MISC                                               SBOXE4
          RC70.LOGON.CLIST                                            SBOXE3
          RC70.PTF.UA03287                            Deleted
          RC70.PTF.UA03288                            Deleted
          RC70.PTF.UA03338                            Deleted
          RC70.PTF.UA03339                            Deleted
          RC70.PTF.UK16029                            Deleted
          RC70.PTF.UK16031                            Deleted
          RC70.PTF.UQ79132                            Deleted
          RC70.PTF.UW82817                            Deleted
          RC70.SC70.ISPF42.ISPPROF                                    SBOXE2
***************************** End of Data Set list ***************************
```

Figure 23-13 DSLIST block command example - result

Excluded data sets

Using the **X** line command in **DSLIST** allows you to exclude data sets from the list. You may want to enter a command against a block of data sets that has excluded data sets in it. The

DSLIST settings panel, which is accessible through **Options** → **DSLIST Settings**, provides a new option to control whether to execute block commands against excluded data sets in a block or not. Block commands can only be entered on non-excluded lines. Figure 23-14 shows the new option in the DSLIST settings panel.

```
 File  Colors  Workstation  Help
-----------------------------------------------------------------
                  Data Set List Settings Main

   General Options
   Enter "/" to select option
   /  Display Edit/View entry panel (*)
   /  Automatically update reference lists
   /  List pattern for MO, CO, D, and RS actions
   /  Show status for MO, CO, D, and RS actions
   /  Confirm Member delete
   /  Confirm Data Set delete
   /  Do not show expanded command
   /  Enhanced member list for Edit, View, and Browse
   /  Display Total Tracks
   /  Execute Block Commands for excluded Data Sets
      Display Expiration Date

 (*) Requires enhanced member list option to be selected

 Press EXIT to save, CANCEL to cancel changes.
```

Figure 23-14 New option on the DSLIST settings panel

Example of excluded data sets

Using the **x** line command for data sets in the DSLIST panel indicates that when using the block delete, these data sets will not be included in the delete.

Figure 23-15 on page 491 shows the **x** action character for the two data sets that are not going to be deleted.

```
DSLIST - Data Sets Matching RC70                                Row 1 of 15
Command ===>                                                 Scroll ===> PAGE

Command - Enter "/" to select action                Message          Volume
------------------------------------------------------------------------------
         RC70.PTF.UA03287                                              SBOX20
         RC70.PTF.UA03288                                              SBOXE0
 x       RC70.PTF.UA03338                                              SBOXA7
 x       RC70.PTF.UA03339                                              SBOXB5
         RC70.PTF.UK16029                                              SBOXFG
         RC70.PTF.UK16031                                              SBOXFF
         RC70.PTF.UQ79132                                              SBOXA8
         RC70.PTF.UW82817                                              SBOXBF
```

Figure 23-15 Place an x as an action character to not delete the data set

After pressing the **Enter** key for the action shown in Figure 23-15 on page 491, Figure 23-16 on page 492 shows the result.

```
DSLIST - Data Sets Matching RC70                                  Row 1 of 15
Command ===>                                                   Scroll ===> PAGE

Command - Enter "/" to select action                Message          Volume

------------------------------------------------------------------------------
         RC70.PTF.UA03287                                                SBOX20
         RC70.PTF.UA03288                                                SBOXE0
          - -  -  -  -  -  -  -  -  -  -  -       2 data set(s) not displayed
         RC70.PTF.UK16029                                                SBOXFG
         RC70.PTF.UK16031                                                SBOXFF
         RC70.PTF.UQ79132                                                SBOXA8
         RC70.PTF.UW82817                                                SBOXBF
```

Figure 23-16 Showing the two data sets not to be deleted

Figure 23-17 on page 492 shows the block delete command.

```
DSLIST - Data Sets Matching RC70                                  Row 1 of 15
Command ===>                                                   Scroll ===> PAGE

Command - Enter "/" to select action                Message          Volume

------------------------------------------------------------------------------
//d      RC70.PTF.UA03287                                                SBOX20
         RC70.PTF.UA03288                                                SBOXE0
          -  -  -  -  -  -  -  -  -  -  -         2 data set(s) not displayed
         RC70.PTF.UK16029                                                SBOXFG
         RC70.PTF.UK16031                                                SBOXFF
         RC70.PTF.UQ79132                                                SBOXA8
//       RC70.PTF.UW82817                                                SBOXBF
```

Figure 23-17 Entering the block delete command

Figure 23-18 on page 492 shows the result of the block delete with the two excluded data sets not being deleted.

```
DSLIST - Data Sets Matching RC70                                  Row 1 of 15
Command ===>                                                   Scroll ===> PAGE

Command - Enter "/" to select action                Message          Volume
------------------------------------------------------------------------------
         RC70.PTF.UA03338                                                SBOXA7
         RC70.PTF.UA03339                                                SBOXB5
```

Figure 23-18 Data sets that were excluded from the block delete

Order of line commands

The order of command execution is changed in z/OS V1R10 to make it more consistent and less error prone. Before z/OS V1R10, the exclude/non-exclude line commands were executed first, even before the scroll commands and primary commands, but all other line commands were executed last. This resulted in some unexpected behavior during scrolling or command execution.

With z/OS V1R10, data set list line commands are executed in the following order:

1. Primary commands
2. Scrolling commands
3. Exclude/non-exclude commands
4. Block line commands
5. Other line commands

23.4 z/OS UNIX directory list utility enhancements

z/OS V1R10 ISPF provides an interface to the z/OS UNIX directory list utility that client applications and ISPF users can use to display the z/OS UNIX directory list from any command line using TSO and ISPF commands and services. This interface must be independent of the type of connection, for example HTTP, between the client and z/OS host. This capability is provided by the SCLM Developer Toolkit product for products and applications which requires SCLM Developer Toolkit to be installed.

In z/OS V1R10, ISPF is enhanced as follows:

- To make the z/OS UNIX Directory List Utility (option 3.17 from the primary ISPF panel) more easily accessible to users and application writers.
- A new ISPF system **`UDLIST`** command allows users to access a Personal Data Set Lists for UDLIST panel from any ISPF command line.
- A new ISPF service, **`DIRLIST`**, provides application writers with a programmatic way to invoke the z/OS UNIX Directory List Utility from a CLIST, REXX exec, or program.

You can use any personal data set list to create complex lists of z/OS UNIX files and directories, similar to those displayed using ISPF option 3.17 (the z/OS UNIX Directory List Utility). The easiest way to use a Personal Data Set List to create a z/OS UNIX list is to type `UDLIST listname` on an ISPF command line (listname is the name of the personal data set list). ISPF uses each pathname entry in the personal list to build the displayed list of files and directories.

23.4.1 ISPF service to display a z/OS UNIX directory list

ISPF in z/OS V1R10 provides a new service to allow application writers to invoke the z/OS UNIX directory list utility from any command line. The new service, **`DIRLIST`**, can be invoked using ISPEXEC and in call format. Figure 23-19 shows the command invocation format of the service.

```
ISPEXEC DIRLIST PATH(path-var)
                [CONFIRM(YES|NO)]
                [CONFDRD(YES|NO)]
                [PANEL(panel-name)]
                [COLS(column-list)]
                [FIXCOLS(YES|NO)]
                [LCMDS(line-command-list)]
```

Figure 23-19 DIRLIST service - command invocation format

Figure 23-20 shows the call invocation format of the service.

```
CALL ISPLINK  ('DIRLIST ',
              path-var
              ,['YES     '|'NO      ']
              ,['YES     '|'NO      ']
              ,[panel-name]
              ,[column-list]
              ,['YES     '|'NO      ']
              ,[line-command-list];
```

Figure 23-20 DIRLIST service - call invocation format

DIRLIST uses the following parameters:

PATH This refers to the name of an ISPF variable containing the pathname for a z/OS UNIX directory.

CONFIRM This controls whether the Confirm Delete panel appears when using the **D** line command to delete a file.

CONFDRD This controls whether the Confirm Non-empty Directory Delete panel appears when using the **D** line command to delete a directory containing files.

PANEL This refers to the name of the panel used for the display of the directory list. The default is the ISPF directory list panel ISRUUDL0.

COLS This specifies the columns of data displayed on the z/OS UNIX Directory List and their width. If omitted, the column arrangement defined by the user is used. If an asterisk (*) is specified, the ISPF-defined default column arrangement is used. The list of columns is described in Table 23-1 on page 494.

FIXCOLS This controls whether the user is able to change the column arrangement for the directory list display.

LCMDS This allows the caller to process the line command entered on the z/OS UNIX Directory List display. The caller specifies a directory list command processor and a list of line commands to be processed by the line command processor rather than the ISPF command processor.

Table 23-1 Columns on the z/OS UNIX Directory List Utility

Column	Abbreviation for COLS parameter	Maximum Width
Type	TY	4
Permissions	PE	10
Permissions - Octal	PO	4
Audit	AU	6
Extended Attributes	EX	4
Format	FM	4
Owner	OW	8
Group	GR	8
Links	LI	14

Column	Abbreviation for COLS parameter	Maximum Width
Size	SZ	20
Modified Date/Time	MD	19
Changed Date/Time	CH	19
Accessed Date/Time	AC	19
Created Date/Time	CR	19

DIRLIST REXX example

Figure 23-21 shows an example of invoking **DIRLIST** from a REXX exec. The COLS parameter is used to indicate only the Permissions, Owner, Group and Size columns are to be displayed with width of 10, 8, 8 and 7 characters respectively. The LCMDS parameter is used to indicate that the command processor LCPROC is to be invoked for the **LL**, **B**, and **UPD** line commands.

```
/* REXX */

dir = '/u/peleg'

address ISPEXEC
"DIRLIST PATH(dir) COLS(PE,10,OW,8,GR,8,SZ,7) LCMDS(LCPROC,LL,B,UPD)"
```

Figure 23-21 REXX program to invoke DIRLIST

The z/OS UNIX Directory List panel that is a result of executing the REXX exec is shown in Figure 23-22 on page 496.

```
   Menu  Utilities  View  Options  Help
------------------------------------------------------------------------------
                           z/OS UNIX Directory List         Row 13 to 26 of 26
 Command ===>                                                 Scroll ===> CSR

 Pathname . : /u/peleg

 Command  Filename        Message          Permission Owner    Group    Size
 ------------------------------------------------------------------------------
          mylibar.a                        rw-r--r--  PELEG    SYS1       61648
          mylibar.c                        rwx------  PELEG    SYS1          88
          mylibar.o                        rw-r--r--  PELEG    SYS1       61440
          mylibar.x                        rw-r--r--  PELEG    SYS1          80
          out1                             rw-r--r--  PELEG    SYS1      723816
          setupJSec.sh                     rwx------  PELEG    SYS1         269
          temp                             rw-r-----  PELEG    SYS1        8400
          temp2                            rw-r--r--  PELEG    SYS1          22
          tryar                            rwxr-xr-x  PELEG    SYS1       73728
          tryar.c                          rwx------  PELEG    SYS1         128
          tryar.o                          rw-r--r--  PELEG    SYS1        4640
TestJSec.class                  rw-r--r--  PELEG    SYS1        1504
          TestJSec.java                    rwx------  PELEG    SYS1        1012
 ****************************** Bottom of data ********************************
```

Figure 23-22 DIRLIST panel resulting from the REXX program

23.4.2 Displaying z/OS UNIX directory list from the command line

UDLIST is an ISPF system command added in z/OS V1R10. You can enter the **UDLIST** command from any ISPF command line to start the ISPF z/OS UNIX directory list utility. The syntax of the **UDLIST** command is as follows:

```
UDLIST [listname|pathname]
```

The optional parameters of the **UDLIST** command are:

listname This is the name of a personal data set list containing entries for z/OS UNIX files and directories. These entries are used to build the displayed list.

pathname This is path name of a z/OS UNIX directory.

The pathname parameter specifies a z/OS UNIX directory name which is case sensitive. Therefore, you must enter the command from an ISPF panel that supports case-sensitive commands such as the ISPF Command Shell panel (option 6 from the primary ISPF panel). A case-sensitive command line can be accessed from any ISPF command line by entering the **CMDE** ISPF system command. The **CMDE** command opens an ISPF command entry panel, which is case sensitive.

UDLIST command with no parameters

Alternatively, you can enter the **UDLIS**T command with no parameters. A pop-up window is opened that prompts you to select a personal list or to specify a path name, as shown in Figure 23-23 on page 497.

```
                    Personal Data Set Lists for UDLIST          List 1 of 2
Command ===>                                                 Scroll ===> PAGE

Specify the pathname for a z/OS UNIX directory:
Name . . . . .                                                          +

Or select a personal data set list as a filter for UDLIST:

   Name      Description                   Created   Referenced
.  PAULROGE  personal list                 07/09/05  08/08/21 10:26
.  REFLIST   Last 30 referenced data sets            08/08/21 10:42
   **End**
```

Figure 23-23 UDLIST pop-up window

UDLIST command with parameters

Alternatively, you can enter the **UDLIS**T command with parameters. A pop-up window is opened showing the personal list specified in the command, as follows:

```
UDLIST PAULROGE
```

```
  Menu  Utilities  View  Options  Help
--------------------------------------------------------------------------------
                          z/OS UNIX Directory List               List 1 of 3
Command ===>                                                 Scroll ===> PAGE

List . . . : PAULROGE

Command  Pathname        Message          Type Permission Audit  Ext  Fmat
--------------------------------------------------------------------------------
________ /u/rogers                        Dir  rwx------  fff--- --s- nl
________ /u/rogers/cbprn                  File rwx------  fff--- --s- nl
________ /u/rogers/gener                  File rwx------  fff--- --s- nl
   **End**
```

Figure 23-24 UDLIST command specifying a listname

Tip: You can easily access your z/OS UNIX home directory list from any ISPF panel command line, even if it is not case sensitive, by entering **UDLIST ~**. The ~ sign represents the home directory defined in your RACF OMVS segment.

Personal Data Set List panel support

A personal data set lists upto 30 data set names and z/OS UNIX files. For data sets, each name can include a member name or a volume name, or both. z/OS UNIX file path names can be for regular files, directories, or symbolic links to directories or regular files. Personal data set lists can also contain workstation file names and data set name levels.

Personal data set lists are a good way to group (by project, for example) those data sets and z/OS UNIX file path names that you use frequently. You can use personal data set lists to avoid typing in data set names and z/OS UNIX file path names and to create customized data set lists similar to those using ISPF Option 3.4.

To create or use a personal data set list, perform one of the following two actions:

1. Select the Personal Data Set List choice from the RefList pull-down. Enter either Option 1 or 2, as shown in Figure 23-25 on page 498, to select a Personal Data Set List.

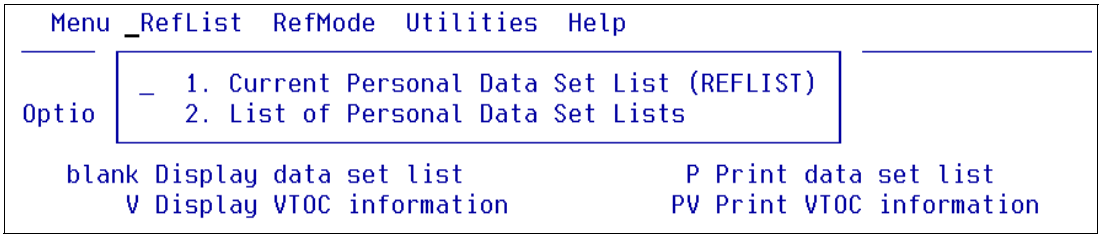


Figure 23-25 Using the RefList pulldown to select a Personal Data Set List

Selecting Option 1 displays Figure 23-26 on page 498.

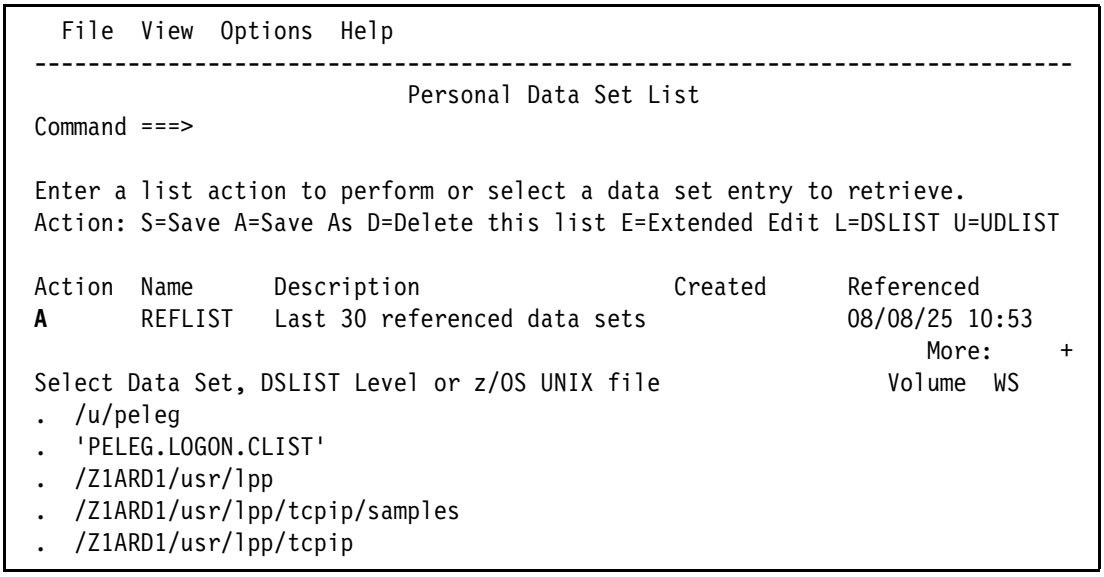


Figure 23-26 Personal Data Set List panel

Entering `A` saves the list with a specified list name, as shown in Figure 23-27 on page 498.

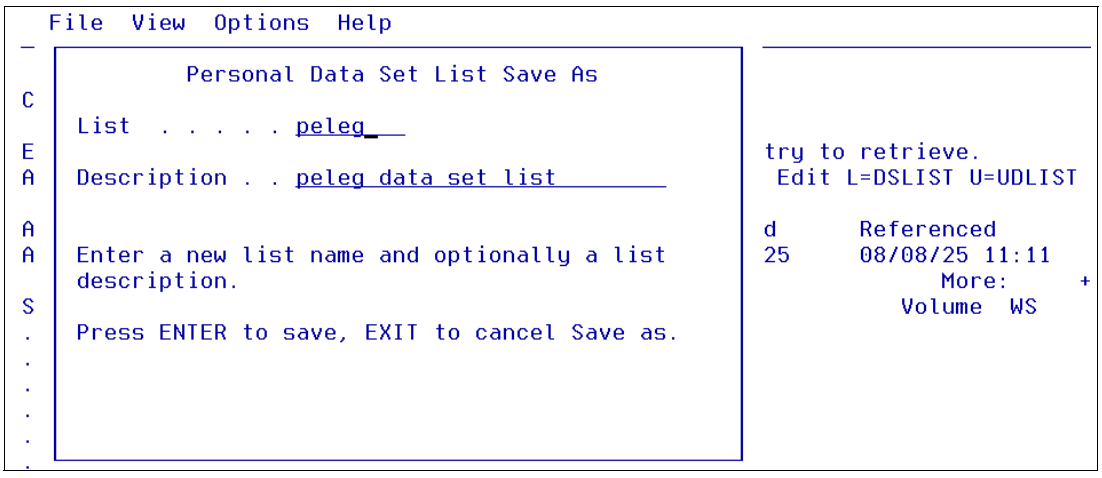


Figure 23-27 Create a list name to save Personal Data Set List with a new name

Selecting Option 2 from the Reflist pulldown (shown in Figure 23-25 on page 498) displays the current Personal Data Set List panel now with a new name of PELEG, as shown in Figure 23-28 on page 499.

```
  File  View  Options  Help
--------------------------------------------------------------------------------
Active: PELEG              Personal Data Set Lists               List 1 of 3
Command ===>                                                  Scroll ===> PAGE

Action: O=Open  A=Save As  D=Delete  E=Edit  L=DSLIST  U=UDLIST

   Name             Description                    Created   Referenced
_  ABC              new list peleg                 08/09/08  08/09/08 16:57
u  PELEG            list peleg                     08/09/08  08/09/08 16:59
_  REFLIST          Last 30 referenced data sets             08/09/08 16:59
   **End**
```

Figure 23-28 Option 2 from the RefList pulldown displays this list

Entering u next to PELEG displays Figure 23-29 on page 499.

```
  Menu  Utilities  View  Options  Help
--------------------------------------------------------------------------------
                           z/OS UNIX Directory List              List 1 of 16
Command ===>                                                  Scroll ===> CSR

List . . . : PELEG

Command  Pathname        Message          Type Permission Audit  Ext  Fmat
--------------------------------------------------------------------------------
         /                                Dir  rwxr-xr-x  fff---
         /u/peleg                         Dir  rwxr-xr-x  fff---
         /u/peleg/aaaa                    File rwx------  fff--- --s- nl
         /u/peleg/bapi.c                  File rwx------  fff--- --s- nl
         /u/peleg/bapi2.                  File rwx------  fff--- --s- nl
         /u/peleg/myliba                  File rwx------  fff--- --s- nl
         /u/peleg/setupJ                  File rwx------  fff--- --s- nl
         /u/peleg/temp                    File rw-r-----  fff--- --s- nl
         /u/peleg/temp2                   File rw-r--r--  fff--- --s- nl
         /u/peleg/tryar.                  File rwx------  fff--- --s- nl
         /u/peleg/CEEDUM                  File rw-rw-rw-  fff--- --s- nl
         /u/peleg/TestJS                  File rwx------  fff--- --s- nl
         /SC70/var                        Dir  rwxrwxrwt  fff---
         /Z1ARD1/usr/lpp                  Dir  rwxr-xr-x  fff---
         /Z1ARD1/usr/lpp                  Dir  rwxr-xr-x  fff---
         /Z1ARD1/usr/lpp                  Dir  rwxr-xr-x  fff---
   **End**
```

Figure 23-29 Display of Personal Data Set List panel

2. A second way to display the Personal Data Set List panel is to Enter the **REFACTD** command from the command line. This command displays the same panel shown in Figure 23-29 on page 499.

23.4.3 New line action command U=UDLIST

With z/OS V1R10, the Personal Data Set List panel now supports a new line action command, **U**, as shown in Figure 23-30 on page 500 to open the UDLIST panel for a z/OS UNIX file or directory contained in a personal list. This is shown in Figure 23-31 on page 500.

```
  File  View  Options  Help
-------------------------------------------------------------------------------
                           Personal Data Set List            UDLIST processed
 Command ===>

 Enter a list action to perform or select a data set entry to retrieve.
 Action: S=Save A=Save As D=Delete this list E=Extended Edit L=DSLIST U=UDLIST

 Action  Name      Description                  Created      Referenced
 U       PELEG1    new peleg                    08/09/08     08/09/08 17:41
                                                                 More:     +
 Select Data Set, DSLIST Level or z/OS UNIX file                Volume  WS
 .  /u/peleg
 .  'PELEG.LOGON.CLIST'
 .  /Z1ARD1/usr/lpp
 .  /Z1ARD1/usr/lpp/tcpip/samples
 .  /Z1ARD1/usr/lpp/tcpip
 .  /SC70/var
 .  /u/peleg/aaaa
```

Figure 23-30 UDLIST command specifying a listname

```
  Menu  Utilities  View  Options  Help
-------------------------------------------------------------------------------
                           z/OS UNIX Directory List              List 1 of 16
 Command ===>                                                 Scroll ===> CSR

 List . . . : PELEG1

 Command  Pathname        Message          Type Permission Audit  Ext  Fmat
 -------------------------------------------------------------------------------
          /                                 Dir  rwxr-xr-x  fff---
          /u/peleg                          Dir  rwxr-xr-x  fff---
          /u/peleg/aaaa                     File rwx------  fff--- --s- nl
          /u/peleg/bapi.c                   File rwx------  fff--- --s- nl
          /u/peleg/bapi2.                   File rwx------  fff--- --s- nl
          /u/peleg/myliba                   File rwx------  fff--- --s- nl
          /u/peleg/setupJ                   File rwx------  fff--- --s- nl
          /u/peleg/temp                     File rw-r-----  fff--- --s- nl
          /u/peleg/temp2                    File rw-r--r--  fff--- --s- nl
          /u/peleg/tryar.                   File rwx------  fff--- --s- nl
RA        /Z1ARD1/usr/lpp                   Dir  rwxr-xr-x  fff---
          /u/peleg/CEEDUM                   File rw-rw-rw-  fff--- --s- nl
          /u/peleg/TestJS                   File rwx------  fff--- --s- nl
**End**
```

Figure 23-31 Result from the UDLIST commands (U)

New RA line command

A new line command, **RA**, is added to the z/OS UNIX Directory List panel to allow adding z/OS UNIX files and directories to personal directories, as shown in Figure 23-31 on page 500.

A pop-up window opens, prompting you for a personal list name to add the z/OS UNIX file or directory to, as shown in Figure 23-32 on page 501.

```
   Personal Data Set List Add

Enter a Personal List Name:

List Name  xyz. . .




Press ENTER to add data set.
Press CANCEL to cancel Refadd.

 F1=Help           F2=Split
 F3=Exit           F7=Backward
```

Figure 23-32 Add a listname to add the directory to

The List Name **xyz** is added to the pop-up window. After pressing Enter, added to the new Personal Data Set List **xyz** is the pathname **/Z1ARD1/usr/lpp**, as shown in Figure 23-33 on page 501.

Use the **UDLIST** command to display the new listname, as shown in Figure 23-33 on page 501.

```
                  Personal Data Set Lists for UDLIST          List 1 of 3
Command ===>                                                Scroll ===> PAGE

Specify the pathname for a z/OS UNIX directory:
Name . . . . .                                                          +

Or select a personal data set list as a filter for UDLIST:

   Name      Description                   Created   Referenced
S  XYZ                                     08/08/21  08/08/21 13:02
.  PAULROGE  Last 30 referenced data sets             08/08/21 12:31
.  REFLIST   Last 30 referenced data sets             08/08/21 13:05
   **End**
```

Figure 23-33 Using the UDLIST command to display the listname just created

Then, by selecting XYZ in the list of names, Figure 23-34 on page 502 is displayed showing the pathname.

```
  Menu  Utilities  View  Options  Help
--------------------------------------------------------------------------------
                            z/OS UNIX Directory List               List 1 of 1
 Command ===>_____________________________________________________ Scroll ===> PAGE

 List . . . : PELEG1

 Command  Pathname        Message          Type Permission Audit  Ext  Fmat

--------------------------------------------------------------------------------
L________ /Z1ARD1/usr/lpp                 Dir  rwxr-xr-x  fff---
    **End**
```

Figure 23-34 Directory list for XYZ

Entering a list command, **L**, for the new pathname in XYZ, Figure 23-35 is displayed showing the path to filename Ja_JP which was selected using the **RA** line command in Figure 23-30 on page 500.

```
   Menu  Utilities  View  Options  Help
 --------------------------------------------------------------------------------
                            z/OS UNIX Directory List               List 1 of 3
 Command ===> _____________________________________________________ Scroll ===> PAGE

 Pathname . : /Z1ARD1/usr/lpp

 Command  Filename        Message          Type Permission Audit  Ext  Fmat
 --------------------------------------------------------------------------------
_________ .                               Dir  rwxr-xr-x  fff---
_________ ..                              Dir  rwxr-xr-x  fff---
_________ IBM                             Dir  rwxr-xr-x  fff---
    **End**
```

Figure 23-35 Listing of filenames for the added the new pathname in XYZ

24

Service aids enhancements

MVS service aids are a group of tools to aid in detecting problems and gathering documentation needed to solve the causes, and to communicate with support locations remote from where materials may have originally been collected. This chapter describes the following service aids enhancements:

- Hung SDUMP detection
- SDUMP serviceability enhancement in support of 64-bit
- Support for greater than 2 GB TDUMPs
- System trace buffers above the bar
- IPCS enhancements
- Standalone Dump enhancements

24.1 Hung SDUMP detection

An SVC dump is written to an SVC dump data set, which is specified on the DCB parameter of the SDUMP or SDUMPX macro, available as SYS1.DUMPxx, or automatically allocated. If a SYS1.DUMPxx data set is not available, the dump is captured and the system asks the operator to provide a SYS1.DUMPxx data set. An operator can also determine the current options under which SVC dump is processing and the status of SVC dump.

An SVC dump is obtained in one of the following ways:

- By issuing an SDUMP or SDUMPX macro in an authorized program
- Entering a DUMP or SLIP command
- Using a SLIP command in the IEASLPxx parmlib member

Concurrent SDUMPs

SDUMP uses an internal locking mechanism to prevent concurrent SDUMPs. Only one SDUMP can be captured at a time. If an SDUMP is attempted while the serialization indicators show that a dump is already in progress, the following message is issued:

```
IEE711I SYSTEM DUMP NOT TAKEN. ANOTHER DUMP WAS IN PROGRESS
```

When there is a flood of SDUMP requests, there are timing windows where an SDUMP serialization indicator may be left on when there is actually no dump in progress. When this occurs any attempt to take an SDUMP will fail with message `IEE711I`. The SDUMP serialization indicators are:

- The high order bit (X'80' in byte 1) of CVTSDBF and
- The SDUMP parameter list, RTCTSDPL, being non-zero

The only resolution when this occurs is to cancel the DUMPSRV address space via the MVS command:

```
C DUMPSRV
```

DUMPSRV will restart automatically and the SDUMP serialization indicators are cleared.

One of the major concerns was that there is no indication that SDUMP is in this state. It could be days until the next SDUMP is attempted and only then would the serialization problem be detected.

z/OS V1R10 support for hung SDUMP

The hung SDUMP detection support in z/OS V1R10 will detect the hang and take the following actions:

- Take a TDUMP from the DUMPSRV address space and DUMPSRV data spaces with dump option `SDATA(NUC,CSA,LPA,RGN,TRT,GRSQ,SQA)` which may be used to diagnose the cause of the SDUMP.
- Issue the following message:

  ```
  IEA044E Dumping Services Function is unavailable
  ```

  Message `IEA044E` indicates that SDUMP is hung and SDUMP requests will fail with IEE711I. This gives a timely indication that there is a problem. DUMPSRV should now be restarted via the following MVS command:

  ```
  C DUMPSRV
  ```

The TDUMP should be sent to IBM support for analysis. If the dump causing the serialization problem has not been written to a dump data set and is still in the DUMPSRV dataspace, it will be captured by the TDUMP ensuring that data is not lost when DUMPSRV is restarted.

The TDUMP is written to an automatically allocated data set. The default SDUMP data set name pattern is used, replacing the '..xx&SEQ.' qualifier with '..T&DS.'. This allows multiple data sets to be used if the size of the TDUMP is greater than 2 GB.

24.2 Choices for IEATDUMP data sets

Transaction dump processing supports both pre-allocated and dynamically allocated dump data sets. The dump is allocated from the generic resource SYSALLDA. There are three ways to manage the dump data sets, as follows:

- For the pre-allocated data sets, IEATDUMP writes to the data set without first capturing the dump into a dataspace. The dump can contain more than 2 gigabytes if the data set capacity permits.
- For a data set name template without &DS. at the end, the dump is captured to a dataspace, the data set is allocated with the size needed to contain the captured data, and then the dump is written to disk. The dump cannot exceed 2 gigabytes. If dynamic allocation fails, message IEA820I is issued, and the dump is deleted.
- For a data set name template with &DS. at the end, IEATDUMP does not first capture the dump to a data space. Instead, IEATDUMP writes the dump directly to disk. If it runs out of space in a data set, IEATDUMP allocates another with a higher value for &DS. Each data set has an extent size of 500M that can be changed using ACS routines. These extents are written to until the disk runs out of space or no more extents can be created. At that time, a new data set in the sequence is created. Multi-data set IEATDUMPs utilize up to 999 data sets. The maximum size depends on the amount of space on the volumes where these data sets get allocated.

24.2.1 Transaction dump (TDUMP)

Similar to SNAP dumps, an application can issue an IEATDUMP macro to dump virtual storage areas of interest when the application is running. Today you can request that the TDUMP be written to a data set that is either pre-allocated or automatically allocated. To request a pre-allocated data set, specify the DDNAME parameter in IEATDUMP, which identifies a data set that contains sufficient space in one or more extents for the entire dump to be written. If you do not provide a large enough data set, you will receive a partial dump.

Current implementation of TDUMP uses a passed-in data set name template, in which case the dump is captured to a 2 GB dataspace and then copied to a generated data set. This behavior remains unchanged.

Support for >2 GB TDUMPs to dynamically allocated data sets

With z/OS V1R10, as systems become capable of supporting larger amounts of real storage, the exploitation of that storage is likely to result in larger dumps. 64-bit virtual has already created environments where there is more than 2 GB of private storage that needs to be included in dumps.

Users of TDUMP generally prefer dynamic dump data set allocation instead of the task of pre-allocating data sets, as follows:

- TDUMP can only capture up to 2 GB of data per dump if dynamic data set allocation is enabled.
- For address spaces exploiting 64-bit virtual storage, the 2 GB limit is frequently too restrictive.
- In a 64-bit Java environment, large heaps can result in truncated TDUMPs and the requirement for problem recreates.

For the new support, there is no change to the case when DDNAME is specified. If DSN or DSNAD is specified and &DS is not specified at the end of the dump data set name, then the processing is the same as today. To request automatic allocation, specify the DSN or DSNAD parameters. This is the case where a 2 GB limitation currently exists.

This new support in z/OS V1R10 makes it possible to dump large 64-bit applications. In particular, it will allow you to have a Java environment with large (>2G) heaps, which can still be dumped. Combined with the prioritization of storage for dumping, larger TDUMPs can be captured with higher probability of capturing data needed to debug the problem.

CHNGDUMP command

Most problems can be debugged without dumping the nucleus. If a problem arises that requires that the nucleus be dumped, use the operator CHNGDUMP command to add the NUC SDATA option to all IEATDUMPs. This applies to other options as well. Starting with z/OS V1R10, the system detects a hung situation and takes a TDUMP from the DUMPSRV address space. The following message is issued:

```
IEA044E Dumping Services Function is Unavailable"
```

TDUMP will include system data and DUMPSRV dataspaces when specifying:

```
SDATA=(NUC,CSA,LPA,RGN,TRT,GRSQ,SQA)
```

Note: The DUMPSRV dataspaces might contain captured SVC dumps. These captured dumps can be extracted from the TDUMP with an IPCS COPYDUMP command. Message IEA822I is issued to the system operator to indicate that a complete or partial TDUMP was written to the TDUMP data sets if the request of TDUMP was successful. Message IEA820I is issued to the system operator to indicate that the TDUMP was requested but not taken if the TDUMP processing was not successful.

This leads to improved communication of dumping services availability so it can be recycled by the installation. Either TDUMP of DUMPSRV can help diagnose the situation that led to the failed dump or improper lock condition. Data that has been captured in DUMPSRV dataspaces will not be lost.

TDUMP data sets

The TDUMP is written to an automatically allocated data set. The default SDUMP data set name pattern is used and replaces the '..xx&SEQ.' qualifier with a '..T&DS.' symbol. This allows multiple data sets to be used if the size of the TDUMP is greater than 2 GB. The IEA827I message is issued to the system operator to indicate where the TDUMP was written to. To recycle the dumping services, issue the CANCEL DUMPSRV command.

&DS symbol

Support is added in z/OS V1R10 to recognize the &DS (for Dump Section) symbol at the end of dump data set name template. When the &DS symbol is present, IEATDUMP does not first

capture the dump to a dataspace. Instead, IEATDUMP writes the dump directly to disk. IEATDUMP will allocate the first dump, ending with a “1”. If this data set runs out of disk space or if it uses up all 16 extents before the dump is completed, IEATDUMP processing will allocate, open, and continue dumping to a data set with the same name, but ending in a “2” and so forth. This continues until the entire dump is written. Multi-data set IEATDUMPs utilize up to 999 data sets. The maximum size depends on the amount of space on the volumes where these data sets get allocated.

New TDUMP messages

Dump sections may be located on different volumes and TDUMPs write to volumes that have a STORAGE or PUBLIC use attribute. Data sets allocated using the &DS symbol have a primary extent size of 500 MB and a secondary extent size of 500 MB. These values can be changed by providing an ACS routine that will be driven by DFSMS. The IEA827I message tells you where the TDUMP was written to, as follows:

```
IEA827I {COMPLETEorPARTIAL} TRANSACTION DUMP WRITTEN TO number DATASETS
STARTING FROM dsname COMBINE USING COPYDUMP PRIOR TO VIEWING IN IPCS
```

The resulting dump data sets need to be combined using COPYDUMP before they can be viewed in IPCS. The IPCS COPYDUMP command already supports an input list of data sets and combines them to create a single large dump data set.

The IEA822I message is issued to the system operator to indicate that a complete or partial TDUMP was written to the TDUMP data sets if the request of TDUMP was successful.

24.3 SDUMP serviceability enhancement in support of 64-bit

There are SDUMP enhancements in z/OS V1R10 in support of 64-bit virtual storage and for 64-bit common storage. SDUMP has also been enhanced to honor the new DUMP= keyword introduced by RSM for the IARV64 macro.

24.3.1 SDUMP enhancement for 64-bit virtual storage

SDUMP dump sizes are increasing and due to the increase in system capacity and utilization of high virtual storage, will continue to increase. IBM applications such as WebSphere or DB2 obtain large ranges of high virtual storage, some of which have contents that are not critical, nor sometimes even necessary, for problem determination efforts. Temporary data structures, data caches are examples of data areas that may not need to be routinely dumped, and/or can safely be sacrificed in lieu of more important areas of storage.

Prior to z/OS V1R10, storage was dumped in ascending order. This means that if the MAXSPACE for dumps is exceeded storage in the high virtual storage ranges would not be dumped. For AMODE 64 Language Environment (LE) applications, such as WebSphere, critical debugging data resides in High virtual storage. Heap storage which can be very large resides lower in virtual storage than the LE stacks. The LE stacks are critical as they are needed for debugging problems but would very likely not be dumped when MAXSPACE is exceeded due to the large heap storage that is dumped first.

In z/OS V1R10, SDUMP will honor the DUMPPRIORITY associated with 64-bit virtual memory objects. DUMPPRIORITY is a new keyword for IARV64 GETSTOR calls introduced by Real Storage Manager (RSM) in z/OS V1R10. AMODE 64 Language Environment now uses memory object dump priorities to prioritize the above-the-bar storage being written to a dump data set, to improve the chance that critical diagnostic information is captured. AMODE

64 LE applications can use the __moservices() function to create and detach memory objects with user-defined dump priorities, and to set a shared memory dump priority.

The DUMPPRIORITY is included in the IAVR64 GETSTOR request. DUMPPRIORITY is an optional keyword for the new DUMP=LIKERGN keyword. The default is DUMP=LIKERGN,DUMPPRIORTY=99.

See RSM xxx and LE xxx for more details.

24.3.2 SDUMP enhancement for 64-bit memory objects

In z/OS V1R10, RSM introduced new options on the IARV64 macro.

For a private virtual storage request via IARV64 REQUEST=GETSTOR, the DUMP= keyword can specify DUMP=LIKERGN or DUMP=LIKELSQA.

- When the SDATA dump option RGN is specified, then 64-bit memory objects obtained with DUMP=LIKERGN will be dumped.
- When the SDATA dump option LSQA is specified, then 64-bit memory objects obtained with DUMP=LIKELSQA will be dumped.

For a common virtual storage request via IARV64 REQUEST=GETCOMMON, the DUMP= keyword can specify DUMP=LIKECSA or DUMP=LIKESQA.

- When the SDATA dump option CSA is specified, then 64-bit memory objects obtained with DUMP=LIKECSA will be dumped.
- When the SDATA dump option SQA is specified, then 64-bit memory objects obtained with DUMP=LIKESQA will be dumped.

The use of the DUMP= keyword in IARV64 REQUEST=GETSTOR and request=GTETCOMMON is explained in more detail in 4.7.1, “Dump priority of LE memory objects” on page 122.

24.3.3 Order of dumping 64-bit storage

There are no changes in the order in which storage is dumped but 64-bit memory objects are included when dumping the storage for the related SDATA option.

Global storage is dumped first in the following order:

- Global storage below 2 GB
- Global storage above 2 GB
  - If SDATA=SQA is specified, storage obtained with DUMP=LIKESQA
  - If SDATA=CSA is specified, storage obtained with DUMP=LIKECSA

Then private storage for each address space specified is dumped in the following order:

- Private storage below 2 GB
- Private storage above 2 GB
  - If SDATA=LSQA is specified, storage obtained with DUMP=LIKELSQA
  - Storage specified in LIST64. This may include shared or common storage that was not already dumped together with the global storage.
  - If SDATA=RGN is specified, the rest of the storage obtained with DUMP=LIKERGN

24.3.4 SDUMP enhancements in support of WebSphere Application Server z/OS 64-bit

SDUMP dump sizes are on the rise, and with the increase in system capacity and utilization of high virtual storage, will continue to increase. IBM applications such as WebSphere or DB2 obtain large ranges of high virtual storage, some of which have contents that are not critical, nor sometimes even necessary, for problem determination efforts. Temporary data structures, data caches are examples of data areas that may not need to be routinely dumped, and/or can safely be sacrificed in lieu of more important areas of storage.

SDUMP sizes continue to grow with the increase in system capacity and utilization of high virtual storage:

- Storage is dumped in ascending order.
- Some data captured is of limited interest in problem determination.
- Run out of room before dumping critical data.
- WebSphere Application Server 64-bit users with large heaps can end up with dumps where the system and LE stacks needed to debug them are truncated.

One avenue for the large dump relief is to be more selective in choosing areas of high virtual storage in an SDUMP. Some enhancement already addresses this relief by providing a mechanism for users of high virtual storage to prioritize their storage areas, and for SDUMP to honor these priorities while dumping.

By allowing the obtainer of the high virtual storage to specify a priority on the IARV64 GETSTOR or SHAREMEMOBJ invocation, each high virtual memory object can be marked so that SDUMP can access the highest priority items first when capturing storage.

Priorities are numeric, with 1 being the highest priority, and 99 being the lowest. The default for any memory object is 99.

This dump prioritization makes it possible to be more selective in choosing areas of high virtual storage to include in SDUMP:

- Include an optional priority on each high virtual memory object by specifying DUMPPRIORITY on IARV64 GETSTOR or SHAREMEMOBJ.
- SDUMP honors the priority specified by the obtainer of the storage to determine which memory objects need to be captured first.
- Storage dumped by SDATA=RGN is in priority order.
- LIST64 ranges are dumped before RGN ranges.
- Duplicate ranges are removed.
- Storage below the bar is dumped exactly as it has always been dumped.

24.4 Support for greater than 2 GB TDUMPs

As systems become capable of supporting larger amounts of real storage, the exploitation of that storage is likely to result in larger dumps. 64-bit virtual has already created environments where there is more than 2 GB of private storage that needs to be included in dumps.

Today you can request that a transaction dump (TDUMP scheduled via the IEATDUMP macro) be written to a data set that is either pre-allocated or dynamically allocated. To request a pre-allocated data set, the DDNAME parameter is specified in the IEATDUMP

macro. This identifies a data set that contains sufficient space in one or more extents for the entire dump to be written. If you do not provide a large enough data set, you will receive a partial dump.

Specifying DSN or DSNAD instead of DDNAME in the IEATDUMP macro results in a dynamically allocated dump data set that is limited to 2 GB. Because most users prefer to use dynamically allocated dump data sets, the 2 GB limit results in partial dumps for address spaces that exploit 64-bit virtual storage. In 64-bit Java applications large heap sizes often result in partial dumps.

The IEATDUMP macro has been enhanced in z/OS V1R10 so that an optional symbol**,** &DS**,** can be specified at the end of the data set name pattern specified via the DSN or DSNAD keywords. When the &DS symbol is present, IEATDUMP allocates the first data set for dumping, ending with 001. If this runs out of disk space or uses up all 16 extents before the dump is completed, dumping will be continued to data sets with the same name, but ending in 002, 003, and so on, until the entire dump is written. Each of these data sets is allocated with a primary extent size of 500 MB and a secondary extent size of 500 MB, but it is possible to change these values via DFSMS ACS routines.

An example of coding the IEATDUMP macro including the &DS keyword in the data set name pattern specified via DSN= is shown in Figure 24-1.

```
An example using DSN:
IEATDUMP DSN=DUMPDSN,HDR=DUMPTTL2
         .
         .
         .
DUMPDSN DC AL1(E2-S2)
S2 DC C'HLQ.TDUMP.D&&YYMMDD..T&&HHMMSS..&&SYSNAME..&&JOBNAME..&DS.'
E2 EQU * DUMPTTL2 DC AL1(E3-S3)
S3 DC C'IEADUMP TO AUTOMATICALLY ALLOCATED DATA SET'
E3 EQU *
```

Figure 24-1 Coding the IEATDUMP macro including the &DS keyword

TDUMP messages

Message IEA827I is issued for TDUMP dynamically allocated data sets with the &DS symbol:

```
IEA827I {COMPLETEorPARTIAL} TRANSACTION DUMP WRITTEN TO number DATASETS
STARTING FROM dsname COMBINE USING COPYDUMP PRIOR TO VIEWING IN IPCS
```

As explained in the IEA827I message, the IPCS COPTYDUMP command can be used to combine the multiple TDUMP data sets into a single data set for use with IPCS, for example, using IPCS COPYDUMP panels.

24.5 System trace buffers above the bar

The system trace is vital for diagnostic purposes but can wrap in a short time interval losing vital diagnostic data. A trace buffer is maintained for each CPU and resides in LSQA storage in the TRACE address space. Because private region storage below the 2 GB bar is limited, we need to allow system trace buffers to reside above the bar to increase the trace buffer size.

In z/OS V1R10, system trace buffers are moved above the 2 GB bar. Trace buffer vector tables (TBVTs) and trace table copy headers (TTCHs) remain below the bar. Only the trace buffers pointed to by them have moved above the bar.

- The size of each trace buffer remains at 4K.
- The default trace buffer size per processor has been increased from 256K to 1M.
- The minimum trace buffer size per processor is 1M.
- Changes to the trace structure size will be allowed in megabyte or gigabyte units only.
- System trace buffers will use Large Pages when the System z10 hardware is available and the LFAREA parameter in IEASYSxx is set.

  While allocating the trace buffers above the bar, Large Pages are used to back them in real storage. The amount of real storage to be set aside for Large Pages is specified through the FAREA parameter in IEASYSxx. Whenever Large Pages are not available, system trace will allocate Memory Objects above the bar, fix them, and use them for trace buffers.

Restriction: The combined trace structure size of all processors cannot exceed half the amount of online real storage.

This enhancement allows system trace to cover a longer period of time before wrapping, which aids in problem diagnosis.

24.5.1 Issuing the TRACE command

The trace buffer size allocated per CPU and the total trace buffer size is set with the TRACE command. The trace buffers are allocated in megabytes with 1 MB per processor as the default. The TRACE command still allows a value of 1-999K to be specified but is rounded to 1 MB. The total trace buffer size is specified via the BUFSIZ keyword which allows values of 1-99999M or 1-999G to be specified.

The TRACE command has the following syntax:

```
TRACE [STATUS  ]
      [ST[,nnnK|nnnM|nG][,BR={ON|OFF}][,MODE={ON|OFF}] ]
      [ST[,BUFSIZ=nnnnnK|nnnnnM|nnnG][,BR={ON|OFF}][,MODE={ON|OFF}] ]
      [ST[,OFF]
```

The amount of trace buffer size allocated per processor is determined by a combination of the trace buffer size per processor and the total trace buffer size specified with the TRACE command, and also the number of processors.

The following rules apply for trace buffer allocation:

- A minimum trace buffer size per processor of 1 MB.
- A minimum overall trace buffer size of the size specified for each processor * the number of CPUs.
- The overall trace buffer size is allocated in chunks = (number of CPs * 1MB) and is rounded up.

For example, if the processor has 12 CPUs the overall trace buffer size is allocated in 12 MB chunks. If an overall trace buffer size of 13 MB is specified, it will be rounded up to 24 MB with each processor allocated a trace buffer size of 2 MB.

Figure 24-2 shows the result of a sequence of TRACE commands for a system with 12 CPUs with the default trace buffer size specified at IPL. The commands are issued, as follows:

- Initially the trace buffer size is 1 MB per processor resulting in an overall trace buffer size of 12 MB.
- Changing the buffer size per processor to 2 MB results in an overall buffer size of 24 MB.
- Changing the buffer size per processor back to 1MB results the overall buffer size being reduced to 12 MB.
- Changing the overall trace buffer size to 13 MB results in the trace buffer size per processor being increased from 1 MB to 2 MB and the overall trace buffer size increased from 12 MB to 24 MB.
- Increasing the trace buffer size per processor to 8 MB was not allowed. This would have increased the overall trace buffer size to 96 MB, which would have exceeded half the real storage available. This system had 128 MB of real storage.

```
TRACE ST
IEE839I ST=(ON,0001M,00012M) AS=ON BR=OFF EX=ON MO=OFF MT=(ON,024K)

TRACE ST,2M
IEE839I ST=(ON,0002M,00024M) AS=ON BR=OFF EX=ON MO=OFF MT=(ON,024K)

TRACE ST,1M
IEE839I ST=(ON,0001M,00012M) AS=ON BR=OFF EX=ON MO=OFF MT=(ON,024K)

TRACE ST,BUFSIZ=13M
IEE839I ST=(ON,0002M,00024M) AS=ON  BR=OFF EX=ON  MO=OFF MT=(ON,024K)

TRACE ST,8M
IEE931I TRACE    INSUFFICIENT STORAGE FOR COMMAND
IEE839I ST=(ON,0002M,00024M) AS=ON  BR=OFF EX=ON  MO=OFF MT=(ON,024K)
```

Figure 24-2 TRACE command

24.6 IPCS enhancements

System outage problems that are high severity often involve analysis of a standalone dump (SAD). For large systems with many CPUs and high storage usage SADs can be very large, often exceeding 10 GBs. These large SADs need to be sent via FTP to IBM support or other vendor support sites and can take many hours to transmit and then many hours to initialize under IPCS for analysis.

24.6.1 IPCS COPYDUMP

It is normal practice to use IPCS COPYDUMP to extract a subset of address spaces from the SAD to allow faster transmission so that analysis can begin sooner. In z/OS V1R10, IPCS COPYDUMP has been enhanced to perform dump initialization while it is transcribing the dump. This eliminates the need to initialize the dump, which can take several hours, after it has been received at the support site for analysis.

In addition, two new keywords have been added to IPCS COPYDUMP to aid in extracting a subset of critical address spaces for faster analysis:

- `JOBLIST(j1[,j2][,j3]…[,jn])`
- `EASYCOPY`

JOBLIST

The JOBLIST keyword is an alternative to the existing ASID keyword. It is useful when storage for a critical application needs to be included in the COPYDUMP but the ASID is not easily determined, and also for critical system address spaces. Although system address spaces are always assigned the same ASID at IPL, the ASID can change after the address space is restarted.

Note: COPYDUMP always copies the dump records for address spaces 1 through 4 to the output data set regardless of the ASIDLIST or JOBLIST specified.

EASYCOPY

EASYCOPY subsetting distinguishes two types of dumps:

- The first is a dump that does not contain special records that associate ASIDs in the dumped system with job names. EASYCOPY causes ASIDs 1-20 to be included in these dumps.
- The second is a dump that does contain the special records. EASYCOPY copies ASID(1) plus a list of ASIDs associated with preselected job names. Currently these names are ALLOCAS, ANTAS000, ANTMAIN, CATALOG, CONSOLE, DEVMAN, DUMPSRV, IEFSCHAS, IOSAS, IXGLOGR, JESXCF, JES2, JES3, OMVS, PCAUTH, RASP, RS, SMSPDSE, SMSPDSE1, SMSVSAM, TRACE, WLM, and XCFAS. This allows selection of ASIDs that are often started after z/OS has put all of the ASIDs between 1 and 20 into use.

24.6.2 IPCS COPYDDIR

The following functions were added to IPCS COPYDDIR:

- EXPORT dump directory records pertaining to one source
- IMPORT dump directory records generated by the EXPORT function into the dump directory for the current session

This allows the dump directory records created by COPYDUMP to be transmitted with the dump and used by IBM support to eliminate the time it takes to initialize the dump under IPCS before analysis can begin.

Prior to this support COPYDDIR implicitly assumed that the source data set was a RECFM=VB file that was to be IMPORTed. The EXPORT keyword allows an IPCS dump directory to be EXPORTed to a RECFM=VB file.

Restriction: The EXPORT keyword is not currently supported via the IPCS panels, option 3.1. The EXPORT keyword for COPYDDIR can only be used via a batch job or an IPCS subcommand via option 6.

24.6.3 IPCS support for Extended Address Volumes

In z/OS V1R10, IPCS supports Extended Address Volumes (EAVs) for dump directories. The new keyword EADSCB=OK is specified when issuing the OBTAIN for dump directories.

24.6.4 New IPCS messages

Informational messages have been added to alert potential performance problems in IPCS processing.

Reporting standalone dump time

BLS1810I has been added to report the time it takes to write out an SAD. BLS1810I is issued during SAD intialization and also during COPYDUMP processing.

```
BLS18310I Stand alone dump required mm:ss to record to dsn
```

Explanation: COPYDUMP has transcribed SADMP data set dsn and has located the statistics information that the standalone dump placed into the dump data set. Recording, exclusive of time involved prompting the system operator, required mm:ss minutes and seconds.

Reporting dump directory CISIZE

BLS21110I has been added to report when the CISIZE of the IPCS dump directory is less than 24K.

```
BLS21110I CISIZE(cisize) is less than 24K. It may degrade IPCS performance.
```

Explanation: The directory associated with ddname IPCSDDIR was allocated with control interval size of cisize. Control interval sizes less than 24K have been shown to be more vulnerable to fragmentation when used as IPCS dump directories, and IPCS performance can be degraded when such fragmentation occurs.

Reporting dump directory usage

BLS21111I has been added to report the percentage of the IPCS dump directory that is in use.

```
BLS21111I  percentage of dump directory logical capacity is in use.
```

Explanation: The directory associated with ddname IPCSDDIR was allocated without the extended addressability attribute, limiting its logical capacity to 4 gigabytes of data. IPCS detected that the high used RBA for the directory currently exceeds 2 gigabytes.

24.6.5 Using the IPCS enhancements

The following sequence of screen captures illustrate the following:

- The messages issued initializing a standalone dump
- The messages issued during IPCS COPYDUMP
- The messages issued during IPCS COPYDDIR

```
 IKJ56650I TIME-08:36:19 AM. CPU-00:00:00 SERVICE-30017 SESSION-00:02:15 MAY 15,
2008
 BLS18122I Initialization in progress for DSNAME('SYS1.SADMP')
 BLS18124I TITLE=TESTING R10 SADMP PROGRAM ON 5/12/08
 BLS18223I Dump written by z/OS 01.10.00 stand alone dump - level same as IPCS l
evel
 BLS18222I z/Architecture mode system
 BLS18125I CPU(1) STATUS available
 BLS18125I CPU(2) STATUS available
 BLS18125I CPU(3) STATUS available
 BLS18125I CPU(4) STATUS available
 BLS18310I Stand alone dump required 00:19 to record to SYS1.SADMP
 BLS18123I 36,570 blocks, 152,131,200 bytes, in DSNAME('SYS1.SADMP')
 IKJ56650I TIME-08:36:30 AM. CPU-00:00:00 SERVICE-65745 SESSION-00:02:25 MAY 15,
2008
 BLS18224I Dump of z/OS 01.10.00 - level same as both SADUMP and IPCS levels
 ***
```

Figure 24-3 Initializing a SAD

```
 --------------------------------- IPCS UTILITY MENU  -------------------------
                                                      ********************
   1  COPYDDIR    - Copy dump directory data           * USERID  - IBMUSER
   2  COPYDUMP    - Copy a dump data set               * DATE    - 08/05/15
   3  COPYTRC     - Copy trace data                    * JULIAN  - 08.136
   4  DSLIST      - Process list of data set names     * TIME    - 08:42
   5  DAE         - Process DAE data                   * PREFIX  - IBMUSER
   6  SADMP       - SADMP dump data set utility        * TERMINAL- 3278
                                                      * PF KEYS - 12
 Enter END command to terminate                       ********************
```

Figure 24-4 IPCS option 3

```
 ---------------------------- Copy an MVS Dump Data Set -----------------------
 Command  ===>

 Enter Input parameters to copy an MVS dump data set.
 Use ENTER to specify output parameters, END to terminate.

 INPUT DATA SET LIST
  NAME 'SYS1.SADMP'                                    (DSNAME or DDNAME)
  NAME                                                 (DSNAME or DDNAME)
  NAME                                                 (DSNAME or DDNAME)

  NAME TYPE      ENTER "/" TO SELECT    ASID RANGE      / ONLY SYSTEM ADDRESS
  1 1. DSNAME      LEAVE                START X'      '   SPACES ("/" TO SELECT)
    2. DDNAME      CLEAR                END   X'      '

  SKIP        (No. of dumps to skip, nnn or EOF)
  JOBLIST IBMUSER                                        (Job names)
```

Figure 24-5 Using COPYDUMP

```
---------------------------- Copy an MVS Dump Data Set ----------------------
Command  ===>

Enter Output  parameters to copy an MVS dump data set.
Use ENTER to copy data, END to terminate.

OUTPUT DATA SET:        NULLFILE         INITIALIZE / (Enter "/" to select)
 NAME 'SYS1.SADMP.COPYDUMP'                          (DSNAME or DDNAME)

FULL DATA SET:          NULLFILE         INITIALIZE   (Enter "/" to select)
 NAME                                                (DSNAME or DDNAME)

COMPLEMENT DATA SET:    NULLFILE
 NAME                                                (DSNAME or DDNAME)



 NAME TYPE       SPACE (No. of Records)    EXECUTION        DEFAULT
 1 1. DSNAME      PRIMARY   1500           2 1. BATCH
   2. DDNAME      SECONDARY 1500             2. FOREGROUND
```

Figure 24-6 Using COPYDUMP

```
 BLS18168D Proceed with copy?  Enter Y to continue, N to terminate
y
 BLS18124I TITLE=TESTING R10 SADMP PROGRAM ON 5/12/08
 BLS18223I Dump written by z/OS 01.10.00 stand alone dump - level same as IPCS
l
evel
 BLS18222I z/Architecture mode system
 BLS18125I CPU(1) STATUS available
 BLS18125I CPU(2) STATUS available
 BLS18125I CPU(3) STATUS available
 BLS18125I CPU(4) STATUS available
 BLS18310I Stand alone dump required 00:19 to record to SYS1.SADMP
 ***
```

Figure 24-7 IPCS COPYDDIR

```
 IPCS OUTPUT STREAM ----------------------------------------- Line 0 Cols 1 78
  ****************************** TOP OF DATA ********************************
  BLS18173I Dump 1 - Title=TESTING R10 SADMP PROGRAM ON 5/12/08
  BLS18169I Dump 1 is being copied
  BLS18170I 25,449 records 105,867,840 bytes, copied
  BLS18184I 11,121 extraneous records not copied
  BLS18171I End of input data set reached
  ****************************** END OF DATA ********************************
```

Figure 24-8 IPCS COPYDDIR

24.7 Standalone dump enhancements

In response to customer requirements, SAD processing has been enhanced to:

- Avoid issuing WTOR ICK21836D during SAD IPL text creation
- Suppress AMD029D during SAD processing

24.7.1 Avoid issuing WTOR ICK21836D during SAD IPL text creation

In z/OS V1R10 SAD processing exploits ICKDSF APAR PK16403 to avoid issuing WTOR ICK21836 during SAD IPL text creation. WTOR ICK21836 is issued when IPL text is being created on a volume which already has IPL text.

A new option, IPLEXIST =YES|NO, on the AMDSAMP macro allows the choice to bypass WTOR ICK21836 when creating SAD IPL text.

- The default is IPLEXIT=NO for compatibilty.
- When IPLEXIST=YES, SADMP includes IPLEXIST with ICKDSF parameters, so that ICKDSF does not prompt the operator with message ICK21836D if there is already IPL text on the volume.

24.7.2 Suppress AMD029D during SAD processing

A new option, AMD029 = YES | NO on the AMDSAMP macro allows message AMD029 to be suppressed when a 3270 console screen becomes full.

```
AMD029D REPLY W TO WAIT AFTER NEXT FULL SCREEN, ELSE REPLY N; REPLY=
```

- The default is AMD029 = YES for compatibility.
- When AMD029 = NO is specified, SADMP does not issue AMD029D when a 3270 console screen becomes full. SADMP behaves as if the operator had replied `N` to AMD029D.

Note: The parameter is meaningless when the system console is used, since AMD029D is never issued for the system console.

25

XCF/XES enhancements

In this chapter we describe a number of XCF/XES enhancements. These include performance improvements for CF processing and exploitation of new function. The following enhancements will be discussed:

- Improved protocols for CF duplexing
- Performance improvement for CF lock requests
- XES Shared Message Notification enhancements
- XCF exploitation of AUTOIPL
- New and enhanced XCF health checks

25.1 Improved protocols for CF duplexing

Coupling Facility (CF) requests have longer service times when issued to a duplexed structure compared to a simplex structure. The duplexed command pairs must execute in a coordinated fashion within the two Coupling Facilities, a process that requires the exchange of signals. The system incurs additional instruction path length for duplexed requests as a result of the need to split requests into a duplexed command pair and to merge the output of the two operations into a single result. The impact on coupling efficiency for a particular application will vary according to the rate at which CF requests are generated, as well as the relative proportion of requests that modify structure objects (writes) to requests that do not (reads).

The service times for duplexed structures get longer as the distance between the structures increases. This has been measured as approximately 10 microseconds per kilometer and so has a significant performance impact for Geographically Dispersed Parallel Sysplex™ (GDPS) implementations.

New function has been introduced in z/OS V1R10 to significantly improve service times for duplexed structures.

25.1.1 CF duplexing request flow

When a structure is being duplexed by system-managed CF structure duplexing, exploiters continue to issue CF requests via the existing interfaces, which are unchanged for duplexing support (IXLCACHE, IXLLIST, and IXLLOCK). XES determines whether the target structure is duplexed. If the target structure is duplexed, XES determines whether the command needs to be duplexed. A pure read command, for example, would not need to be duplexed. A duplexed request is split into two commands and issued separately to each CF. The two commands are not necessarily identical. For example, with an IXLLIST READ and DELETE command, only one of the coupling facilities need do the I/O to return the data that is to be read.

The Coupling Facilities containing the duplexed instances of the structure exchange ready to execute (RTE) signals to coordinate starting of the duplexed command pair. Duplexing signals are implemented as list notification (LN) commands against special “signalling vectors” in the CF. No data is transferred with these signals and the CFCC is not involved in receiving/storing these signals. Thus they have very little overhead and tend to be quite fast. Upon completion of the command pair, the Coupling Facilities exchange ready to complete (RTC) signals to commit the results.

The result of each command is returned independently to XES. When the results of both commands are available, XES inspects the two results, reconciles them, and presents a single response to the exploiting application.

Figure 25-1 on page 521shows a cross-site implementation with a z/OS image and CFA at Site 1 and CFB at Site 2. The time T is shown to represent relative time of the completion of the different phases of the request in the duplexing protocol. Communication between the z/OS image and CFA takes T=0. It is negligible because it is within a site. Communication between CFB and either the z/OS image or CFA takes T=1 because of the distance.

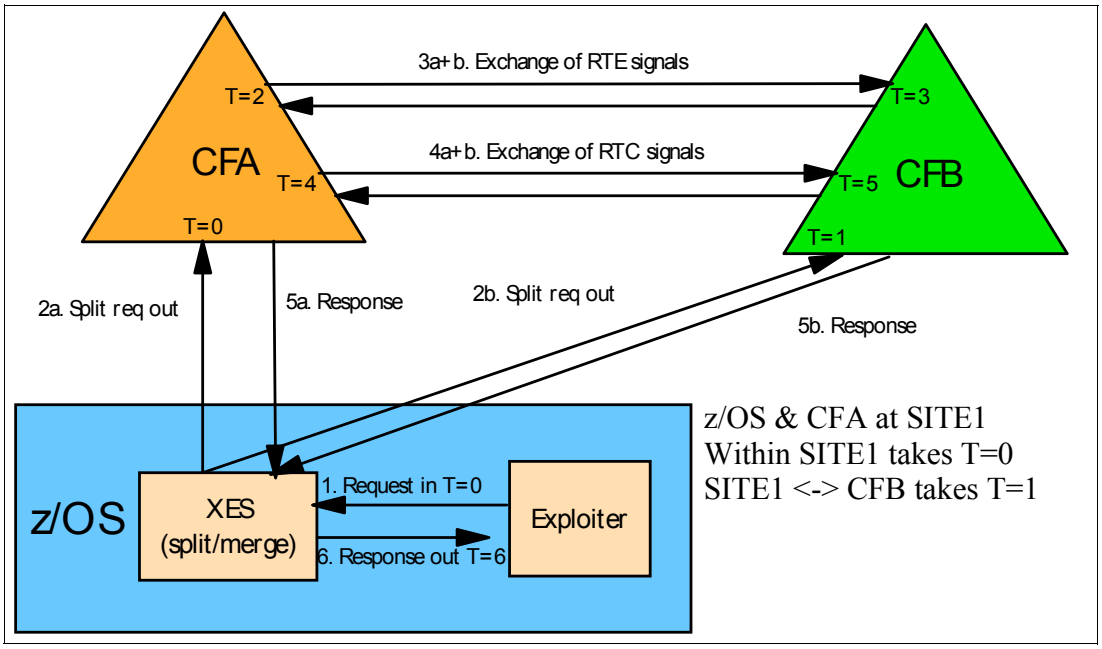


Figure 25-1 CF request flow

25.1.2 Execute without RTE (EWOR)

Execute without RTE (EWOR) refers to the CFLEVEL 15 RTE suppression support. Suppressing RTE signals eliminates approximately one third of the duplexing protocol, which in the past has required three exchanges: command/response to/from the Coupling Facilities, RTE exchange between Coupling Facilities, and RTC exchange between Coupling Facilities. However, the improvement for any particular customer will depend on the particular workload and access patterns for the CF structure.

The danger in suppressing RTE signals is the potential to increase chances that the structure will fall out of duplexing. If one CF executes the command but the other does not, the structures are out of sync and duplexing cannot be preserved. The RTE signal exchange allows both Coupling Facilities to reach a sync point wherein they each know that the other has received the command.

With the suppression of RTE signals, as shown in Figure 25-2 on page 522, XES assumes that both command instances will be received and executed by their respective CFs. However. if one CF processes the command but the peer CF can does not process its command in time, the structure will fall out of duplexing. Thus suppression of RTE signals should not be performed if there appears to be the potential for commands to be delayed in making it to a CF. Delays could be caused by CF issues, which could surface as an InterFace Control Check (IFCC). Delays could be caused by path busy conditions that can occur with shared links.

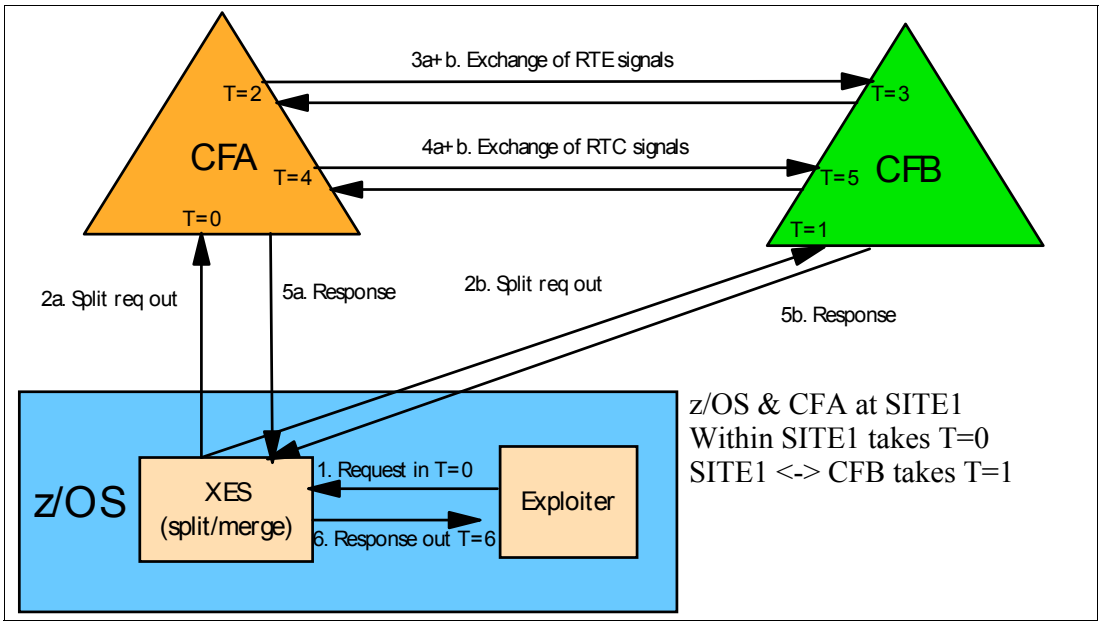


Figure 25-2 RTE suppression

25.1.3 Immediate RTC completion

Immediate RTC completion (IRTC) enablement allows a duplexed request to complete without waiting for the RTC exchange between Coupling Facilities. Rather than eliminating the RTC exchange, the RTC exchange occurs asynchronous to the completion of the request, improving the response time for the request. Unlike EWOR which can be enabled only for some single-entry commands when things are running “well”, IRTC is enabled for a slightly larger set of single-entry duplexed requests all the time. IRTC does not suffer from the increased potential of breaking duplexing like EWOR does. Like EWOR, the improvement for any particular customer will depend on the workload and its access patterns for the structure. Like EWOR, this new protocol will not provide benefit for simplex requests to a duplexed structure.

IRTC provides improved performance for duplexed requests. A 30% improvement in average response is expected when IRTC is enabled and EWOR is not used. The amount of improvement depends on the duplex/simplex ratio and the eligible/ineligible command ratio. The most dramatic improvements are seen for CFs that are a large distance apart.

Figure 25-3 on page 523 illustrates the CF signals when IRTC is enabled.

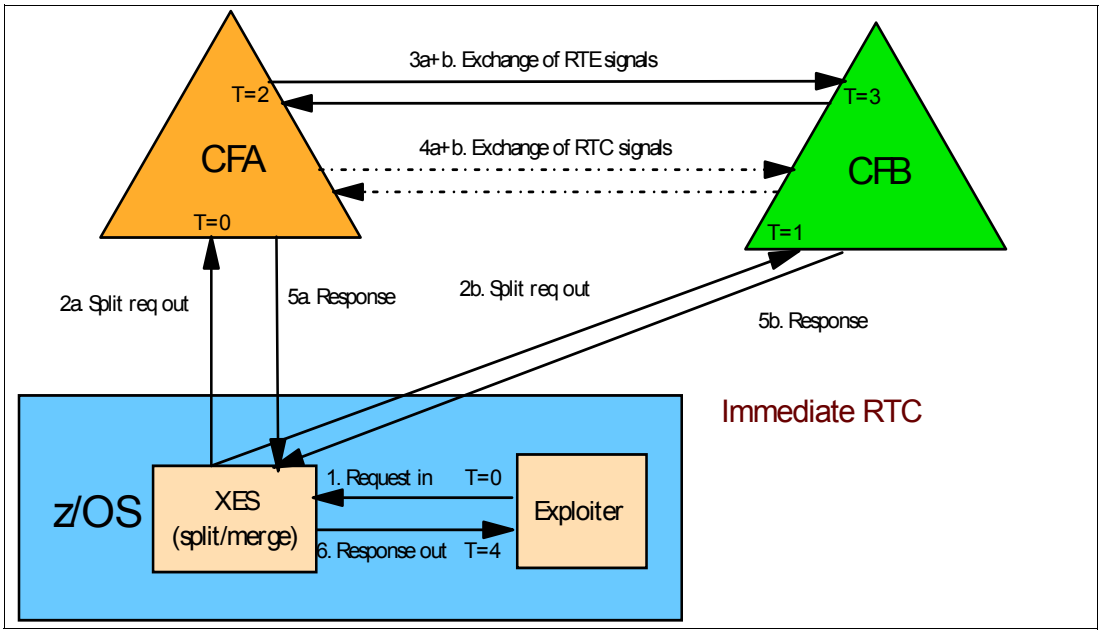


Figure 25-3 Immediate RTC completion

25.1.4 CF duplexing with both EWOR and IRTC

The uses of EWOR and IRTC are independent. Figure 25-4 illustrates the CF signals when both IRTC and EWOR are enabled. Using both may allow the request at CFA to complete at T= 2 minus the same amount of time it would take to complete a simplex request at CFB.

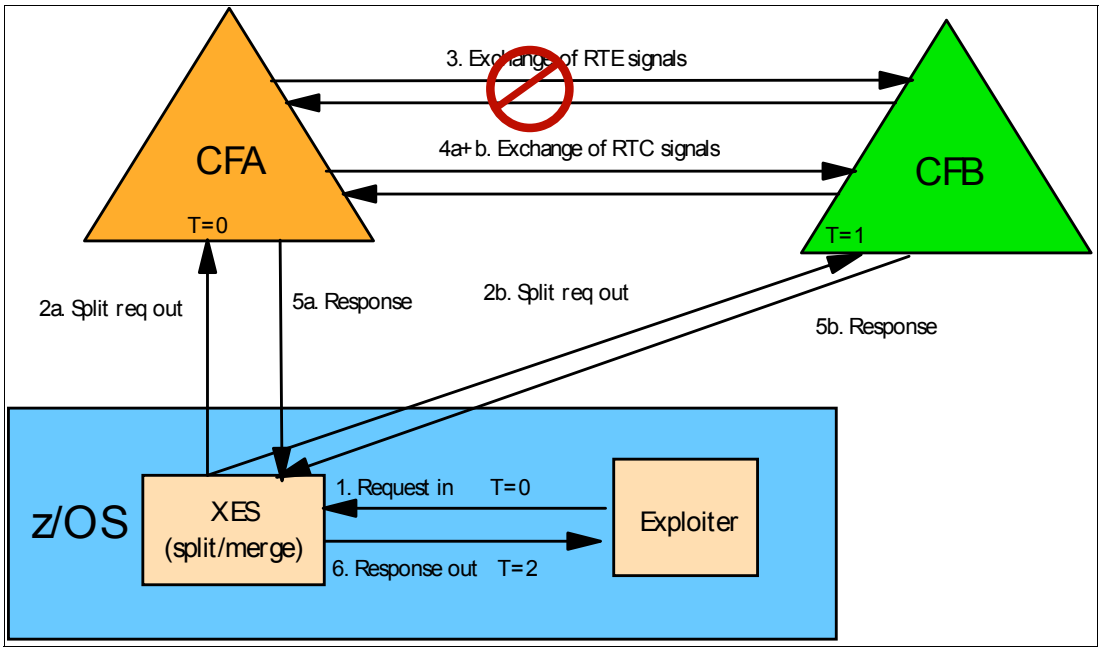


Figure 25-4 CF duplexing with both EWOR and IRTC

25.1.5 Benefits of EWOR and IRTC enablement

Immediate RTC enablement allows a duplexed request to complete without waiting for the RTC exchange between Coupling Facilities. Instead of eliminating the RTC exchange, the RTC exchange will occur asynchronous to the completion of the request, improving the response time for the request. However, the improvement for any particular customer will depend on the workload and access patterns for the CF structure. This new protocol will not provide benefit for simplex requests to a duplexed structure. Nor will it benefit commands that do not support the protocol. The greater the portion of supported commands that run as duplexed requests, the more likely it is that the performance improvement will be noticeable.

- Approximately 30% improvement for duplexed requests that cannot be enabled for EWOR (e.g. cache requests).
- Approximately 50% improvement for duplexed requests that were enabled for EWOR.
- Approximately 60% improvement for duplexed requests that were not using EWOR, but now are (for example, upgrade from CFLEVEL 14 to CFLEVEL 16).
- Improvement is greater as the distance between CFs increases.

Users of system-managed duplexed CF structures will benefit without making any changes.

- The new protocol can only be used when the structure is system-managed duplexed between two CFLEVEL 16 (or higher) Coupling Facilities.
- The new protocol will only be used when the z/OS image performing the request has the support.

25.1.6 Implementation requirements

The new function can be explicitly enabled or disabled, allowing the implementation to be planned in a controlled manner and to fall back to the original system behavior in the event of a problem.

Software requirements

z/OS V1R10 or APAR OA21913 for z/OS V1R7, V1R8 and V1R9. This provides support for

- EWOR or RTE suppression delivered with CFLEVEL 15
- Immediate RTE completion delivered with CFLEVEL 16

No toleration or preconditioning APARs are required.

The new function can be enabled or disabled by both parmlib and command interfaces.

Using the COUPLExx parmlib member

The FUNCTIONS keyword can be used to enable or disable the new function. To enable/disable the new function delivered with CFLEVEL 16 you would specify:

```
FUNCTIONS ENABLE(DUPLEXCF16)
FUNCTIONS DISABLE(DUPLEXCF16)
```

Using the SETXCF command

The new function can also be enabled and disabled via the SETXCF command. To enable/disable the new function delivered with CFLEVEL 16 you would specify:

```
SETXCF FUNCTIONS ENABLE(DUPLEXCF16)
SETXCF FUNCTIONS DISABLE(DUPLEXCF16)
```

25.2 Performance improvement for CF lock requests

XES in z/OS V1R10 provides new function to improve performance for CF lock requests. We review the current CF lock request processing and then discuss the new function.

25.2.1 CF lock request overview

Coupling Facility (CF) lock requests are given preferential treatment to ensure that they are always started as quickly as possible. If there are no subchannels available to start the CF lock request they will spin synchronously waiting for the next available subchannel. This can lead to significant CPU overhead when processors must wait for the next available subchannel. These CPU spins are becoming costlier as processors are becoming faster and faster. In configurations with many CPs, there may actually be more processors than there are subchannels. This can lead to significant CPU overhead when processors must wait for an active CF operation to complete so that its subchannel can be reused to drive the next request.

25.2.2 New function for CF Lock requests

CF Lock operations are changed to support a queuing protocol similar to the current processing for CF List and CF Cache requests. CF Locking requests will still be treated as high priority so that they continue to receive referential treatment. This results in a reduction of CPU time, especially in locking intensive environments.

Not all locking requests will be updated, some requests will continue to be processed as immediate synchronous requests.

This new protocol changes the meaning of the queue times reported as part of the accounting and measurement data returned for a structure entry for a lock structure on the ?IXLMG interface for either a CFNAME or STRNAME request.

The following queue times in the IXLYAMDSTRL structure entry for lock structures are affected:

- IXLYAMDSTRL_QueueTimeCount
- IXLYAMDSTRL_QueueSumTime
- IXLYAMDSTRL_QueueSumTimeSQR

Previously, in z/OS V1R9 and earlier, queue time values for lock structures = the amount of time spinning synchronously for a subchannel.

With the new function in z/OS V1R10:

Queue time values for lock structures = a combination of:

- The time spent on a queue waiting for a subchannel
- The time spent spinning synchronously for a subchannel

Note: Applications referencing these fields may need to be aware of the content change.

This new function should provide a reduction in z/OS CPU consumption since synchronous spinning for subchannel will be avoided. More significant CPU reduction will be seen in configurations where CF Lock structures are both simplex and duplexed. Particularly system-managed duplexed and subchannels have become constrained.

25.3 XES shared message notification enhancements

New function for z/OS V1R10 provides support for enhancements to the CF's notification mechanisms for sublist monitoring. This is intended to improve the operational efficiency of exploiters who make use of this existing sublist notification capability.

In z/OS V1R9 and earlier, when multiple connectors monitor the same sublist (where a sublist might represent, say, a particular transaction class that an exploiter can process), the notifications that result from a sublist transition from the empty to the not-empty state can generate a lot of “false scheduling” because multiple connectors are simultaneously notified that something has been placed onto the monitored sublist, and they race to be the one to read in and process whatever work elements were placed onto the monitored sublist.

CF Level 16 and this new function in z/OS V1R10 implement a mechanism to give a “head start” to some connector in the sysplex for each notification that occurs. The expectation is that the head start connector will win the race to read and process the sublist, before any others have even been notified that the sublist transition has occurred. By eliminating the “race” to process a sublist transition and the overhead typically incurred by “losers” the overall performance of a Parallel Sysplex workload employing a shared message queue mechanism is improved.

25.3.1 Subsidiary list notification delay (SLND)

This new function is called subsidiary list notification delay (SLND). The shared message queue notification delay mechanism is completely transparent to exploiters and provides benefit for all users of sublist notification, regardless of any explicit exploitation. Two known exploiters expected to benefit are:

- IMS shared message queues for IMS TM transaction routing
- IMS shared queues

A list structure supports sublist monitoring when it is defined to have keyed list entries and (storage for) event monitor controls (EMC). Any connector to such a structure can provide a “list notification exit”, which is driven whenever a sublist transitions from empty to not-empty. It is a Coupling Facility responsibility to recognize such transitions and notify all connectors attached to the monitored structure. Before, all notifications occurred immediately, starting a race to handle the transition.

With CFLEVEL16, the CF will notify one connector immediately, then wait the SLND time before notifying the rest. The connector given this head start is selected in a round-robin fashion. z/OS can set the SLND, via the Allocate List Structure (ALST) command, enabling the notification delay. The SLND may be set to any value between 0 and the SLNDL, provided by the Coupling Facility.

Currently, the SLNDL is just over 1 second, and the SLND is always set to 1 millisecond.

25.3.2 Installation of subsidiary list notification delay

For installations already exploiting shared message queues (sublist monitoring), the new function needs to be in z/OS and CFLEVEL 16 Coupling Facilities are required.

To exploit SLND:

- The monitored structure must reside in (be allocated in) a CFLEVEL 16 Coupling Facility.
- At least one z/OS system containing this support must connect to the structure.

  - z/OS V1R10
  - z/OS V1R6 – z/OS V1R9 with new function APAR OAxxxxxx
- Shared message queue exploiters require no enhancements for this support.
- No toleration, compatibility or preconditioning APARs.
- z/OS support and CFCC support can be installed in any order.

z/OS sets the SLND whenever allocating *or* connecting to the monitored list structure in the CFLEVEL16 Coupling Facility. Currently SLND is always set to 1 millisecond.

Note: All systems benefit if at least 1 “uplevel” z/OS connects.

25.4 XCF exploitation of AutoIPL

Some customers have expressed concern that since disabled wait states and nonresponsive systems do not happen often, that about 5-10 minutes are spent by the operations staff just refreshing themselves on the procedure to initiate the standalone dump, locate usable DASD volumes, and many other tasks. Starting the dump and initiating the re-IPL remains a human-intensive process.

z/OS customers have requested that dumping for disabled wait state problems be driven without operator intervention due to the infrequency of taking SADMPs, and the “5 to 10 minute” delay described above.

AutoIPL provides an automated policy-driven means to allow the system to initiate either a standalone dump (SAD) or re-IPL of z/OS or both when a z/OS system is about to enter a disabled wait state.

The benefits include:

- Improved serviceability and first-failure data capture via quick, automatic SAD initiation
- Improved z/OS system availability via quick, automatic re-IPL of z/OS images
- Minimizing z/OS downtime in the event of a disabled wait state

25.4.1 AutoIPL overview

AutoIPL quickly and automatically initiates either a standalone dump (SAD) or re-IPL of z/OS, or both when a z/OS system enters a disabled wait state.

- AutoIPL policy is specified via DIAGxx parmlib member.
- A hard-coded wait state action table designates disabled wait state codes and reason codes for which AutoIPL processing should or should not be performed, and which actions to be taken.
- Most non-restartable disabled wait states/reasons will be eligible for AutoIPL processing
- VARY XCF commands support new options to IPL standalone dumps, re-IPL z/OS or both after the target image has been partitioned out of the sysplex.

The default action for non-restartable wait states is to follow the AutoIPL policy, unless the WSAT says otherwise. The default for restartable waits is to ignore the AutoIPL policy.

25.4.2 Hardware and software requirements

Hardware support for DIAGNOSE 308 (Program-Directed IPL support) is required. This support is available on z10 and z9 machines.

AutoIPL is supported on z/OS V1R10 and higher only. Roll-back APARs will not be provided for earlier releases.

25.4.3 AutoIPL implementation

A new AutoIPL statement in the DIAGxx parmlib member defines the AutoIPL Policy for a system image. Statements in DIAGxx parmlib member define the actions taken by AutoIP.

DIAGxx parmlib member statement

The statement syntax is:

```
AUTOIPL SADMP(sadmp info) MVS(mvs info)
```

Where:

sadmp info Specifies either (device, loadparm) or NONE.

When z/OS is about to enter a wait state, SADMP will be loaded from this volume with this load parameter. If NONE is specified, an SADMP will not be taken.

mvs info Specifies either (device, loadparm) or (LAST) or (NONE).

When z/OS is about to enter a wait state, it is IPLed from this device using this load parameter, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

If MVS with LAST was specified, MVS will be IPLed from the same device and load parameter used for the current IPL, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

If MVS with NONE was specified, MVS will not be IPLed, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

AutoIPL statement processing will be terminated if an invalid parameter is specified.

- IGV009I is issued for an error encountered while processing a device.
- IGV001I is issued for an environment error (DIAGNOSE 308 not fully supported).

Note: Any valid specification of AUTOIPL will cause any prior AutoIPL information to be replaced.

25.4.4 Controlling AutoIPL

To activate the AutoIPL policy, IPL with a DIAGxx parmlib that specifies AUTOIPL options.

To deactivate the AutoIPL policy, specify SADMP(NONE) MVS(NONE) in DIAGxx and issue:

```
SET DIAG=xx
```

To display the AutoIPL Policy in effect, issue the z/OS command:

```
D DIAG
```

Using the wait state action table

The wait state action table (WSAT) is a hardcoded table of wait state codes and reason codes with specific AutoIPL actions assigned to them. The possible actions are:

- None. Let the system enter the wait state condition. This is the default if no AutoIPL policy is specified.
- Initiate a standalone dump.
- Initiate a standalone dump, followed by re-IPL of z/OS.
- Initiate re-IPL of z/OS only.

When the SADMP completes, it checks whether to IPL z/OS and obtains the z/OS IPL information to perform the IPL.

A new option byte is created in the SADMP load parameter, and a bit within that byte can be set to indicate to SADMP to IPL MVS at the conclusion of SADMP's processing (provided that an AutoIPL policy is in place that includes MVS device and load parameter information).

VARY XCF command

In z/OS V1R10, the VARY XCF command supports new options to IPL a standalone dump, re-IPL z/OS or both. Two new optional keywords, REIPL and SADMP, are accepted on the VARY XCF command.

VARY XCF is used to request the removal of a system from the sysplex via sysplex partitioning. The new keywords indicate that the system that is varied out of the sysplex AutoIPL processing will be done when VARY XCF loads a wait state on that system. This assumes that the target system is not already in some other wait state, and has not already been system reset, manually re-IPLed, or fenced at the time of the VARY XCF processing.

By default, systems removed from the sysplex via VARY XCF will not be subject to any AutoIPL processing.

The new syntax of VARY XCF is:

```
VARY XCF, sysname,OFFLINE
[,FORCE] [RETAIN=YES|NO] [,SADMP] [,REIPL]
```

- Either SADMP or REIPL, or both, may be requested after the target image is partitioned out of the sysplex.
- SADMP and REIPL options are not allowed with FORCE.
- The REIPL option is not allowed with RETAIN=NO.
- VARY XCF with Auto-IPL options is rejected if the target system image does not support the requested options. Message IXC372I is issued with a new insert:

```
DOES NOT SUPPORT THE REQUESTED AUTOIPL OPTION(S)
```

25.4.5 New disabled wait state 0A2 reason codes

When the VARY XCF command is issued to partition a system from the sysplex, XCF checks whether the target system has an AutoIPL policy in effect. If it does, XCF drives sysplex partitioning processing with one of three new partitioning reasons, to reflect the AutoIPL options that were specified. The new reason codes are:

- 0A2-17C - Operator requested VARY XCF with REIPL option
- 0A2-180 - Operator requested VARY XCF with SADMP option
- 0A2-184 - Operator requested VARY XCF with both SADMP and REIPL options

These replace the existing reason code:

- 0A2-004 - The operator entered the VARY XCF,sysname,OFFLINE command to remove the system from the sysplex.

Note: Other XCF wait state codes related to partitioning systems out of the sysplex for other reasons are also eligible for AutoIPL processing.

25.4.6 AutoIPL processing

AutoIPL processing is performed when LOADWAIT is entered. If an AutoIPL policy is in place, LOADWAIT will compare the wait state code and the reason code against the Wait State Action Table (WSAT) to determine whether to initiate a StandAlone Dump and/or initiate IPL of z/OS.

Although the WSAT is not a programming interface, it will be externalized so that customers can decide whether to create a policy and activate the AutoIPL function. AutoIPL will be inactive by default.

Each wait state and reason code entry has a flag to indicate whether the SADMP part of the policy should be honored, and a flag to indicate whether the MVS part of the policy should be honored. This is needed because a few MVS wait states are inappropriate for a SADMP or a re-IPL.

When the LOADWAIT component of MVS is invoked to load a disabled wait state, it checks the requested wait state and reason code against the table.

- For non-restartable wait states, LOADWAIT will honor the AutoIPL policy unless a table entry is found that says otherwise.
- For restartable wait states, LOADWAIT will ignore the policy unless a table entry is found that says it should be honored.

Wait state action table (WSAT) entries

The WSAT contains four-byte entries for predetermined wait states and reason codes. Looking at the four-byte entries as eight 4-bit “nibbles”:

- Nibble 1 - contains the AutoIPL action in the last two bits:
  - b’00’ - no action
  - b’10’ - StandAlone dump
  - b’01’ - IPL z/OS
  - b’11’ - StandAlone dump followed by IPL of z/OS
- Nibbles 2 to 5 - contain the reason code
- Nibbles 6 to 8 - contain the wait state code

Table 25-1 shows the entries in the WSAT and explains the action taken for each wait state/reason code combination.

Table 25-1 Entries in the WSAT

WSAT entry	Wait state-reason code	AutoIPL action
X'000040A2'	0A2-0004	No action
X'1017C0A2'	0A2-017C	IPL z/OS
X'201800A2'	0A2-0180	SADMP
X'301840A2'	0A2-0184	SADMP and IPL z/OS
X'200010B5'	0B5-0001	SADMP
X'200020B5'	0B5-0002	SADMP
X'A0000001'	001-0000	SADMP
X'A0000007'	007-0000	SADMP
X'A0000008'	008-0000	SADMP
X'A0000009'	009-0000	SADMP
X'A0000010'	010-0000	SADMP
X'A0000037'	037-0000	SADMP
X'A0000039'	039-0000	SADMP
X'A0000056'	056-0000	SADMP
X'A0000079'	079-0000	SADMP

The WSAT is part of the z/OS nucleus, load module IEANUC01 - CSECT BLWWSATC; see Figure 25-5. It can be found via IPCS by following the pointer chain:

```
CVT -> CSD -> LWVT -> WSAT
```

```
IPCS L CVT+294?+CC?+15C? LE(112)
01993FB0. C2D3E6E6 E2C1E3C3 F0F261F2 F561F0F8 |BLWWSATC02/25/08|
01993FC0. 40C8C2C2 F7F7F5F0 E6E2C1E3 01000000 | HBB7750WSAT....|
01993FD0. 00000011 00000075 00000000 30000FFF |................|
01993FE0. 00000FFF 000040A2 1017C0A2 201800A2 |...... s..{s...s|
01993FF0. 301840A2 200010B5 200020B5 A0000001 |.. s............|
01994000. A0000007 A0000008 A0000009 A0000010 |................|
01994010. A0000037 A0000039 A0000056 A0000079 |...............`|
```

Figure 25-5 Viewing the WSAT in BLWWSATC via IPCS

25.4.7 New and updated messages for AutoIPL processing

There are new messages for AutoIPL processing related to SADMP, DIAGxx and XCF.

New SADMP messages

```
AMD113I  IPLDEV: dddd LOADP: pppppppp AUTOIPL REQUESTED BY [MVS | SADMP
LOADPARM]
   Explanation:  Stand-alone Dump is about to initiate a re-IPL of MVS, from
   device dddd with load parameter pppppppp.
```

```
AMD114I  AMDSADMP INITIATED BY MVS, WAIT STATE CODE = wwwwwwww
   Explanation:  MVS initiated this stand-alone dump, in lieu of loading wait
   state wwwwwwww.
```

New messages for DIAGxx processing

```
IGV009I IN PARMLIB MEMBER=memname ON LINE linenum:   AUTOIPL WAS NOT PROCESSED,
DUE TO AN INVALID PARAMETER OR PROCESSING ERROR
   Explanation:  DIAGxx processing detected an invalid parameter or suffered a
   processing error.

IGV010I IN PARMLIB MEMBER=memname ON LINE linenum:   AUTOIPL WAS NOT PROCESSED,
DUE TO AN ENVIRONMENTAL ERROR
   Explanation:  DIAGxx processing determined that AutoIPL actions cannot be
   performed on this machine because required hardware support is not present.
```

Updated XCF messages

IXC101I and IXC105I have been updated with the following new message reasons:

```
OPERATOR VARY REQUEST WITH REIPL OPTION
   Explanation: An operator requested that system sysname be removed from the
   sysplex  and the system be re-IPLed by the AutoIPL function.

OPERATOR VARY REQUEST WITH SADMP OPTION
   Explanation: An operator requested that system sysname be removed from the
   sysplex and that stand alone dump (SADMP) be IPLed for this system by the
   AutoIPL function.

OPERATOR VARY REQUEST WITH SADMP AND REIPL OPTIONS
   Explanation:  An operator requested that system sysname be removed from the
   sysplex and that stand alone dump (SADMP) be IPLed for this system by the
   AutoIPL function, followed by MVS being re-IPLed by SADMP.
```

IXC220W have been updated with the following new message inserts:

```
AN OPERATOR REQUESTED PARTITIONING WITH THE VARY XCF COMMAND WITH THE REIPL
OPTION
   Explanation: The operator entered a VARY XCF command for this system.  At
   the conclusion of partitioning processing for this  system, the AutoIPL
   function, if activated, will re-IPL the system.

AN OPERATOR REQUESTED PARTITIONING WITH THE VARY XCF COMMAND WITH THE SADMP
OPTION
   Explanation: The operator entered a VARY XCF command for this system.  At
   the conclusion of partitioning processing for this system, the AutoIPL
   function, if activated, will IPL SADMP for this system.

AN OPERATOR REQUESTED PARTITIONING WITH THE VARY XCF COMMAND WITH THE SADMP AND
REIPL OPTIONS
   Explanation: The operator entered a VARY XCF command for this system. At the
   conclusion of partitioning processing for this system, the AutoIPL function,
   if activated, will IPL SADMP for this system followed by a re-IPL of the
   system.
```

IXC370I has been updated with changed text and new inserts.

```
Old: THE ONLY ALLOWABLE OPTIONS ARE OFFLINE AND RETAIN
Updated: THE ONLY ALLOWABLE OPTIONS ARE OFFLINE, RETAIN, REIPL, SADMP AND FORCE


New:  REIPL NOT ALLOWED WHEN SPECIFYING RETAIN=NO
   Explanation: The definitions of the devices for signaling paths to the
   removed system must be retained if an automatic re-IPL of the removed system
   is requested.  Do not specify RETAIN=NO if a re-IPL of the removed system is
   desired.

New:  REIPL OR SADMP NOT ALLOWED WHEN SPECIFYING FORCE
   Explanation: An automatic re-IPL or automatic stand-alone dump of a system
   can not be requested on a VARY command that specifies the FORCE keyword.
   Issue the VARY command without the SADMP and REIPL keywords.
```

IXC372I VARY REJECTED, SYSTEM sysname text:

```
IS NOT PART OF THE SYSPLEX OR IS IPLING
   Explanation: The system specified on the VARY XCF command is not defined to
   the sysplex, or is IPLing into the sysplex.

DOES NOT SUPPORT THE REQUESTED AUTOIPL OPTION(S)
   Explanation: The system specified on the VARY XCF command is not configured
   to support the AutoIPL options requested on the VARY XCF command.
```

25.4.8 AutoIPL limitations and restrictions

There are some AutoIPL limitations and restrictions.

Determining the wait state code

During normal processing when a system enters a disabled wait state the last CPU to perform LOADWAIT processing enters a wait state resulting in the following message to be displayed on the hardware management console (HMC):

```
Central processor (CP) x is in a nonrestartable stopped state due to a System
Control Program (SCP) initiated reset of the I/O interface for partition n
```

In addition, a hardware log entry is created that shows the wait state.

When an AutoIPL action is to be performed, the last CPU does not enter the wait state, and a message does not appear on the HMC or hardware log.

- If a SADMP was taken by AutoIPL before the re-IPL of z/OS, the wait state code will be displayed in the AMD114I message issued by SADMP.
- If AutoIPL performs the IPL of z/OS without taking a SADMP first there is no way to tell what the wait state code was.

Tip: Always request a SADMP in the AutoIPL actions so that the wait state can be determined.

PPRC and GDPS restrictions

GDPS automation is intended to be the sole manager of IPLs and reIPLs of z/OS images in a GDPS environment. AutoIPL should not be configured for use on any system being managed as part of a GDPS environment.

A PPRC secondary cannot be used as the IPL device in the AutoIPL policy.

Sysplex Failure Management (SFM) considerations

AutoIPL processing may be delayed for a period of time based on the systems's XCF Failure Detection Interval before initiating its load of SADMP or z/OS, in order to provide time for SFM to perform fencing isolation. During that time, the system will appear to be hung.

25.5 New and enhanced XCF health checks

z/OS V1R10 provides a number of new and enhanced XCF health checks.

New XCF health checks:

- XCF_CF_SYSPLEX_CONNECTIVITY
- XCF_CF_ALLOCATION_PERMITTED
- XCF_SFM_SUM_ACTION
- XCF_SFM_CONNFAIL
- XCF_SFM_SSUMLIMIT
- XCF_CDS_SPOF
- XCF_CF_STR_DUPLEX
- XCF_CF_STR_AVAILABILITY
- XCF_CF_STR_NONVOLATILE

Existing XCF health checks that have been enhanced:

- XCF_CDS_SEPARATION
- XCF_SYSPLEX_CDS_CAPACITY
- XCF_SIG_PATH_SEPARATION
- XCF_SIG_STR_SIZE

In addition, XCF in z/OS V1R10 runs the following health checks as global health checks:

- XCF_CF_SYSPLEX_CONNECTIVITY
- XCF_CF_ALLOCATION_PERMITTED
- XCF_SFM_ACTIVE
- XCF_SFM_CONNFAIL
- XCF_CF_STR_PREFLIST
- XCF_CF_STR_EXCLLIST
- XCF_CF_STR_DUPLEX
- XCF_CF_STR_AVAILABILITY
- XCF_CF_STR_NONVOLATILE

A global check runs on one system but reports on sysplex-wide status. A global check shows up as disabled for all systems in the sysplex, except for the one where it is actually running. Prior to z/OS V1R10, XES/XCF did not have any global health checks.

XCF_CF_SYSPLEX_CONNECTIVITY

Check that the required number of Coupling Facilities are defined in the CFRM policy and connected to all active systems in the sysplex.

The parameters accepted:

- MINCFS - Minimum number of CFs that must be connected to all systems

  The default is 2 to ensure that structures can always be rebuilt into a CF that is connected to all systems (assuming the preference lists permit use of the CFs).

Output messages:

- IXCH0220E - if minimum number of CFs not available to all systems
- IXCH0908I - listing CFs and systems to which not connected
- IXCH0909I - listing all active systems

Figure 25-6 shows the syntax for XCF_CF_SYSPLEX_CONNECTIVITY:

```
UPDATE CHECK(IBMXCF,XCF_CF_SYSPLEX_CONNECTIVITY)
       SEVERITY(MED) INTERVAL(001:00) DATE(20070707)
       PARM('MINCFS(2)')
       REASON('Multiple coupling facilities should be'
              'connected to all systems.')
       VERBOSE(YES)
```

Figure 25-6 Syntax for XCF_CF_SYSPLEX_CONNECTIVITY

XCF_CF_ALLOCATION_PERMITTED

Check that CFs are not mistakenly left in a condition in which structure allocation is not permitted. It serves as a reminder to remove CFs from maintenance mode to avoid situations where structures cannot be allocated.

There are no parameters.

Output messages:

- IXCH0216I - if all CFs permit allocation
- IXCH0215E - if allocation not permitted in any CF
- IXCH0213I - no CFs defined in policy
- IXCH0214I - identifying CFs in which allocation not currently permitted

Figure 25-7 shows the syntax for XCF_CF_ALLOCATION_PERMITTED

```
UPDATE CHECK(IBMXCF,XCF_CF_ALLOCATION_PERMITTED)
       SEVERITY(MED) INTERVAL(004:00) DATE(20070707)
       REASON('Coupling facilities should have allocation permitted.')
```

Figure 25-7 Syntax for XCF_CF_ALLOCATION_PERMITTED

XCF_SFM_SUM_ACTION

Check that the sysplex failure management (SFM) policy specifies the recommended indeterminate status actions for the local system. This is to encourage the use of automatic isolation rather than prompt.

The parameters accepted:

- SUMACTIONSFM - indeterminate status action.
  - Must be one of ISOLATE, RESET, DEACTIVATE, or PROMPT.
  - The default is ISOLATE.
- SUMINTERVALSFM - indeterminate status interval
  - Decimal value between 0 and 86400 seconds inclusive.
  - The default is 0.
  - Ignored with SUMACTION(PROMPT).

> **Note:** The defaults to SUMACTION(ISOLATE) SUMINTERVAL(0) are equivalent to SFM policy specification of ISOLATETIME(0).

The check is disabled if SFM is not active and automatically enabled if SFM becomes active in the sysplex. The check is considered successful if the policy interval is less than or equal to the parameter interval. In other words, the SUMINTERVALSFM parameter specifies the maximum acceptable policy interval.

Output messages:

- IXCH0515I - if policy matches specification
- IXCH0516E - if policy doesn't match specification

Figure 25-8 shows the syntax for XCF_SFM_SUM_ACTION

```
UPDATE CHECK(IBMXCF,XCF_SFM_SUM_ACTION)
       SEVERITY(MED) INTERVAL(004:00) DATE(20070707)
       PARM('SUMACTION(ISOLATE),SUMINTERVAL(0)')
       REASON('Allow SFM to fence and partition a system'
              'without operator intervention and without'
              'undue delay.')
```

Figure 25-8 Syntax of XCF_SFM_SUM_ACTION

XCF_SFM_CONNFAIL

Check that the sysplex failure management (SFM) policy specifies the recommended action to be taken when one or more systems lose signaling connectivity.

The parameters accepted:

- CONNFAIL(YES | NO) - SFM policy CONNFAIL setting
  - Defaults to CONNFAIL(YES).
  - CONNFAIL(YES) is recommended in a non-GDPS environment.
  - CONNFAIL(NO) is recommended in a GDPS environment.

The check is disabled if SFM is not active and automatically enabled if SFM becomes active in the sysplex.

Output messages:

- IXCH0518I - if policy matches specification
- IXCH0519E - if policy doesn't match specification

Figure 25-9 shows the syntax for XCF_SFM_CONNFAIL

```
UPDATE CHECK(IBMXCF,XCF_SFM_CONNFAIL)
       SEVERITY(MED) INTERVAL(004:00) DATE(20070707)
       PARM('CONNFAIL(YES)')
       REASON('Allow SFM to reconfigure the sysplex when one'
              'or more systems lose signaling connectivity.')
```

Figure 25-9 Syntax for XCF_SFM_CONNFAIL

XCF_SFM_SSUMLIMIT

Check that the sysplex failure management (SFM) policy specifies the recommended action to be taken when a system becomes *status update missing* but continues to send signals.

SSUMLIMIT was introduced by OA11591 to limit the time a system can remain in degraded condition in which it is sending XCF signals but not updating sysplex CDS.

The parameters accepted:

- SSUMLIMIT
  - NONE or the time limit for remaining in degraded condition with a value between 0 and 86400 inclusive.
  - Default is 60.

The check is disabled if SFM is not active and automatically enabled if SFM becomes active in the sysplex. The check is considered successful if the policy interval is less than or equal to the parameter interval. In other words, the SSUMLIMIT parameter specifies the maximum acceptable policy interval.

Output messages:

- IXCH0521I - if policy matches specification
- IXCH0522E - if policy doesn't match specification

Figure 25-10 shows the syntax for XCF_SFM_SSUMLIMIT.

```
UPDATE CHECK(IBMXCF,XCF_SFM_SSUMLIMIT)
       SEVERITY(MED) INTERVAL(004:00) DATE(20070707)
       PARM('SSUMLIMIT(60)')
       REASON('Allow SFM to initiate status update missing'
              'processing when a system remains in a degraded'
              'state too long.')
```

Figure 25-10 Syntax for XCF_SFM_SSUMLIMIT

XCF_CDS_SPOF

Detects and reports single points of failure in CDS configurations via the IOSSPOF function. IOSSPOF detects hardware configuration errors and issues IOS-prefixed Health Checker messages to report each problem found. Refer to ??? for more details about IOSSPOF.

There are no parameters required.

Output messages:

- IXCH0243I - if no single points of failure detected
- IXCH0242E - if at least one single point of failure found
- IXCH0907I - to report complete CDS configuration if appropriate
- IXCH0251I or IXCH0252I - if not checked for CDS type

Note: The XCF_CDS_SPOF check produces a CDS configuration report via message IXCH0907I if any single point of failure is found or if the check is run in debug or verbose mode.

Figure 25-11 shows the syntax for XCF_CDS_SPOF

```
UPDATE CHECK(IBMXCF,XCF_CDS_SPOF)
       SEVERITY(HIGH) INTERVAL(001:00) DATE(20070730)
       REASON('Ensure that couple data sets are configured'
              'without single points of failure.')
       VERBOSE(NO)
```

Figure 25-11 Syntax for XCF_CDS_SPOF

XCF_CDS_SEPARATION

Checks that couple data sets that experience heavier I/O activity reside on separate volumes.

This check verifies that the following best-practice recommendations are implemented:

- The sysplex and CFRM primary couple data sets reside on different volumes.
- The LOGR primary CDS resides on a volume separate from the sysplex and CFRM primaries, if the I/O activity rate to the LOGR CDS warrants it.
- Each primary couple data set resides on a different volume than its corresponding alternate couple data set.

The parameters accepted:

- LOGR(NO | YES) - indicates whether the system logger (LOGR) couple data set (CDS) is to be checked for separation from other performance-sensitive CDS types.

Note: GDPS implementations should specify LOGR(YES) as the LOGR CDS volume should be mirrored via PPRC while the other CDS volumes should not.

Output messages:

- IXCH0241I - if adequately separated
- IXCH0240E - if at least one pair of CDSes is inadequately separated
- IXCH0907I - to report complete CDS configuration if appropriate
- IXCH0251I or IXCH0252I - if not checked for CDS type

Note: The check produces a CDS configuration report via message IXCH0907I if any single point of failure is found or if the check is run in debug or verbose mode.

Figure 25-12 shows the syntax for XCF_CDS_SEPARATION.

```
UPDATE CHECK(IBMXCF,XCF_CDS_SEPARATION)
       SEVERITY(HI) INTERVAL(001:00) DATE(20080104)
       PARM('LOGR(NO)')
       REASON('Ensure that CDS separation has been maintained.')
       VERBOSE(NO)
```

Figure 25-12 Syntax for XCF_CDS_SEPARATION

XCF_SYSPLEX_CDS_CAPACITY

Check that the sysplex CDS is formatted to allow for adequate sysplex growth.

The parameters accepted:

- Recommended growth space for systems - Default is 1.
- Recommended growth space for groups - Default is 2.
- Recommended growth space for members - Default is 5.

Output messages:

- IXCH0601 - if all attributes leave sufficient room for growth
- IXCH0602E - if any attribute does not leave room for growth
- IXCH0911I - comparing peak and maximum values of system growth attributes.

  System growth attributes are not checked for monoplex systems.

Figure 25-13 shows the syntax for XCF_SYSPLEX_CDS_CAPACITY

```
UPDATE CHECK(IBMXCF,XCF_SYSPLEX_CDS_CAPACITY)
       SEVERITY(MED) INTERVAL(000:30) DATE(20070425)
       PARM('1,2,5')
       REASON('Check sysplex CDS capacities.')
```

Figure 25-13 Syntax for XCF_SYSPLEX_CDS_CAPACITY

XCF_CF_STR_DUPLEX

Check that each structure in the CFRM active policy that is currently allocated and has DUPLEX specified as either ALLOWED or ENABLED is actually duplexed.

There are no parameters required.

Output messages:

- IXCH0211I - all ALLOWED or ENABLED structures are duplexed
- IXCH0210E - at least one eligible structure not duplexed
- IXCH0920I - reports structures that should be duplexed but are not
- IXCH0209I - no allocated structures

Figure 25-14 shows the syntax for XCF_CF_STR_DUPLEX

```
UPDATE CHECK(IBMXCF,XCF_CF_STR_DUPLEX)
       SEVERITY(MED) INTERVAL(001:00) DATE(20070707)
       REASON('Allocated structures with DUPLEX(ALLOWED) or'
              'DUPLEX(ENABLED) specified in the CFRM active'
              'policy should be duplexed.' )
       VERBOSE(NO)
```

Figure 25-14 Syntax for XCF_CF_STR_DUPLEX

XCF_CF_STR_AVAILABILITY

Check that for each structure in the policy, the preference list defines at least two usable CFs in different CECs.

The availability of at least two CFs ensures that the structure can be rebuilt and remain available in the event of a CF or CEC failure. To be usable for structure allocation the Coupling Facility must have at least one system connected and have allocation permitted.

When the structure has a policy change pending, the preference list from the pending policy is used for making this check.

There are no parameters required.

Output messages:

- IXCH0217I - all structures have at least two usable CFs in the preference list.
- IXCH0212E - if any structure does not have at least two usable CFs in the preference list.
- IXCH0920I - lists structures and their preference lists.
- IXCH0213I - there are no CFs.

Figure 25-15 shows the syntax for XCF_CF_STR_AVAILABILITY

```
UPDATE CHECK(IBMXCF,XCF_CF_STR_AVAILABILITY)
       SEVERITY(MED) INTERVAL(004:00) DATE(20070707)
       REASON('Each structure preference list definition'
              'should have two usable coupling facilities'
              'which are in different CECs.')
       VERBOSE(NO)
```

Figure 25-15 Syntax for XCF_CF_STR_AVAILABILITY

XCF_CF_STR_NONVOLATILE

Checks that connectors are failure-isolated from the CF and whether connector non-volatility requirements are satisfied.

- If one connector specified NonVolReq=YES, the check concludes that *all* active users should be failure-isolated (even if they specified NonVolReq=NO).
- If a connector is failure-isolated from any structure instance that is being considered, the connector is considered failure isolated. This means the check requirements are met if the structure is duplexed and the connector is failure-isolated from either structure instance.

There are no parameters required.

Output messages:

- IXCH0223I - if all connectors are failure-isolated and their volatility requirements are met
- IXCH022E - if any connector is not failure-isolated or its volatility requirement is not met
- IXCH0910I - reports the relationships between structures, CFs and connectors

Figure 25-16 shows the syntax for XCF_CF_STR_NONVOLATILE.

```
UPDATE CHECK(IBMXCF,XCF_CF_STR_NONVOLATILE)
       SEVERITY(MED) INTERVAL(008:00) DATE(20070707)
       REASON('Coupling facility structure non-volatility and'
              'failure isolation from connectors should be'
              'provided when requested.')
       VERBOSE(NO)
```

Figure 25-16 Syntax for XCF_CF_STR_NONVOLATILE

XCF_SIG_STR_SIZE

Checks that signalling structures are sized to support current or maximum sysplex configurations.

The XCF_SIG_STR_SIZE check is enhanced in z/OS V1R10 to produce one “summary message” of the check results, and a report via message IXCH0915I showing the disposition of all signalling structures defined to the system (check runs on every system every 2 hours). Further, the check parameter changes to dictate whether signalling structures should be judged for their ability to support the current sysplex (SYSTEMS(ACTIVE)) or the maximum sysplex (SYSTEMS(MAXSYSTEM)). SYSTEMS(ACTIVE) is the default. However, even for this setting, an informational message will indicate potential problems if the sysplex were to grow to MAXSYSTEM .

The parameters accepted:

SYSTEMS(ACTIVE | MAXSYSTEM) - specifies the size of the signalling configuration.

- ACTIVE
  Checks that all signaling structures in use by XCF are large enough to support the number of active systems in the sysplex.
- MAXSYSTEM
  Checks that all signaling structures in use by XCF are large enough to support the maximum number of systems that can be in the sysplex. MAXSYSTEM resolves to the value specified when the primary sysplex couple data set was formatted by the IXCL1DSU utility.
- The default is ACTIVE.

Output messages:

- IXCH0249I - all signalling structures sized sufficiently
- IXCH0248I - ACTIVE config OK, but MAXSYSTEM would fail
- IXCH0247E - at least one structure fails size requirements
- IXCH0915I - report of all signalling structures and their sizes. Issued with IXCH0247E, IXCH0248I, or in VERBOSE mode

Figure 25-17 shows the syntax for XCF_SIG_STR_SIZE.

```
UPDATE CHECK(IBMXCF,XCF_SIG_STR_SIZE)
       SEVERITY(MED) INTERVAL(002:00) DATE(20071101)
       PARM('SYSTEMS(ACTIVE)')
       REASON('XCF signaling structures should be of'
              'sufficient size to support all systems in the'
              'target sysplex.')
       VERBOSE(NO)
```

Figure 25-17 Syntax for XCF_SIG_STR_SIZE

The IXCH0915I report shows data highlights:

- Structures failing ACTIVE configuration with an * in column 1
- Structures that would fail MAXSYSTEM configuration with a dash in column 1
- Structures that will handle any number of systems joining the sysplex

Figure 25-18 shows an example of the IXCH0915I report.

```
                            Structure                   List
 Structure Name    CF Name       Size       Lists      Entries
 ---------------  --------  ----------  ----------   ----------
*IXCTL_SIGNAL03    LF02          14336          50*        1575
*IXC1              LF01          11264          64          880*
-IXC2              LF02          12288          64*        1136*
 IXCTL_SIGNAL02    LF01          18512          80         1680
```

Figure 25-18 Example of the IXCH0915I report

26

HLASM

This chapter describes enhancements to HLASM provided with z/OS V1R10 (also known as HLASM R6):

- HLASM services interface
- QY- and SY-type address constants
- Longer machine instruction and extended mnemonics
- Suffix tags for mnemonics
- Removal of O’ attribute reference from non-conditional-assembly statements
- Rollup of SPEs
- Integration of HLASM for Linux on zSeries

26.1 HLASM services interface

The need has arisen of letting I/O exits and external functions request basic services in a platform-independent way, as is the case for:

- Obtain/free memory
- Get time/date data
- “WTO”-like service

The solution introduced by HLASM R6 is of adding a word to the end of the primary parameter list pointing to the services interface block.

This opens the possibility of writing exits and functions to run on all platforms, with HLASM doing the work for the programmer.

26.2 QY- and SY-type address constants

The relocation dictionary (RLD) contains an entry for each address constant that must be modified before a module is executed or requires adjustment during the binding process. The entry specifies both the address constant location within a section and the external symbol used to compute the value of the address constant. (The external symbol can be defined in an ESD entry in another section.)

The binder uses the RLD to adjust (relocate) the address constants for references to other control sections or elements. The RLD is also used to readjust these address constants after the program management loader reads a program object or load module from a program library into virtual storage for execution.

There are new RLD entries in z/OS V1R10, as follows:

- SY-type:
  - Three-byte long-displacement S-type constant
  - DC SY(*) produces BdddDD, where ddd is the traditional 12-bit displacement and DD is the high-order 8 bits. The constant is halfword aligned.
- QY-type:

  QY-con type is an assembler notation that supports long-displacement type instructions in which the displacement is held in discontiguous bytes (DL-DH). This support is provided in the z/OS V1R10 variant of the PO5 format and later formats.
  - 20-bit long-displacement offset constant.
  - Implemented to let C++ replace three instructions by one for references to items in a function’s writable static area.
  - The typical use in compiled code would be something like the following:

    ```
    LY  5,QY(Static_Var).
    ```

    Such use in Assembler statements (expected to be rare) requires the coder to construct the pieces of the instruction. The constant is halfword aligned.
  - The compiler (or assembler) provides the base register in the instruction using the QY-con.

These two new types of address constants trigger enhancements such as the following:

- SY-cons extends support for z/Architecture instructions.

  The binary floating-point facility in ESA/390 introduced the RXE instruction format.

  Bits 8-31 of the RXE format provide the same register, base, index, and displacement fields as the RX format. However, the opcode is 16 bits, split between the first and last bytes of the instruction. Bits 32-39 of the instruction are reserved; see Figure 26-1.

  With the advent of z/Architecture, the RS instruction format was extended in a similar manner to form the RSE format. The RSE and RXE instruction formats were used extensively in implementing the new 64-bit architecture.

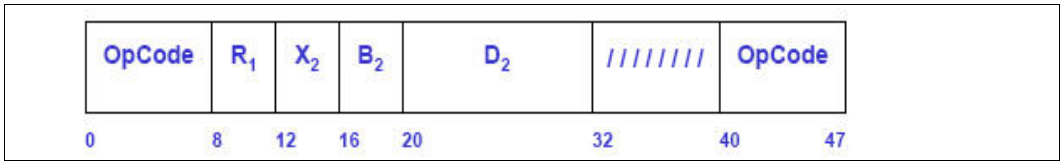


Figure 26-1 The RXE instruction format

The long-displacement facility builds upon the RSE and RXE instruction formats introduced in z/Architecture. The new RSY and RXY instruction formats (Figure 26-2) have all of the same fields as the RSE and RXE instructions, but with an additional field occupying the previously-reserved bits 32-39. A new SIY format, a long-displacement analog to the SI format, is also introduced.

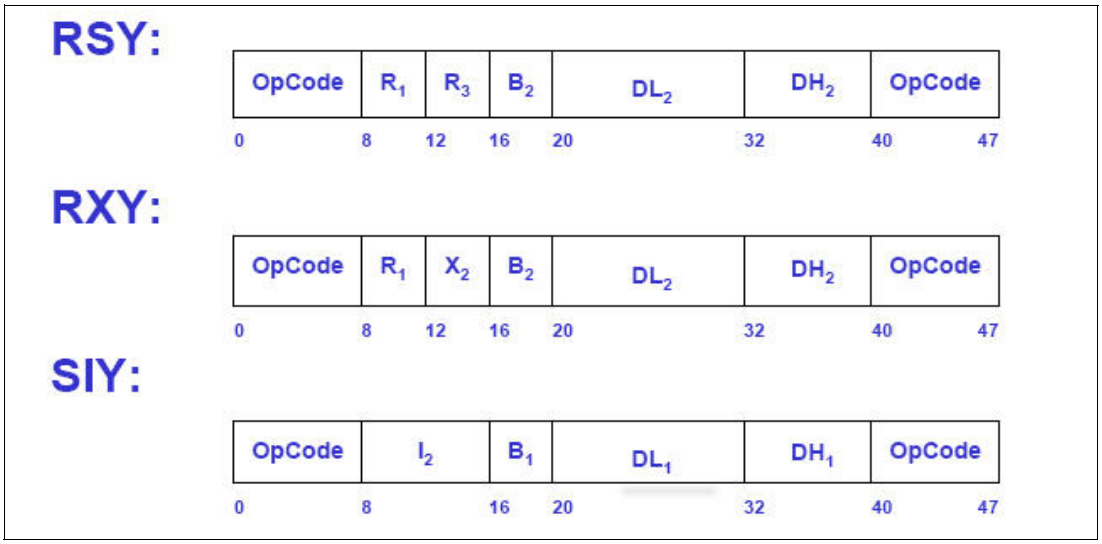


Figure 26-2 The new RSY and RXY instruction formats

Prior to the long-displacement facility, the displacement field in an instruction was a 12-bit unsigned field, providing a displacement range from 0-4,095 bytes.

The new formats contain a 20-bit signed displacement (Figure 26-3 on page 546, thus providing a positive or negative displacement of 512K.

Bits 32-39 of the instruction form the displacement high (DH) field that provides the most-significant bits of the displacement. Bit 32 is the sign bit. The DH field, concatenated with the displacement low field (DL, that is, the classic 12-bit displacement in bits 20-31) form the 20-bit signed value.

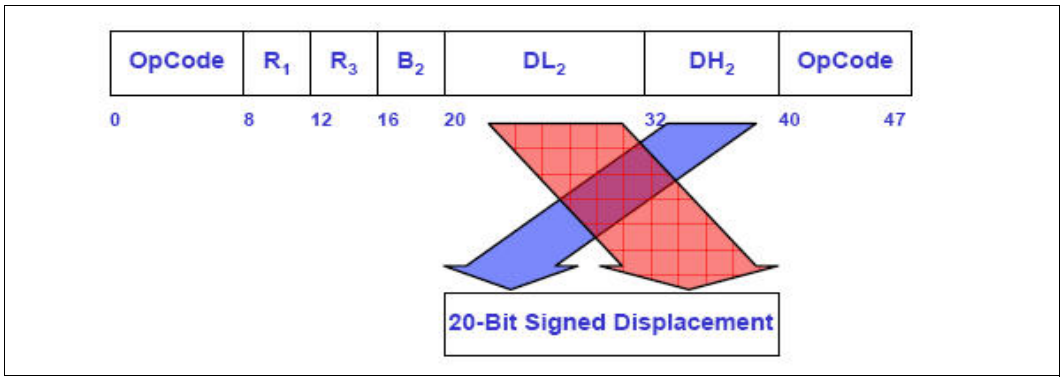


Figure 26-3 The 20-bit signed displacement

A significant number of the z/Architecture RSE and RXE instructions (that is, those that provided 64-bit support) were converted to long displacement (RSY and RXY format). Instruction-level compatibility for programs developed using 12-bit displacements is assured, since HLASM will generate zeros for the reserved fields.

Decimal and floating-point operations were not converted to long displacement.

New instructions were defined to provide long-displacement analogs for most of the 32-bit RS, RX, and SI instructions (that is, those ported to Z from ESA/390). The letter “Y” was appended to the mnemonic to indicate the long-displacement form. For example, the 32-bit LOAD includes both L and LY.

- QY-cons is expected to provide performance enhancements.

26.3 Longer machine and extended mnemonics

Support for new z/Architecture instructions is provided, which leads to:

- Basic mnemonics with up to 5 characters
- Extended mnemonics, which adds E, NE, H, NH, L, NL suffixes

There are, however, possible conflicts with existing macro names, as can happen with any new machine instructions.

There are also possible problems with tools assuming that instruction mnemonics will never exceed 5 characters.

26.3.1 Suffix tags for instruction mnemonics

HLASM now supports mnemonic tags:

- Mnemonic:asm always uses the definition in the assembler’s opcode table, whether or not a macro of the same name has been defined.
- Mnemonic:mac always uses a macro definition, whether previously defined or found by library search.
- Suffixes may be abbreviated to one or two characters.

Such mnemonic tags allow coexistence of macros and “built-in” instructions with the same name.

Because this new facility opens the potential to use identical mnemonics, it avoids the need to rename macros, or use lower-level opcode tables.

Another solution is to specify ACONTROL OPTABLE in source code.

Following is an example of the ACONTROL OPTABLE solution using a macro to simulate “Load and Test”:

```
LT O,X Now generates machine code for the new LT instruction
ACONTROL OPTABLE(ZOP) (or some other opcode table not supporting LT)
LT O,X Invokes the LT macro (heavily used in VM/CMS, for example)
```

26.3.2 Removal of O’ Attribute reference from non-conditional assembly

Opcode attribute is valid in any conditional assembly statement.

In macros or in “open code”, it cannot return a usable result in non-conditional contexts.

For example:

```
X    DC    A(O'X) No useful meaning
```

There is no impact expected. One can easily emulate current behavior with an intermediate SETC assignment.

Use of O’ in non-conditional code returns nonsensical results.

Emulation:

```
&Temp SETC O'X
X DC A(&Temp.)
```

It is being removed to avoid possible pitfalls for unwary programmers.

26.4 Rollup of SPEs

Many minor enhancements made to HLASM R5 since 2004 are incorporated into HLASM R6:

- Manuals will be updated for R6.
- The WORKFILE option reinstates use of the SYSUT1 utility file.

  WORKFILE option: by default, HLASM uses only virtual storage; large assemblies should specify WORKFILE to enable the SYSUT1 utility file.
- Constant types for decimal floating point.

  Constant types: ED, DD, LD for decimal float; LQ for quadword-aligned hex float.
- Options: FLAG(PAGE0) is being improved.
- Enhanced statements: EXTRN/WXTRN supports declaration of external “Parts” (items in C/C++ writable static area).
- New symbol attributes: ASSEMBLER and PROGRAM.
- Other minor items:
  - Symbol XREF updates, unsigned binary integers
  - 12 new messages

- Enhanced Structured Programming Macros in HLASM Toolkit is provided with Structured Programming macros:
  - ASMMREL controls generation of based or relative branch instructions
  - ASMLEAVE exits from one or more enclosing DOs
  - ELSEIF allows multiple IF alternatives
  - ITERATE branches to ENDDO of any enclosing DO
  - DO with no operand for statement grouping
  - NEXTWHEN for continued WHEN processing
  - DOEXIT can specify which DO to exit
  - Added checks for proper nesting

26.5 Integration of HLASM for Linux on zSeries

HLASM for Linux on zSeries is currently available as PRPQ 5700-TCQ with ordering and service rather complex.

With HASLM R6 it is now a priced feature of HLASM.

27

Console services

With z/OS V1R10, the final phase of console restructure is implemented. In this final phase, console processing was redesigned to reduce serialization contention by reducing the scope of serialization for many operations from a console class down to an individual console. Additionally, support is provided to increase the maximum number of MCS, SMCS, and subsystem consoles in a sysplex from 99 per sysplex to 99 active consoles per system; also, defining up to 250 consoles per system is supported (of which up to 99 may be concurrently active), and wildcard support is added for the DISPLAY CONSOLES command along with improved command response messages.

This chapter describes the enhancements to the console service functions, as follows:

- Reduce ENQ contention problems caused by extended use of the sysplex serialization while transporting console data
- Scope of console data serialization reduced
- Number of available MCS, SMCS, and subsystem consoles increased
- Continued recovery enhancement
- Improved component diagnostics

27.1 Console support

z/OS console support can be operated in one of two modes: shared mode and distributed mode. *Consoles shared mode*, as it is currently called, was originally introduced in MVS/ESA. Every system maintained an identical view of the consoles global data which was divided into five classes, each of which was governed by an ENQ.

Therefore, if one system wished to activate a console, it would acquire a sysplex-wide ENQ, and make the appropriate change locally. The change would then be broadcast to the other systems in the sysplex who would need to acknowledge receipt of the data. When all the systems had received the update, the originating system would then cause a commit to be initiated, and each system would again have to notify the originator that the data had been committed. At that point the resource could be released.

When a system joined the sysplex, or found itself with an old version of the data, it would require all the console data to be sent to it while it held the global serialization.

Shared mode is the name given to the way that z/OS console support has always operated in a sysplex environment. As sysplex environments have grown larger, various shortcomings in shared mode have been identified and are addressed by the distributed mode of operation.

27.1.1 Console support with z/OS V1R10

The new console data structures no longer require sysplex-wide serialization to make most updates. The system on which a console is active owns the data, and will hold ENQ SYSZCNZ CONNAME#consname. The other systems may not display the newest data.

Distributed mode

Distributed mode is a new mode of console operation. Distributed mode improves the operation of z/OS console support when running in a sysplex environment by:

- Potentially reducing the time that it takes to IPL a system.
- Potentially reducing the time that it takes a system to join a sysplex.
- Potentially reducing the scope of console related hangs.
- Reducing the possibility of console related outages.
- Allowing more MCS, SMCS, and subsystem consoles to be configured.

The current mode status can be displayed with the DISPLAY OPDATA,MODE command. This command will indicate the current mode, whether the systems are capable of migrating, the current status of a migration, and whether the sysplex would currently be able to migrate.

In Figure 27-1 on page 551, system SC63 is a z/OS V1R9 system and the sysplex cannot be migrated to distributed mode. This is indicated by the message shown in the display for the command:

```
SYSPLEX ABLE TO MIGRATE: NO
```

```
D OPDATA,MODE
CNZ9006I 12.03.23 DISPLAY 0,MODE 714
CURRENT: SHARED
SYSTEM   MIGRATION STATUS
SC63     z/OS LEVEL DOES NOT SUPPORT MIGRATION
SC65     MIGRATION SUPPORTED
SC70     MIGRATION SUPPORTED
SC64     MIGRATION SUPPORTED
SYSPLEX ABLE TO MIGRATE: NO
```

Figure 27-1 D OPDATA,MODE in a mixed-level sysplex

Console dataspace

Distributed mode defines many of the console data structures in a dataspace where they are accessible only through controlled, well-defined services, drastically reducing the possibility that an errant program can damage the data structures and cause an outage. All of the new distributed mode programming includes robust new error handling and recovery capabilities.

Consoles per sysplex image

Distributed mode relieves the 99 MCS, SMCS and subsystem console per sysplex constraint by allowing up to 99 MCS, SMCS and subsystem consoles to be active per z/OS image. Additional flexibility is provided by allowing up to 250 MCS, SMCS and subsystem consoles to be defined per z/OS image (of which up to 99 may be concurrently active on that image). The number of MCS, SMCS and subsystem consoles that may be concurrently active in a sysplex is now a function of the 99 active console limit of each image multiplied by the number of z/OS images in the sysplex.

27.1.2 Migration to distributed mode

To take advantage of distributed mode, you must perform a sysplex-wide migration to distributed mode. You can do so through a sysplex-wide IPL or dynamically, while you are running. The ability to fall back dynamically to the shared mode of operation is also provided. However, the fallback is potentially disruptive to your operations.

In shared mode, any CONSOLxx CONSOLE definition after the 99th console in the sysplex is rejected.

In distributed mode, each system is allowed to define 250 (or 251 if one is for the system console) consoles. They can be any combination of MCS, SMCS, and SUBSYSTEM consoles. As in shared mode, the SMCS and SUBSYSTEM definitions are available to any system in the sysplex. The MCS definitions now define how the console behaves on that system. Another system may have a unique definition (as long as it is an MCS console) that governs the console's behavior on that system.

Migration implementation

To perform the migration, all of your systems must be at the z/OS V1R10 level or higher. Figure 27-2 on page 552 indicates that all systems in the sysplex are at the z/OS V1R10 or higher level and the console mode can be changed to distributed mode:

```
SYSPLEX ABLE TO MIGRATE: YES
```

```
D OPDATA,MODE
CNZ9006I 12.03.23 DISPLAY 0,MODE 714
CURRENT: SHARED
SYSTEM   MIGRATION STATUS
SC63     MIGRATION SUPPORTED
SC65     MIGRATION SUPPORTED
SC70     MIGRATION SUPPORTED
SC64     MIGRATION SUPPORTED
SYSPLEX ABLE TO MIGRATE: YES
```

Figure 27-2 D OPDATA,MODE in an all z/OS V1R10 sysplex

Distributed mode can be entered in one of the following ways:

- The single system, or first system in the sysplex can specify DISTRIBUTED as part of the CON= system parameter during the IPL

  The IPL prompt CON= parameter specifies to operate in DISTRIBUTED mode or to continue to operate in SHARED mode1:

  ```
  CON=(...,DISTRIBUTED)

  CON=(...,SHARED)
  ```

- In a sysplex, the SETCON MODE=DISTRIBUTED command can be executed. The mode cannot be changed in a single system (Local-mode or Monoplex). To move to distributed mode all systems must first be at z/OS V1R10 or higher. The migration is aborted if there are any lower-level systems in the sysplex.

  There should be no change to the operation of the sysplex when moving to distributed mode. All the console data from shared mode will be converted to the appropriate distributed mode structures.

Fallback to shared mode

Falling back to shared mode, on the other hand, is a potentially destructive operation. It was designed for emergency situations in cases where being in distributed mode was distressing for the system. If there have been few changes to the consoles in the sysplex since moving to distributed mode, then the movement back to shared will probably be painless. If, however, additional consoles have been defined to the sysplex such that there are now more than 99 MCS/SMCS/SS consoles defined, then some of the definitions will be lost during the migration. If any active consoles will be affected, the operator is prompted to see if they really want to continue. The following conditions occur on the fallback:

- MCS, SMCS and subsystem consoles beyond the first 99 are deleted, even if they are active.
- The operator is prompted if active consoles are going to be deleted.
- There are no changes to EMCS consoles.
- Subsystems owning a console are not informed of the loss of their console.

27.1.3 DISPLAY CONSOLES command

The changes to the DISPLAY CONSOLES command can be used to illustrate some of the changes in the console support. These changes apply to both modes and are shown in Figure 27-3 on page 553, as follows:

- The message ID has changed to CNZ4100I.

- The console ID has been removed from the display. Since the console name is the preferred external, and there is no longer any external way to use the one-byte IDs, there is no longer a need to show the console ID.
- The enhanced output shows an echo of the executed command, and the type of console (MCS, SMCS, SUBSYS, EMCS, or SYSCONS) is clearly indicated.
- Other data has been consolidated and moved to make it easier to recognize, such as the device and system where active had been on the left of the display under the console name. The device has been given its own eyecatcher of DEV=, and the system where active has been combined with the console status (ACT, INACT) with the eyecatcher of STATUS= which replaces the COND=.
- The DEFINED=, MATCHED= and ATTRIBUTES ON lines show the system name. DEFINED lists the systems on which this console has been defined. SMCS and SUBSYS consoles will always indicate (*ALL). MATCHED= indicates which definitions of the console match any filters on the issued command. In shared mode MATCHED will always be the same as DEFINED. Finally, ATTRIBUTES ON sysname will delineate each definition of the console.

```
D C,CN=MCS653E0
CNZ4100I 16.11.15 CONSOLE DISPLAY
CONSOLES MATCHING COMMAND: D C,CN=MCS653E0
MSG:CURR=1    LIM=3000 RPLY:CURR=4    LIM=9999  SYS=SY1      PFK=NONE
MCS653E0  TYPE=MCS      STATUS=ACT-SC65
          DEFINED=(SC65)
          MATCHED=(SC65)
   ATTRIBUTES ON SC65
      AUTH=(MASTER)    CMDSYS=*             NBUF=1
      DEV=03E0         LOGON=OPTIONAL       USERID=N/A
      MFORM=(S)        AREA=(Z)             PFKTAB=01
      USE=FC  DEL=RD   RTME=1/4  RNUM=25    SEG=14    CON=N
      LEVEL=(ALL)
      MONITOR=(NONE)                        INTIDS=N  UNKNIDS=N
      ROUT=(ALL)
      MSCOPE=(SC65,SC70)
```

Figure 27-3 DISPLAY CONSOLES command in z/OS V1R10

Console defined on two systems

In Figure 27-4 on page 554, there is a console named MCSCONS which is defined to two systems in the sysplex that is running in distributed mode. The following differences can be noted in the console definition:

- The console is at a different device number on the two systems. This condition is also possible in shared mode.
- The ROUT= shows a different set of routing codes. This can only happen in distributed mode.

Attention: This console can only be active on one system at a time.

```
D C,CN=MCSCONS,FULL
CNZ4100I 21.52.39 CONSOLE DISPLAY 495
CONSOLES MATCHING COMMAND: D C,CN=MCSCONS,FULL
MSG:CURR=0    LIM=3000 RPLY:CURR=3    LIM=9999  SYS=SC65     PFK=NONE
MCSCONS   TYPE=MCS      STATUS=INACT
          DEFINED=(SC65,SC70)
          MATCHED=(SC65,SC70)
   ATTRIBUTES ON SC65
      AUTH=(ALL)       CMDSYS=*            NBUF=0
      DEV=0510         LOGON=DEFAULT       USERID=N/A
      MFORM=(S)        AREA=(Z,A)          PFKTAB=01
      USE=FC  DEL=RD   RTME=1/4  RNUM=25   SEG=*     CON=N
      LEVEL=(ALL)
      MONITOR=(NONE)                       INTIDS=N  UNKNIDS=N
      ROUT=(ALL)
      MSCOPE=(*ALL)
   ATTRIBUTES ON SC70
      AUTH=(ALL)       CMDSYS=*            NBUF=0
      DEV=03D3         LOGON=DEFAULT       USERID=N/A
      MFORM=(S)        AREA=(Z,A)          PFKTAB=01
      USE=FC  DEL=RD   RTME=1/4  RNUM=25   SEG=*     CON=N
      LEVEL=(ALL)
      MONITOR=(NONE)                       INTIDS=N  UNKNIDS=N
      ROUT=(1-10)
      MSCOPE=(*ALL)
```

Figure 27-4 Display of a console defined to two systems

Additional console command changes

There are changes to console commands with z/OS V1R10, as follows:

- RESET CN command

  The purpose of the RESET CN command is to return a console to a “pristine” state such that it can be used again. It is usually used in error situations when a console has become unusable. The scope of the command has been enhanced to also have some effect on subsystem and inactive EMCS consoles.

- VARY CN command

  Only active consoles can now be modified (with an exception for LU and LOGON for an SMCS console). In distributed mode, modifications will not be permanent. It will no longer be possible to alter most attributes of consoles if the console is inactive. In addition, in distributed mode, any changes made to the console will not be remembered when the console deactivates. The next activation will be with the original attributes.

- CMDS DUMP command

  The CMDS DUMP command can be used to cause the CONSOLE and *MASTER* address space to be dumped before removing hung commands. A dump of the CONSOLE and *MASTER* address spaces can be requested before a command is removed.

28

z/OS BCP allocation

System programmers need more control in making tape devices available or unavailable to allocation. There are many reasons for this special need. For example, a device is being serviced, or the list of devices in message IEF877E is too long and confusing, or devices shared among systems need to be restricted to only one system. In all these cases, it would be helpful if we can somehow mark specific devices to let allocation know that it should not use those devices.

Opening a very large number of data sets by an application is causing problems in allocation, in the way these requests are queued.

This chapter describes the changes in z/OS V1R10 to allocation:

- Recovery allocation command changes
- Allocation of a large number of data sets

28.1 Recovery allocation command changes

Because recovery allocation can bring devices online, varying an eligible device offline would not prevent allocation from using the device. It is possible, for a dynamic allocation request, to ask allocation to either consider *all* offline devices or *no* offline devices, via the S99OFFLN flag. There is no intermediate case where allocation will use only *some* offline devices. System programmers need to be able to mark specific devices unavailable for use by the system. Of course, this change to unavailable has to be temporary. There should be an easy way to change unavailable devices back to available.

With z/OS V1R10, new support provides a method to make tapes have a new status, that is, unavailable. The changes are to:

- Display the new device status.
- Make sure that devices do not become unavailable as the system tries to use them by controlling the availability of offline tape devices.
- Make sure the devices are not unavailable and online at the same time by using proper serialization.

DDRSWAP and recovery allocation are two system functions that can bring online an eligible offline device and use it. They can use only available devices.

Message IEF877E

Message IEF877E is issued by recovery allocation, along with message IEF238D. It lists all eligible offline devices for the allocation request, putting the devices in different categories, like those that are offline and inaccessible, those that are offline and in an offline library and so on. This message has a lot of useful information, but sometimes it can be too much information. The list of devices can become too long and confusing, especially with increasing numbers of devices in libraries and esoterics. If some of the devices are marked unavailable, message IEF877E would have a shorter list of eligible devices.

28.1.1 New VARY commands

Using changes to the operator VARY commands, IEEVARYD service options, you can vary one or more offline tape devices unavailable and available. The UNAVAIL command sets the UCBUNAVL bit, and the AVAIL command resets it. UNAVAILABLE and AVAILABLE are valid keywords and can be used instead of UNAVAIL and AVAIL, respectively. Some examples are as follows:

```
V 5B4,UNAVAILABLE
CNZ6000I DEVICE 05B4 IS NOW UNAVAILABLE
VARY 5B4-5B6,AVAIL
IEE457I 10.58.57 UNIT STATUS 890
UNIT TYPE STATUS        VOLSER     VOLSTATE
05B4 3490 OFFLINE                  /REMOV
05B5 3490 OFFLINE                  /REMOV
05B6 3490 OFFLINE                  /REMOV
```

To mark the device unavailable with the IEEVARYD service, indicate in the VDEV block:

```
VDEV_UNAVAIL = ON      VDEV_ON  = ON
```

To make the device available:

```
VDEV_UNAVAIL = ON      VDEV_OFF = ON
```

Note: IEEVARYD varies one or more devices online or offline on a single system, or defines the automatically switchable attribute for a device that supports automatic tape switching. It has the same effect as the VARY device or VARY AUTOSWITCH operator command, but it provides return and reason codes to the calling program, rather than issuing messages to a console.

New VARY command options

The new VARY command options to mark offline tape devices unavailable and available are as follows:

```
VARY dev,UNAVAIL
VARY devn-devm,UNAVAIL
VARY (dev1,dev2,dev3,..devn),UNAVAIL
VARY dev,AVAIL
VARY devn-devm,AVAIL
VARY (dev1,dev2,dev3,..devn),AVAIL
```

New device display command changes

The purpose of this new support is to not let allocation use unavailable devices and to not allow DDR SWAP, if the to-device is an offline tape device that is marked unavailable, to be able to obtain it.

To display the new UNAVL status of the unavailable devices, use any of the following commands:

```
D U,,,dev
D  U,,,dev,number
D U,TAPE,UNAVAIL,dev,number
D U,,UNAVAIL,dev
```

An example of the display unit command is:

```
D U,,,5B2,1
  IEE457I 17.23.12 UNIT STATUS
 UNIT TYPE STATUS        VOLSER     VOLSTATE
  05B2 3490 F-UNAVL                     /REMOV
D U,TAPE,UNAVAIL
IEE457I 17.20.54 UNIT STATUS
UNIT TYPE UNIT TYPE UNIT TYPE UNIT TYPE
05B4 3490 05B5 3490 05B6 3490 05B7 3490
D U,,UNAVAILABLE,5A0,4
 IEE457I 17.21.58 UNIT STATUS
 UNIT TYPE UNIT TYPE UNIT TYPE UNIT TYPE
 05A0 348S 05A1 348S 05A2 3480 05A3 3480
```

Note: UNAVL is the change to the IEE457I message for a device that has been marked unavailable for allocation by the VARY xxxx,UNAVAIL operator command.

28.2 Allocating a large number of data sets

Recent testing efforts showed that when DB2 tried to open thousands of data sets at one time, as it does when it is restarting and trying to get back to its “previous operating state”, the time it takes to open the 64 thousandth data set was significantly longer than the first few data

sets it opened. Analysis showed that allocation, catalog management, and VSAM OPEN were affected by the linear queue that allocation creates to describe all of the outstanding allocated data sets.

z/OS V1R10 DSAB hash table

The solution was for allocation to create another structure with better performance capabilities to avoid searching the DSAB queue. Existing allocation services, such as GETDSAB and IEFDDSRV, were updated to use this structure. Programs simply need to use the documented services (instead of locating and searching the DSAB queue) to obtain the benefits of this line item.

This creates a high-performance structure for products to locate allocation information (DSAB and TIOT entry), given a DD name. The enhancement to existing external services (GETDSAB, IEFDDSRV) now utilizes new hash tables. This is designed to significantly reduce queue search times by OPEN and catalog, providing value to DB2 and any other environment with many data sets.

Using the new function provides the capability to allow products to create environments with more data sets, without worrying about the elapsed time to allocate and open them. DB2 may open tens of thousands of data sets during operation and during restart. The value of this is that it ensures the continued scalability of allocation and the subsystems and products that use its control blocks, such as catalog and Open/Close/End-of-volume. It may even provide better performance for these services and lead to reduced DB2 restart time.

Application use of new services

Any allocation request, whether by batch JCL, dynamic allocation by a program, or the TSO ALLOCATE command will invoke the support provided the new service updates to build new structures that speed access of the allocation information. The GETDSAB and IEFDDSRV services, invoked by products and programs, as well as (potentially) application programs, will utilize the new structures to reduce the time needed to locate a DSAB or TIOT entry, given the DD name.

The GETDSAB and IEFDDSRV services are called by:

- VSAM open
- Open/Close/End-of-volume
- Catalog
- Vendor products
- Application programs authorized for GETDSAB or any program for IEFDDSRV

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 561. Note that some of the documents referenced here may be available in softcopy only.

- *HFS to zFS Migration Tool*, REDP-4328-00
- *z10 EC Configuration Setup*, SG24-7571

Other publications

These publications are also relevant as further information sources:

- *z/OS MVS Extended Addressability Guide*, SA22-7614
- *High Level Assembler for z/OS & z/VM & z/VSE Programmer’s Guide*, SC26-4941
- *z/OS Migration*, GA22-7499
- *z/OS Introduction and Release Guide*, GA22-7502
- *z/OS Program Directory*, GI10-0670
- *z/OS License Program Specifications*, GA22-7503
- *z/OS MVS Planning: Operation*, SA22-7601
- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS UNIX System Services Planning*, GA22-7800
- *ServerPac: Using the Installation Dialog*, SA22-7815
- *IBM Health Checker for z/OS User’s Guide*, SA22-7994
- *z/OS MVS Planning: Global Resource Serialization*, SA22-7600
- *z/OS MVS Diagnosis: Reference*, GA22-7588
- *z/OS DFSMS Advanced Copy Services*, SC35-0428
- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407
- *z/OS DFSMS Installation Exits*, SC26-7396
- *z/OS DFSMS Introduction*, SC26-7397
- *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408
- *z/OS DFSMS Managing Catalogs*, SC26-7409
- *z/OS DFSMS Storage Administration Reference (for DFSMSdfp, DFSMSdss, DFSMShsm)*, SC26-7402
- *z/OS DFSMS Using the New Functions*, SC26-7473

- *z/OS DFSMS Using Data Sets*, SC26-7410
- *z/OS DFSMS Using ISMF*, SC26-7411
- *z/OS DFSMSdfp Advanced Services*, SC26-7400
- *z/OS DFSMSdfp Diagnosis*, GY27-7618
- *z/OS DFSMSdfp Utilities*, SC26-7414
- *z/OS DFSMSdss Storage Administration Guide*, SC35-0423
- *z/OS DFSMShsm Diagnosis*, GC52-1083
- *z/OS DFSMShsm Implementation and Customization Guide*, SC35-0418
- *z/OS DFSMShsm Managing Your Own Data*, SC35-0420
- *z/OS DFSMShsm Storage Administration Guide*, SC35-0421
- *z/OS MVS Programming Assembler Services Reference*, SA22-7606
- *z/OS Common Information Model User's Guide*, SC33-7998
- *z/OS MVS Capacity Provisioning User's Guide*, SA33-8299
- *z/OS XML System Services User's Guide*, SA23-1350
- *z/OS TSO/E REXX User's Guide*, SA22-7791
- *z/OS MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*, SA22-7609
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference,* SA22-7803
- *z/OS DFS/SMB/zFS Messages and Codes*, SC24-5917
- *z/OS MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDTINFO-IXGWRITE)*, SA22-7610
- *z/OS Resource Management Facility (RMF) Performance Management Guide*, SC33-7992
- *z/OS Resource Management Facility (RMF) Report Analysis*, SC33-7991
- *z/OS Resource Management Facility (RMF) User's Guide*, SC33-7990
- *z/OS XL C/C++ User's Guide*, SC09-4767

Online resources

These Web sites are also relevant as further information sources:

- The Cryptographic Support for z/OS V1R7 through z/OS V1R9 and z/OS.e V1R7 through z/OS.e V1R8 Web deliverable is available. This Web deliverable supports z/OS V1R7 through z/OS V1R9 and z/OS.e V1R7 through z/OS.e V1R8. To obtain this Web deliverable, see

  `http://www.ibm.com/server/eserver/zseries/zos/downloads`

- z/OS and z/OS.e Web deliverables are available from:

  `http://www.ibm.com/eserver/zseries/zos/downloads/`

- The recommended exclusion list and list of DFSMS instances for the migration assistance tracker are available on a Web site for downloading:

  `http://www-03.ibm.com/servers/eserver/zseries/zos/downloads/`

- The new version of the OpenPegasus OpenSource CIM Server implementation (version 2.7) can be obtained from the following Web sites:

  http://www.dmtf.org/standards/cim
  http://www.openpegasus.org

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

F

G

H

N

O

P

R

S

T

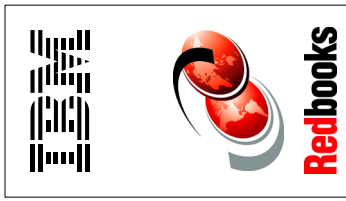
U

V

W

X

Z



z/OS Version 1 Release 10 Implementation

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

z/OS Version 1 Release 10 Implementation

EAV volumes, 64-bit common, WLM, JES3, JES2

CIM, XML, RRS, SMP/E, LE, z/OS UNIX

HiperDispatch, IOS, ISPF, XCF/XES

This IBM Redbooks publication positions the new z/OS Version 1 Release 10 for migration by discussing many of the new functions that are available. The goal for the z/OS platform is to eliminate, automate, and simplify tasks without sacrificing z/OS strengths, and to deliver a z/OS management facility that is easy to learn and use.

This release of the operating system supports new capabilities that are designed to provide the following enhanced and new functions:

- An architectural limit of hundreds of terabytes (TBs) for DASD volumes, up from the current limit of approximately 54 GB per volume. Known as Extended Address Volume (EAV), this function is planned to initially support 223 GB per volume on z/OS V1R10 and IBM System Storage DS8000, when available.
- Up to 64 engines, up to 1.5 TB per server with up to 1.0 TB of real memory per LPAR, and support for large (1 MB) pages on the z10 EC, providing performance for critical workloads.
- HiperDispatch can help provide increased scalability and performance of higher n-way z10 EC systems by improving the way workload is dispatched within the server.
- A new Basic HyperSwap capability (to be enabled by TotalStorage Productivity Center for Replication Basic Edition for System z).
- Improved management for processor capacity via a new Capacity Provisioning Manager.
- Improved productivity with simplifying diagnosis and problem determination.
- Expanded Health Check services.
- Automatic dump and re-IPL capability.
- zIIP-assisted z/OS Global Mirror (XRC) and additional z/OS XML System Services exploitation of zIIP and zAAP help make these workloads more attractive on System z.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7605-00 ISBN 0738432288