IBM

z/OS Version 1 Release 11 Implementation

- EAV volumes, HCD/HCM, HyperSwap, JES2, HiperDispatch, PFA
- JES3, SDSF, Service Aids, GRS, LE, System REXX, STP
- z/OS UNIX, zFS, ISPF, RRS, CIM, XCF, SMF, DCM

Paul Rogers
Robert Hering
George Kozakos
Lutz Kuehner
Jean-Louis Lafitte
Jaqueline Mourao
Edison da Silva Nunes
Gil Peleg
Evanir Philipi
Giancarlo Rudolfi

Redbooks

ibm.com/redbooks

International Technical Support Organization

z/OS Version 1 Release 11 Implementation

March 2010

SG24-7729-00

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

First Edition (March 2010)

This edition applies to Version 1 Release 11 of z/OS (5694-A01) and to 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:

1-2-3®
CICS®
DB2®
DS6000™
DS8000®
ECKD™
Enterprise Storage Server®
ESCON®
eServer™
FICON®
FlashCopy®
GDPS®
HiperSockets™
HyperSwap®
IBM®
IMS™
Language Environment®
Lotus®
MQSeries®
NetView®
OS/390®
Parallel Sysplex®
PR/SM™
Predictive Failure Analysis®
RACF®
Redbooks®
Redbooks (logo) ®
Resource Link™
RETAIN®
S/390®
Sysplex Timer®
System Storage™
System z10™
System z9®
System z®
System/390®
SystemPac®
TDMF®
Tivoli®
TotalStorage®
VM/ESA®
VTAM®
WebSphere®
z/Architecture®
z/OS®
z/VM®
z/VSE™
z9®
zSeries®

The following terms are trademarks of other companies:

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

ACS, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

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

Java, 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.

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 11 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.

z/OS is a highly secure, scalable, high-performance enterprise operating system on which to build and deploy Internet- and Java™-enabled applications, providing a comprehensive and diverse application execution environment.

This book describes the following new and changed functions:

- IBM z/OS Management Facility
- Allocation enhancements in z/OS V1R11
- BCPii function enhancements in z/OS V1R11
- JES2 and JES3 enhancements
- zFS file sharing enhancements
- Extended access volume enhancements
- Choosing whether to run zAAP work on zIIP processors
- System REXX enhancements in V1R11
- RRS global panel options
- Service aids enhancements in V1R11
- GRS ENQ contention notification enhancements and analysis for GRS latches
- Basic HyperSwap® support enhancement
- Message Flood Automation enhancements
- Program Management new Binder IEWPARMS
- Predictive Failure Analysis® (PFA)
- SMF enhancements in V1R11
- System Logger enhancements
- XCF/XES enhancements in V1R11
- AutoIPL support
- Displaying PDSE caching statistics
- ISPF enhancements
- IBM Health Checker for z/OS enhancements

The team who 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 21 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England for seven years, providing OS/390® and JES support for IBM EMEA. He also worked in the Washington Systems Center for three years, and has been with IBM for more than 42 years.

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

Lutz Kühner is a Senior System z® IT Specialist with the System Technology Group sales organization in IBM Germany who provides technical System z support for clients in the financial market. He has 23 years of experience in the mainframe field. His areas of expertise include z/OS and z/OS UNIX, CICS® and high availability topics in general. Lutz has written extensively about z/OS-related topics and has contributed to several Redbooks.

George Kozakos is a Senior IT Specialist with IBM Australia. He has more than 21 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.

Jean-Louis Lafitte is Senior IT Architect at GATE Informatic SA, an IBM Premier Business Partner in Switzerland. He has 38 years of experience with IBM Large Enterprise Systems, and has worked on different parallel machines (IBM RP3, CM2, KSR1, IBM SP1, SP2 and BG/L, P). He has been associated with Parallel Sysplex® since 1984. Jean-Louis holds a Ph.D. degree in Theoretical Computer Science and several patents in S/390® architecture. He is a member of ACM and IEEE.

Jacqueline Mourao is a Systems Programmer at Banco do Brasil, an IBM client. She has seven years of mainframe experience and holds a degree in Information Systems. Jacqueline's areas of expertise include z/OS, installation and maintenance, Parallel Sysplex and security.

Edison da Silva Nunes is a System Programmer at Banco do Brasil in Brazil. He has 20 years of experience in the mainframe field. Edison's areas of expertise include z/OS and Storage Management.

Gil Peleg is a mainframe system programmer and manager of PSMFS in Israel. He has 12 years of experience in mainframe system programming and in teaching mainframe-related courses. He holds a degree in Computer Science. Gil has written extensively about z/OS and was also part of the team who wrote the z/OS 1.8, 1.9, and 1.10 implementation Redbooks.

Evanir Philipi is a Certified z/OS IT Specialist working with IBM Global Services, Brazil. He has worked with IBM mainframes for 37 years. Evanir is a z/OS instructor and a z/OS Technical Leader (Banco do Brasil).

Giancarlo Rodolfi is a System z Technical Sales Specialist in Brazil. He has 23 years of experience in the zSeries® field. Giancarlo has written extensively about the z/OS Communication Server, security, and z/OS.

Thanks to the following people for their contributions to this project:

Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 publications 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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts
- Follow us on twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

z/OS V1R11 overview

This chapter describes the functional enhancements available with z/OS V1R11. We introduce the various themes and functional changes in this release, as follows:

- Failure avoidance

  Predictive Failure Analysis is designed to help provide early warning about system trends that can cause system or application impacts, in many cases before they impact your business. The output from predictive failure analyses can be used for reporting and logging or to drive responsive automation intended to prevent serious problems.

- Simplified management

  Updates to z/OS health checks, new migration health checks, new autonomics for defining and managing I/O, simplified application of network security, as well as other operational improvements are intended to help simplify systems management, improve administrator, operator, and developer productivity, and ultimately provide less opportunity for error.

- Responsive networking

  The new z/OS Communications Server has designs intended to improve networking in a Parallel Sysplex, thereby enabling more efficient workload distribution in a sysplex and helping to improve the quality of the load balancing in multitiered z/OS server and application environments.

- Trusted system

  z/OS V1.11 has numerous designs for implementing and integrating new security technologies. New key generation and archival functions are planned to enable you to generate and recover private keys from PKI Services. Support for new SSL functions is planned to be integrated in z/OS and updates to IBM Tivoli® Directory Server for z/OS are intended to help simplify the migration of security-related LDAP-based applications to z/OS, integrate them with RACF®, and ultimately enable unified enterprise-wide identity and access management.

- Accountability

  Superior measurement and data collection and reporting capabilities are updated and can be used for comprehensive risk management, auditing, and compliance plans. For example, a new identity propagation function can allow z/OS subsystems to associate

distributed identities to RACF for improved cross-platform interoperability and auditing capabilities.

- Storage scalability

  An additional data set type is supported in the extended addressing space (EAS) on an extended address volume (EAV). This allows better exploitation of the capacity of an EAV, which can be as large as 223 GB. This helps to relieve storage constraints as well as simplify storage management by providing the ability to manage fewer large volumes as opposed to many small volumes.

- Improved resource optimization and economics

  This means CIM client applications that use the CIM server on z/OS for basic information interchange (such as RMF, WLM, DFSMS, and BCP) or for cross-platform system management such as the System z Capacity Provisioning Manager and the z/OS Management Facility problem determination capability (see Statement of general direction) can benefit.

- z/OS release-to-release performance improvement

  For the past several releases z/OS has provided release-to-release performance improvements by systematically reducing or eliminating constraints and inefficiencies within the base operating system. z/OS V1.11 has designs to continue release-to-release improvements.

1.1 z/OS V1R11 themes

z/OS V1R11 provides a variety of improvements by addressing the functional enhancements in terms of themes.

1.1.1 Theme: Simplification

IBM has embarked on a long-term commitment to simplify the z/OS platform. The past several releases of z/OS delivered improvements in the areas of simplifying diagnosis and problem determination; network and security management; as well as overall z/OS, I/O configuration, sysplex and storage operations. These improvements can help simplify systems management, improve application programmer, system programmer, and operator productivity, and make the functions easier to understand and use.

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

1.1.2 Theme: Availability and Business Resiliency

Beyond server availability, the application and the data must be available with good performance as well. For the System z platform this means hardware, connectivity, operating system, subsystem, database, and application availability, too. z/OS, System z servers, and System Storage™ working together can provide outstanding availability. System z servers are designed to reduce planned and unplanned outages through the use of self-healing capabilities, redundant components, dynamic sparing, and the ability for concurrent upgrades and microcode changes.

With every release, z/OS continues to refine its error checking, fault tolerance, isolation, error recovery, and diagnostic capabilities. Beyond single system availability are 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.

1.1.3 Theme: Economics

Information Technology (IT) is woven into almost everything we do. The demands on IT solutions are greater than ever, often requiring the delivery of more service, value, or capabilities, in less time or with fewer resources. Accomplishing more with less can be achieved by adopting a z/OS platform solution that is designed to drive efficiencies and economies of scale; accommodate business needs through flexible, virtual, and autonomic capabilities; help reduce the risk of lost productivity, downtime, or security breaches; and enable business innovation through the extension of existing investments and adoption of newer technologies.

Just as important as the scale of the system is how it handles scalability. z/OS and its subsystems provide for scalability not only based on chip speeds, but for a single image, clustering, storage, and data handling. The System z environment provides a holistic and balanced approach to scalability, which means that your System z environment with z/OS is capable of handling growth of your user base, applications, business processes, and data processing needs.

In z/OS V1R11, there are performance improvements in a variety of areas. Some are focused on overall performance and others on client pain points. One area of continuing improvement is cache and memory management, and this will continue to be true for the foreseeable future. The time it takes to retrieve data from memory, although progressively shorter in the absolute sense on newer server models, has become progressively longer when measured in processor cycle time increments. In other words, though memory access is much faster than it used to be, processors spend more cycles waiting for it than they ever did. This makes the system's management of cache very important. In z/OS V1R10, HiperDispatch was directed toward improving cache management in multiprocessing LPARs. As in past releases, z/OS V1R11 continues to focus on improved cache management by splitting cache lines to minimize cross-invalidation and aligning fields within cache line boundaries when appropriate to minimize the number of cache lines consumed.

1.1.4 Theme: Business Integration, Application Integration

The z/OS platform supports many new application development technologies, such as Java, Perl, PHP, XML, Unicode, HTML, SOAP, and other Web services in addition to C/C++ and other application development tools. And z/OS continues to update its traditional application development tools as well. Improvement in the application development stack can reduce deployment and maintenance costs for applications deployed on the z/OS platform by bringing the leveraging technical capabilities of the z/OS software stack.

Data serving

Like other operating systems, z/OS provides support for current application enablement technologies, but what sets z/OS apart is the ability to operate both new and existing applications within the same system, and in close proximity to the corporate data residing on z/OS. WebSphere® applications can run on the same z/OS system as the DB2® database, which can enable tight, security-rich local connections ideal for high-volume transactional throughput. Current CICS or IMS™ transactions can be extended with these new technologies to deliver value in new and innovative ways, without incurring the substantial cost required to replace current core assets.

System z and z/OS are ideally suited to perform as a data serving hub for the enterprise. The platform's classic strengths of availability, security, reliability, scalability, and management have made the mainframe the de facto gold standard for data serving and OLTP. Market requirements for increased security and simplified data management, and the increasing need for real-time Business Intelligence make consolidating more data onto the mainframe an attractive option for many enterprises. New technologies, such as XML, represent net data serving workloads on the platform.

Application development

Application development items are intended to be used by programmers to write z/OS-based applications. These items either make application development easier or add capabilities for new and extended applications on z/OS.

1.1.5 Theme: Security

The need for strong platform security is a core attribute and one of the basic values of the z/OS platform. The mainframe is an ideal security hub for the enterprise. The IBM commitment to z/OS System Integrity coupled with the latest security enhancements can help your business protect users, applications, and data, which can ultimately help manage risk and meet compliance guidelines. Centralized definition, application, maintenance, and management of security policies help simplify security infrastructures as well.

Network security

The component updates here provide network accessibility to z/OS security services, leveraging the core security and crypto components of the platform in an enterprise-wide role, thereby amplifying the value of the z/OS core infrastructure. Components such as z/OS PKI services have been maturing to the point where significant value and savings can be realized, helping improve overall TCO and cement z/OS as a core part of IT in these enterprises.

This chapter describes the functional enhancements in z/OS V1R11 for all the components that are changed and enhanced with this release.

1.2 Health checks

The Health Checker provides a mechanism to check the health of an active system and can help you check for and proactively resolve potential problems before application availability is impacted or system or sysplex-wide outages occur. The health check framework allows for unique autonomic functions for the z/OS platform by alerting you about potential problems and allowing their correction before they impact your business.

The new health checks are designed to help check the health of active systems. Use IBM-provided checks to implement the latest IBM hints, tips, and best practices, or you can write your own checks to address application-specific needs. The output from these checks is useful for proactively resolving problems before they occur and can be used as additional documentation for your own audit and compliance needs.

The following checks are added in z/OS V1R11.

Service aids health checks

A health check for dump analysis and elimination (DAE) is designed to suppress duplicate dumps, to help prevent the system from using unneeded resources capturing diagnostic data repeatedly for the same problems. This SHARE(DSN) option will suppress a duplicate of a previous dump on any system in the sysplex. These new checks are intended to help prevent unnecessary system programmer time and system resources from being spent on duplicate dumps. The new health checks are designed to report on your DAE configuration, as follows:

- This check detects whether DAE is active and your settings match those indicated by IBM.
- This check detects that the DAE data set is not being shared in a Parallel Sysplex environment.

AutoIPL health check

AutoIPL support is designed to automatically take a standalone dump, or IPL, or both when the system is about to enter a number of disabled nonrestartable wait states. These checks are intended to help you validate your AutoIPL configurations. The new checks are introduced for AutoIPL as follows:

- This check detects that the hardware configuration supports AutoIPL and you have not established an appropriate AutoIPL policy, or that it is running on a System z9® server, where the hardware feature is optional and available at no cost but not installed. This includes checking for a GDPS environment, because AutoIPL is not intended to be enabled at the same time as GDPS.
- This check is designed to validate the devices specified in the DIAGxx for SADMP and MVS AutoIPL policies. Devices are checked to make sure they are available, are DASD, and that they are not defined as secondary devices in Metro Mirror (PPRC) pairs.

These checks are intended to help you validate your AutoIPL configurations. They are for SADMP and MVS AutoIPL policies.

DFSMS catalog health check

The catalog component of DFSMS now implements a health check to detect IMBED and REPLICATE attributes for VSAM data sets and catalogs. No supported release of z/OS honors the IMBED and REPLICATE attributes for new data sets and catalogs. In fact, these attributes can waste DASD space and often degrade performance, so avoid using the IMBED and REPLICATE attributes. This health check will notify you about any user and master catalogs in your environment having IMBED or REPLICATE attributes.

Integrated Security Server - LDAP server

This health check is designed to notify clients who are still using the ISS LDAP server of its pending removal in V1.11 and the requirement to migrate to IBM Tivoli Directory Server (IBM TDS, LDAP) for z/OS.

1.3 Migration health checks

This is a set of migration health checks that are designed to reduce the effort required to migrate to new levels of z/OS from prior levels.

SDSF SAF security migration check

Using an external security manager such as RACF for SDSF, through the system authorization facility (SAF), offers several benefits over using ISFPARMS-based security, including the ability to change security profiles dynamically, a single repository for security information, auditability, and more-granular protection. In addition, SAF-based security is the only way to protect certain functions, including the MQSeries® use of queues and all JES3 SDSF functions. For all SDSF functions, use SAF-based security. This check is designed to determine whether the SDSF class is active, and is available for z/OS R9 and z/OS R10 with the PTFs for APAR PK68253.

R11 zFS sysplex migration checks

A new sysplex_admin_level is added in z/OS V1R11, sysplex_admin_level=2. To use zFS file systems in a shared environment when one or more systems are at the z/OS V1R11 level requires that the new sysplex_admin_level be used on all sharing systems. The toleration support requires the PTF for APAR OA25026 to be installed on supported sharing systems running z/OS V1R9 and z/OS V1R10. Also, IBM has announced the eventual discontinuation of support for multi-file system aggregates. In z/OS V1R11, two new migration health checks are added for zFS, as follows:

- Verify that zFS is running at sysplex_admin_level=2 for zFS V1R11 toleration support, and is intended to be run on z/OS V1R9 and z/OS V1R10 systems.
- Verify whether zFS multi-file system aggregates are being used, and are intended to be run on all systems, regardless of z/OS level. These checks are included in the PTFs for APAR OA27198.

BCP Service Time Protocol migration check

The Server Time Protocol (STP) feature is designed to allow multiple servers and Coupling Facilities to maintain time synchronization with each other, without requiring a Sysplex Timer®. A migration health check will be done to verify that the current system is running with the STP in use, when appropriate for the system on which z/OS is running. Use Server Time

Protocol because the Sysplex Timer (9037-002) has been withdrawn from marketing and STP is planned to be its replacement.

BCP RTM migration check

The static resource manager health check detects entries listed in the static resource manager list, which is kept in CSECT IEAVTRML of load module IGC0001C. Static resource managers are called every time any task or address space terminates, but they are rarely applicable to very many of them. Because task and address space termination can happen thousands or even millions of times per day on each of your systems, unnecessary calls to static resource managers can consume significant CPU time. This check helps you identify obsolete entries in IEAVTRML for IBM products, and any other entries to be aware of in case they can be eliminated on your systems. This can help you reduce unnecessary resource consumption by static resource managers.

IBM Health Checker for z/OS RFC4301 compliance (RBAPPRFC)

z/OS Communications Server provides a new migration health check in this release, which helps you determine whether IPSec filter rules that do not comply with RFC4301 are active on your systems and provides guidance about the migration procedures and options available to migrate noncompliant IPSec filter rules to become compliant with RFC4301. IBM has announced in a statement of direction that support of IPSec filter rules that are not compliant with RFC4301 will be removed in future z/OS releases. This check is intended to provide an early warning about possible migration actions.

IBM Health Checker for z/OS DNS server check (RBAPPDNS)

The z/OS Communications Server is providing a new migration health check in this release, which helps users determine whether they are using the BIND9 DNS server on their system. IBM has previously indicated, in statements of direction, that support of DNS server functions on z/OS will be removed in a future z/OS release.

IBM Health Checker for DFSMSrmm

The DFSMSrmm migration checks to verify that a supported vital records selection option is in use, to identify whether the new GDG options must be used, and to verify if any dropped REXX stem variables are being used. This support is available for z/OS V1R9 and z/OS V1R10 with the PTFs for APAR OA26947.

Health check for system status detection (SSD)

In this release, a new health check is added to check whether the system status detection (SSD) partitioning protocol is enabled to ensure that failed systems are removed from the sysplex expeditiously and with a minimum of operator involvement.

Build z/OS version root migration health check

Because of release enhancements and service, the size of the z/OS root file system (or “version root file system”) continues to grow from release to release. As of z/OS V1R10, the size of the z/OS root file system, whether HFS or zFS, was approximately 3100 cylinders on a 3390 DASD. This is close to the limit of 3339 cylinders on a 3390-3 device, and if not accommodated can halt the installation.

In z/OS V1R11, a new check is added to examine the volume that the version root file system resides on, and assess whether that volume has an acceptable amount of available cylinders (a value greater than a default of 500 free cylinders, but it is customizable).

If the version root file system data set is SMS-managed, this check is not applicable.

Dynamic Channel Path Management

I/O configuration management comprises a complex set of tasks. Configurations must be defined with various factors in mind, including single point of failure avoidance and expected bandwidth needs. After a configuration has been defined, its I/O bandwidth requirements often change. This means you must monitor the system and respond to changes to ensure that your workloads meet their response time goals.

FICON DCM

Dynamic channel path management (DCM) was introduced for ESCON® channels with z/OS Release 1 on z900 servers. DCM is intended to help simplify I/O configuration definition, maximize the utilization of installed hardware, and enhance RAS. In z/OS V1R11, IBM extends DCM to manage FICON® channels in addition to ESCON.

1.4 CIM Server upgrade to OpenPegasus 2.8

The Common Information Model (CIM) is a standard data model developed by a consortium of major hardware and software vendors (including IBM) called the Distributed Management Task Force (DMTF) as part of the Web Based Enterprise Management (WBEM) initiative. WBEM includes a set of standards and technologies that provide management solutions for a distributed network environment. Interoperability is a major focus of WBEM, and using WBEM technologies can help you develop a single set of management applications for a diverse set of resources. With support for the CIM server on systems running z/OS, your applications can gain access to z/OS resources using an extensible, industry-standard model.

z/OS V1R11 CIM is upgraded to support the latest version of the OpenPegasus CIM Server and DMTF CIM Schema. This adds support of the “CIM Operations over HTTP” specification version 1.3 from the DMTF. At the time of development, this was the most current level of the CIM/WBEM specifications for the DMTF and CIM Server Implementation from TheOpenGroup.

CIM client for Java upgrade

The CIM client for Java Version 2 supports a new “CIM Operations over HTTP” specification version 1.3 from DMTF.

1.5 ISPF enhancements

Following are ease-of-use enhancements for ISPF:

- An extension to the ISPF Editor COMPARE command to allow the data set name of a SYSIN data set to be specified as part of the command syntax. An additional option on the command will allow you to specify that a pop-up panel be displayed so that a full-length data set name can be specified for a data set containing the SuperC process statements for the COMPARE.
- A new option, Prefix Dsname Level, on the Data Set List Utility entry panel which, when specified, allows the Dsname level to be specified with or without quotes. When the quotes are omitted, the TSO prefix (if running with PREFIX ON) will be prefixed as the first qualifier of the Dsname level. When the option is not selected, the behavior of the entry panel remains unchanged. By default, the option is not selected.
- Support for setting and displaying the EATTR attribute for data sets that can reside on extended address volumes (EAV).

- Two new ISPF Editor line commands, HX and HXX, to display individual records in a data set in hexadecimal.
- Support for extended member statistics that allows ISPF to store number of lines greater than 65,535. More space is taken up in the PDS directory for each member reset if this option is chosen.
- Support for a panel source statement input exit that enables dynamic modification of an ISPF panel when the panel is displayed. The exit is passed each panel source record as it is read by ISPF and is able to change, insert, or delete panel source records.
- A configuration option to include an additional qualifier in the names of data sets created by utilities such as the SuperC and Search-For.
- The z/OS UNIX Directory List Utility will be enhanced with new functions to display and update the following attributes for files and directories:
  - Owning user
  - Owning group
  - File format and tag information
  - User auditing options
  - Auditor auditing options

A new function will also be provided to display information about the file system for a file or directory.

1.6 RMF enhancements

RMF reporting enhancements for group capacity

In z/OS V1R8 and later releases, you can define capacity groups for LPARs to manage the 4-hour rolling average utilization of your servers and allow capacity unused by one LPAR to be used to satisfy workload demands on others. The RMF Postprocessor Group Capacity report now includes information about the available capacity in each capacity group and the percentage of time each partition was actually capped during the reporting period. This is in addition to existing fields that report the capacity of each group and whether each partition was eligible to be capped during the same period. This new support is intended to make it easier to manage LPAR capacity groups.

RMF overview reporting enhancements

In z/OS V1R11, RMF introduces new Overview reports and support in the RMF Spreadsheet Reporter. The Spreadsheet Reporter provides a fast way to display postprocessor data graphically using a spreadsheet application such as Lotus® 1-2-3® or Microsoft® Excel. These new extensions completely replace the postprocessor PLOTS control statement, which provided a character-based graphic view of system performance indicators.

RMF postprocessor XML support

RMF has enhanced the postprocessor to produce reports in XML format in addition to the current text output with a width of 132 characters. The new self-describing XML format of the RMF postprocessor reports allows you to display the reports in a form that is more state-of-the-art. For example, combining the XML with a style sheet allows you to display the report in a Web browser. The access to the RMF postprocessor data is planned to be simplified, so that application programs can use the XML format for a better parsing of the postprocessor reports.

RMF use of SMF 23 fields

RSM APAR OA22414 added new statistics to the SMF23 record. With z/OS 1R11, RMF provides these system statistics in the SMF70 and SMF71 records, for use in capacity planning tools.

1.7 Communications Server

A new z/OS Communications Server function simplifies the deployment of network policies. For example, now applying network security for IPSec, IDS, and AT-TLS is made easier with step-by-step tasks complete with setting up RACF security to defining other policy-related configurations.

AT-TLS and IPSec improvements

The Configuration Assistant for z/OS Communications Server is enhanced to provide a simplified dialog for definition of AT-TLS and IPSec policies.

Add ACF/TAP to z/OS Communications Server

In z/OS V1R11, the Advanced Communications Facility Trace Analysis Program (ACF/TAP) is made a part of the z/OS Communications Server element. ACF/TAP provides functions to format trace information, including VTAM® buffer traces and VTAM internal traces. ACF/TAP will also continue to be included in ACF System Support Program (ACF/SSP). This change in ACF/TAP packaging is intended to help you reduce cost and ordering complexity if you do not use the Advanced Communications Facility Network Control Program (ACF/NCP).

Policy infrastructure simplification

The Configuration Assistant for z/OS Communications Server is enhanced to simplify the installation and setup of networking policies. In addition to creating and maintaining policy definition files, the Configuration Assistant is planned to lead administrators through all tasks needed to get z/OS Communications Server policy functions such as IPSec or AT-TLS active and running on the z/OS system. These tasks include everything from setting up RACF security to configuring the Policy Agent and other daemons that might be involved in the function.

This support is intended to simplify deployment of the full networking policy environment and provide improved integration of the various z/OS communications.

New AT-TLS options

The Configuration Assistant for z/OS Communications Server is enhanced to support new system TLS and SSL options when defining AT-TLS policies. This support is designed to allow all system SSL options including FIPS-140-2 compliance, and to allow you to specify new AT-TLS settings that support TLS V1.1 and the message authentication method to be used for AT-TLS.

TCP/IP support for system z10 hardware instrumentation

Many installations use products or functions that employ sampling techniques to help them determine which programs are in use and provide a view of the components of application response time and each component's contribution to response time. z/OS Communications Server code will be changed to make it easier for such products and functions to map load modules and entry points. This is intended to make it easier for you to gain a better view of how the network-related components of your workloads contribute to application response time.

Policy infrastructure management

In z/OS V1R11, z/OS Communications Server design is changed to enhance the networking Policy Agent to provide monitoring, automatic start, and restart for the Defense Manager daemon, Internet Key Exchange daemon, Network Security Server daemon, syslog daemon, and Traffic Regulation Management daemon. This function is similar to the AUTOLOG function in the TCP/IP stack, except that applications will not need to maintain a listening socket. It is intended to provide simpler management and operations for a set of applications that are associated with the policy infrastructure, and to reduce the complexity of enabling and operating the z/OS Communications Server policy infrastructure by providing a single point of control and monitoring for its most widely-used components.

Syslogd enhancements

In z/OS V1R11, the z/OS Communications Server syslog daemon (syslogd) design is changed to make it a multi-threaded application. This is intended to speed processing of z/OS UNIX System Services log messages. Also, this design allows the syslogd job name to match the cataloged procedure name and provide a set of operator commands for starting, stopping, and controlling syslogd. Syslogd is also enhanced to archive z/OS UNIX files to sequential data sets automatically, based on configurable options. These changes are intended to improve processing performance and reliability for z/OS UNIX log messages and to simplify syslogd file space and archive management operations.

Syslogd browser and search facility

In z/OS V1R11, z/OS Communications Server provides a new ISPF-based syslog daemon browser application to support browsing and searching both currently active syslogd files (those currently being written to by the syslogd daemon) and syslogd archive data sets that were created using a new integrated syslogd archive function. This support is intended to provide easier access to the information logged by syslogd.

Improved responsiveness to storage shortage conditions

z/OS Communications Server adds improved internal monitoring and management of storage, which is used by sockets applications that for various reasons build up excessive amount of data on their send or receive queues. In addition, it is planned to add support in z/OS Communications Server to allow the z/OS routing daemon (OMPROUTE) to continue processing during temporary storage shortage conditions.

This change is intended to improve network and system availability during periods when network traffic volume peaks and storage is constrained.

Disable moving DVIPA as source IP address

The TCPSTACKSOURCEVIPA function is enhanced in z/OS V1R11 Communications Server to prevent the stack from using dynamic virtual IP addresses (DVIPAs) in MOVING state as source IP addresses. New TCP outbound connections that are established after the DVIPA has transitioned to the MOVING state no longer use the DVIPA as a source IP address. This enables DVIPAs in the MOVING state to return to their original state in a timely manner.

Sysplex autonomics improvements for FRCA

New function in z/OS Communications Server improves the accuracy of workload balancing decisions for server applications that use the fast response cache accelerator (FRCA) function with persistent HTTP connections, such as Web serving applications.

accept_and_receive API enhancements

The accept_and_recv socket API allows z/OS applications and middleware to process short-lived client TCP connections more efficiently by combining the functionality of multiple

socket calls into one. This support improves performance for TCP server applications that exploit the new combined sockets API.

TCP throughput improvements for high-latency networks

z/OS Communications Server design is changed in z/OS V1R11 to help improve performance for inbound streaming TCP connections over networks with large bandwidth and high latency by automatically tuning the ideal window size for such TCP connections. This support is intended to improve throughput for large file transfers over long distances.

Sysplex Distributor connection routing acceleration

Performance of z/OS Communications Server Sysplex Distributor when forwarding inbound IP packets for distributed connections is enhanced through a new Sysplex Distributor connection routing accelerator function. The Sysplex Distributor connection routing accelerator is based on a new general QDIO routing accelerator function that also is part of z/OS V1R11. This is expected to reduce the CPU time and latency for Sysplex Distributor connection routing.

Resolver DNS cache

z/OS Communications Server provides enhancement to the resolver function by introducing system-wide caching of domain name server (DNS) responses. The resolver cache will be enabled by default and shared across the entire z/OS system image.This new function will be expected to provide significant performance improvements for z/OS workloads that perform repetitive resolver queries, such as Web services workloads. These performance improvements are expected to be realized without the need to set up, monitor, and administer a name server for caching purposes on your system.

NSS private key and certificate services for XML appliances

z/OS Communications Server enhances network security services (NSS) to further enable XML appliance security as a logical extension of z/OS security. Two new services are intended to enhance z/OS centralized management of security for XML appliances by adding:

- A certificate service to provide certificate management operations
- A private key service to allow retrieval of private keys from RACF certificates, and to perform RSA signature creation and RSA decryption using ICSF-protected keys

HPR performance enhancements

SNA High Performance Routing (HPR) performance is enhanced by implementing a new progressive mode Adaptive Rate-Based (ARB) pacing algorithm, which increases HPR performance in virtualized or CPU-constrained environments. In addition, unproductive path switches when an HPR endpoint is unresponsive, is reduced or eliminated. The intent of this new function is to improve SNA Enterprise Extender (EE) performance when EE partners are located on virtualized distributed platforms.

EE IPSec performance improvements

z/OS Communications Server improves performance in several areas to accommodate enhancements in IP security for Enterprise Extender (EE) connections. This new function helps reduce the cost of securing EE traffic with IPsec security.

Sysplex Distributor optimization for multi-tier z/OS workloads

Prior to z/OS V1R11, you can configure Sysplex Distributor with OPTLOCAL to optimize connections when both connection endpoints potentially reside on the same TCP/IP stack within the z/OS sysplex. This feature can be very useful in environments where multi-tier

server applications are co-located within the same z/OS system images in a single sysplex. In this type of environment OPTLOCAL is deployed on the tier 2 Sysplex Distributor to optimize the connections between tier 1 and tier 2 server applications.

z/OS V1R11 allows further optimization if Sysplex Distributor is also being used as the load balancer for the tier 1 server applications. This optimization allows the tier 1 Sysplex Distributor to have visibility into both tiers of the z/OS server applications on a given system when making a load balancing decision on an incoming tier 1 connection request. When using WLM-based recommendations, this optimization allows Sysplex Distributor to compute a composite WLM weight for each system that includes the capacity, performance, and health characteristics of both the tier 1 and tier 2 server applications.

This new function further improves the quality of the load balancing decisions made by Sysplex Distributor in a multi-tier z/OS server environment.

QDIO routing accelerator

The QDIO accelerator function provides accelerated forwarding at the data link control (DLC) layer for IP packets coming inbound over HiperSockets™ and being forwarded outbound over OSA-Express QDIO or HiperSockets and for packets coming inbound over OSA-Express QDIO and being forwarded outbound over OSA-Express QDIO or HiperSockets.

Note: The QDIO accelerator function is intended to improve performance when IP packets are routed through z/OS using any of the supported network interfaces.

TCP/IP path length improvements

The TCP transport layer in z/OS Communications Server TCP/IP is updated to detect and transparently react to two common sockets programming errors that can cause severe performance problems. z/OS Communications Server is now designed to transparently guard against traffic stalls due to use of the Nagle algorithm and delayed TCP acknowledgements and stalls caused by TCP receive buffers that are too small. This new function helps to improve application availability and performance.

IPv6 stateless address autoconfiguration enhancements

In z/OS V1R11, z/OS Communications Server is now designed to allow a client application to use IPv6 temporary auto-configured addresses generated from a random interface ID to help address security and privacy concerns identified by RFC 4941 “Privacy Extensions for Stateless Address Auto-configuration in IPv6.” The use of temporary addresses with random, changing interface IDs embedded in the address is intended to make it more difficult for eavesdropping software to correlate independent transactions using separate IPv6 auto-configured addresses but involving the same z/OS system.

New API to obtain IPv4 network interface MTU

In z/OS V1R11, z/OS Communications Server adds support for a new ioctl programming interface. This new interface is intended to allow applications to determine the MTU (maximum transmission unit) for a TCP/IP stack IPv4 interface. This new support is in addition to similar existing support for IPv6 interfaces. It is intended to enhance sockets application portability with other platforms.

RFC 5095 deprecation of IPv6 Type 0 Routing Header

z/OS Communications Server is changed to deprecate support for IPv6 Type 0 Routing Headers. This new design is intended to comply with IPv6 RFC5095, “Deprecation of Type 0 Routing Headers in IPv6.”

New SMTP client for sending Internet mail

z/OS V1R11 provides a new z/OS Communications Server mail client for sending mail from z/OS to a mail server. The new mail client is designed to support the existing user interface for sending mail from z/OS that was defined by the old SMTP/NJE mail gateway server, and many new mail enhancements such as secure connections, the ESMTP protocol, and IPv6 connections. The new z/OS Communications Server mail client is designed to support sending mail only; it does not support receiving mail from the Internet. This support is designed to provide secure connections to a mail server; provide simplified setup and operation; use fewer resources than the existing SMTP/NJE mail gateway server when sending mail from z/OS; and support existing applications that write mail messages through the JES spool interface.

Removal of NDB, DHCP server, BINL, and BIND 4.9.3

In z/OS V1R11, Communications Server no longer supports the following:

- Boot Information Negotiation Layer (BINL) server function
- Berkeley Internet Name Domain 4.9.3 (BIND 4.9.3) DNS server function, including DNS/WLM Connection Optimization
- Dynamic Host Configuration Protocol (DHCP) server function
- Network Database (NDB) server function

1.7.1 Workload management in a network environment

z/OS has 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 and still meet your application and business priorities.

QDIO enhancements for WLM I/Opriority

In z/OS V1R11, z/OS Communications Server provides a function to map WLM service class importance levels to outbound QDIO priorities in installations where outbound QDIO priority is not already being set through network Quality of Service (QoS) policies.

Taking advantage of already established importance levels is expected to help ease the extension of this prioritization through z/OS and z/OS Communications Server, using OSA-Express and onto the LAN.

QDIO support for OSA interface isolation

OSA-Express connection isolation is now designed to provide the ability to force all packets to be written to and received from a LAN even if direct routing between TCP/IP stacks sharing the OSA-Express port is possible. This support is intended to provide an improved ability to enforce IP filter processing in outboard IP routers for traffic that flows between System z LPARs that share OSA-Express ports.

Network management interface enhancements

The z/OS Communications Server SNA Network Management Interface (NMI) provides additional storage ownership statistics in response to the communications storage manager (CSM) statistics request. Network management applications using this NMI can request all ownership statistics or a subset of statistics based on which address space owns the storage. This support provides improved problem determination based on a more granular network management insight into which address spaces use CSM storage.

Extend number of reporting classes to 2048

With systems and environments becoming larger, more reporting classes are needed in WLM. In z/OS V1RA11, the number of report classes is increased from 999 to 2,048. This increase is expected to allow more fine-grained reporting of your workloads.

Make REQLPDAT callable from unauthorized users

In z/OS V1R11, the service used to request data about a logical partition, REQLPDAT, which can return LPAR and capacity information from the system, is enhanced to support unauthorized applications. This is expected to make it easier to obtain information about the system.

Support for enhanced WLM routing algorithms

The server-specific WLM recommendations that are used by the Sysplex Distributor to balance workload across systems when the SERVERWLM distribution method is chosen are enhanced to allow you to specify how zIIP and zAAP specialty processor capacity influences selection of eligible target systems. In addition, new configuration controls allow you to specify that WLM consider the importance of displaceable capacity when determining server-specific recommendations. These enhancements improve the workload balancing functions provided by WLM and Sysplex Distributor.

1.8 New support with z/OS V1R11 BCP

SDSF syslog support and elimination of HASPINDX

SDSF added support for JES3 in z/OS V1R10. In z/OS V1R11, SDSF adds new function for JES3 environments with new panels for syslog, job classes, JESPlex information, spool volumes, and data set output descriptors. Also, JES2 users of SDSF no longer need to use the SDSF HASPINDX data set after all systems are at the z/OS R11 JES2 level.

APPC (BPXAS) job filtering through extended status API - SDSF

SDSF, together with JES2 and JES3, will provide enhanced transaction filtering on Extended Status and SAPI SSIs. The JES extended status and SYSOUT API SSIs will be enhanced to support selection of SYSOUT based on transaction job names and IDs. Transaction job names and IDs are created by APPC transactions, UNIX transactions that run in BPXAS address spaces, and various applications that associate transaction information with a SYSOUT data set. For the extended status SSI, this can provide performance improvement by reducing the amount of I/O needed to obtain this information. For the SYSOUT API this provides new functionality.

For JES2 to implement this support, additional data will be placed in the JES2 checkpoint, which requires a new $ACTIVATE level. The extended status and SAPI SSI enhancements are available when all members of the JESPLEX are running z/OS V1R11 JES2 and a $ACTIVATE LEVEL=Z11 has been done.

Message Flood Automation to not use IEAVMXIT

The Message Flood Automation function introduced with z/OS V1R9 has been changed to eliminate its use of the IEAVMXIT user exit. Message Flood Automation message processing is now done as part of internal message processing. Also, Message Flood Automation processing no longer uses the System Command Exit. These changes eliminate the customization and user exit integration steps required to use Message Flood Automation and allow Message Flood Automation to take action against all messages, even those sent

through specific MPF exits. Making it easier to use Message Flood Automation is intended to help you avoid system and multisystem outages due to message flooding.

Set and display allocation parameters

Two new commands can help improve system availability by allowing you to change allocation settings without an IPL in z/OS V1R11. The two new commands are:

- The SETALLOC command is designed to allow you to change all the settings specified in ALLOCxx parmlib members with the exception of 2DGT_EXPDT, which specifies how 2-digit years in expiration dates are to be processed. The command uses keywords that are similar to those you specify in an ALLOCxx parmlib member.
- The DISPLAY ALLOC command is designed to show the currently active allocation settings as set from the ALLOCxx parmlib members used during IPL and any subsequent SETALLOC commands. Also, this command allows you to display allocation information that can help IBM Service diagnose problems.

Resource recovery services (RRS)

RRS compression on RM.DATA is changed to remove the every 30 second compress of the RM.DATA log stream. In the past, this compression was done to remove old, duplicate entries which ensured only the latest states of its resource managers. Instead, RRS will now tolerate the duplicate entries so for a specific resource manager, the only entry of interest to RRS is the latest log entry. This change is intended to improve RRS performance due to fewer compresses. In addition, RRS will take actions to bring the RM.DATA log stream into the coupling facility sooner during initialization. This change is intended to allow for faster access to the log stream and much less enqueuing, with the intent of improving performance.

RRS local or GMT time display on the log browser

This new support allows you to specify whether the timestamps displayed by the log browser are in local time or GMT. A new global option is used to specify this setting. This is intended to make the dialog more usable.

Binder IEWPARMS DD statement

In z/OS V1R11, the program management binder adds support for a new IEWPARMS DD statement. IEWPARMS is used to specify a data set (which can be an inline or SYSIN data set) that includes the parameters to be passed to the binder. These parameters are processed in addition to any parameters specified another way (for example, by using the PARM keyword of the JCL EXEC statement). This new support allows options to be added to those passed by an existing program without requiring program changes, and for more than 100 characters of parameter data to be specified in JCL. Also, it is expected to help simplify debugging for programmers developing applications that call the binder API.

Tape load balancing

Device allocation design is changed in z/OS V1R11 to improve its tape load balancing algorithm. This new support will consider the number of devices in each tape library, and how many are online, when allocating tape devices within libraries. For example, a library with 256 online devices is preferred over a library with 16 devices. You can control whether this new algorithm is used with the new SYSTEM parameter for the ALLOCxx parmlib member. This is designed to provide better balancing of tape library requests across the system.

TSO/E LOGONHERE support

TSO/E and z/OS Communications Server now support a new option that enhances the LOGON RECONNECT function of TSO/E when using TN3270 connections. A new LOGONHERE parameter in the IKJTSOxx parmlib member controls whether the

RECONNECT option can be used to re-establish a TSO/E session unconditionally. Without this support, an attempt to reconnect is rejected if the system detects that the user ID is already in use, either on the same workstation or another one. This new feature is intended to help preserve work that might be in progress under TSO/E sessions and avoid operator intervention.

1.8.1 HCD and HCM

HCD/HCM multi user access

This function enables multiple HCM users to access and work with the same configuration at the same time. It is based on the multi-user option of an IODF, thus allowing multiple users sharing the same IODF and HCM configuration file at the same time, even for update. This function is intended to increase client productivity by allowing multiple users to work on the same configuration at the same time.

HCD and HCM enhancements

In z/OS V1R11, a number of display and report enhancements are available for HCD and HCM. These enhancements are intended to improve usability.

HCD edit profile dialog

In z/OS V1R11, a new dialog interface to the HCD profile is introduced that allows you to set and modify standard HCD profile options and also to edit policies. Rather than editing the profile data set directly, you will have prompt and help assist for the profile options available. Additionally, the dialog help is intended to assist in avoiding syntax errors.

HCM support for Windows Vista and HCM/HCD support for IpV6

In z/OS V1R11, HCM is updated to support Windows® Vista. Also, communications between HCD and HCM will support IPv6.

1.8.2 BCP failure analysis

Dynamic LPA for HFS and MAPMVS

This support is to enhance dynamic LPA to be able to take the full path name and file name when the module is passed ByAddr. This saves the path name as part of the information associated with the entity so that MapMVS can get at it when it uses CSVINFO or CSVQUERY. Also, for ByAddr cases, this allows the user to provide the diagnostic data needed to identify the data set (UCB address and CCHH) when the load came from a PDS(E). This support helps to determine the source for an LPA module that originally came from a HFS.

Predictive failure checks

Predictive Failure Analysis is a new technology designed to provide early warning about emerging problems before they impact applications. Introduced in 2009, the first application of this technology is intended to detect anomalous patterns of common storage utilization. In z/OS V1R11, Predictive Failure Analysis is applied to these additional areas:

- Message rate
- The amount of virtual storage used by long-running address spaces such as started tasks.

These checks are expected to help you achieve high availability by providing earlier warning of unusual conditions.

1.8.3 Allocation changes

Eligible device table size reduction

In z/OS V1R11, allocation design is changed to significantly reduce the amount of storage required by the eligible device table (EDT) for many I/O configurations. This new function is intended to provide common area virtual storage constraint relief, both for the EDT itself and for the use of dynamic activation for I/O configuration.

1.8.4 Recovery items

Mean time to recovery (MTTR)

MTTR is interpreted as meaning the time from when a problem is detected until the system is back up doing productive work. It encompasses an orderly shutdown (if possible), the gathering of diagnostic information (if necessary) and the restarting of the system, from the initial loading of the operating system (IPL, NIP and Master Scheduler Initialization), through the bringing up of the major subsystems and middleware (including Communications Server, JES, IMS, CICS, DB2, MQSeries, and WebSphere) and the starting of the major applications on those subsystems and middleware.

The immediate intent of the MTTR initiative is to:

- Reduce the time that it takes the operating system itself to come up by reducing operating system initialization path lengths where possible and by exploiting parallelism where possible.
- Reduce the cost of operating system services used by subsystems and middleware to initialize themselves.
- Work with the subsystems and middleware to reduce the time that it takes them to initialize by reducing path lengths where possible and exploiting parallelism where possible.

1.8.5 Miscellaneous items

IEFBR14 to DELETE without recall

The IEFBR14 program that is delivered with z/OS is often used in JCL to allocate and delete data sets. In z/OS V1R11, when IEFBR14 is used to delete a data set that has been migrated by DFSMShsm, the delete request is passed to DFSMShsm, which will delete the data set without recalling it. This is expected to take significantly less time and consume fewer system resources than waiting for the data set to be recalled before being deleted, thereby helping speed batch processing when a migrated data set is deleted using IEFBR14.

HWISET support

In z/OS V1R11, BCPii is enhanced to allow an authorized z/OS application the ability to change or set data for certain HMC-managed objects associated with a CPC, Image, or Capacity Record. In this release, Capacity Provisioning Manager (CPM) requires the HWISET command to set various CPC attributes related to the capacity of the CPC. This support is intended to provide RAS and performance benefits, as well as a more complete suite of services available to the programmer.

Server Time Protocol (STP) alert

New support is provided by z/OS to recognize additional STP alert external interrupt events and to handle them as appropriate by issuing warning WTOs. These WTOs will warn clients of RAS events which they might need to get corrected.

SRB priority adjustment

Once the need for SRB priority adjustment has been detected, SRBs scheduled to the target address space will be given the dispatch priority of the scheduler (if that priority exceeds the “normal” priority that is otherwise assigned to the SRB). A limited number of SRBs will be given this priority. If the excessive length situation still exists, the same processing will occur again and an additional number will be given the priority adjustment.

This change is intended to help avoid a system outage due to continued SRB scheduling.

1.8.6 Sysplex enhancements

Sysplex partitioning enhancements using BCPii

In z/OS V1R11, XCF sysplex partitioning processing is designed to exploit BCPii functions to improve partitioning. XCF is designed to use the BCPii remote query and event monitoring capabilities to discover when another system is in a state in which XCF can immediately initiate sysplex partitioning without waiting for the failure detection interval (FDI). This is intended to improve time for partitioning by 1 to 2 minutes and avoid operator intervention.

Similarly, XCF is designed to use BCPii to avoid unnecessary cleanup delays. When system isolation fails, XCF is designed to use BCPii to perform a remote reset, again avoiding operator intervention. Also, when manual intervention is required, XCF is designed to verify the responses to messages issued during the partitioning process when possible using BCPii to make sure that systems to be partitioned out have been reset. Finally, XCF is designed to adjust the Failure Detection Interval based on the greater of a value you specify and the system’s spin loop timeout values. These enhancements are intended to help improve sysplex availability and reduce the potential for operator error.

XES locking hash table size increase

This support provided by XES avoids locking hash table scalability issues by increasing the size of the three hash tables (local, global, and message), so as to “spread out” the locking control blocks across a larger hash class “space.” This is intended to result in less hash class latch contention and also shorter queues that can be processed much more efficiently in XES processing. This item satisfies general requirements for z/OS and Parallel Sysplex scalability/constraint relief. As sysplex locking scales up, the need for improved hashing internal to the XES lock management processing scales with it.

Large number of subchannels support in XES

A new XES design is planned to provide algorithmic enhancements for Coupling Facility subchannel operation completion, recovery, and notification processing, and is intended to improve efficiency for processing a large number of Coupling Facility subchannels. Coupling Facility subchannel information provided by console DISPLAY commands is also planned to be changed to accommodate a large number of Coupling Facility subchannels. This new function is intended to support larger CF and CF link configurations that require more concurrent I/O capability.

1.8.7 SMF enhancements

SMF fields expansion

A number of service unit count fields in the performance section of SMF Type 30 records are incremented as an address space runs. To contain these counts for very long-running address spaces that can consume a larger number of service units than the original fields can represent, new larger fields are added to SMF Type 30 records. These new fields can be used

in place of the original fields to gather information about the service units consumed by long-running address spaces. Also, new fields have been added to the Storage and Paging section of SMF Type 30 records to better track the use of storage above 2 GB. These new fields include information about an address space's use of virtual, real, and auxiliary storage above 2 GB. These changes to SMF Type 30 records are intended to help with capacity planning, performance planning, and accounting.

SMF dump program

In z/OS V1R11, the SMF log stream dump utility, IFASMFDL, allows you to specify a range of dates relative to the date on which the program is started. For example, you can specify that the SMF records created yesterday, or those created for six days starting nine days ago, be processed. This design allows date ranges to be specified by days, weeks, or months. Also, IFASMFDL supports new options to manage the retention of the data in the SMF log stream and allow you to specify that data by archived (dumped and deleted) or deleted from the log stream. These new IFASMFDL functions are intended to make it easier to exploit log stream-based SMF data management.

1.8.8 GRS enhancements

In z/OS V1R11, the following GRS enhancements are provided.

GRS GQSCAN and ISGQUERY improvements

In z/OS V1R11, GRS design changes provide storage constraint relief for GRS GQSCAN and ISGQUERY star-mode global ENQ processing. Work buffers used by parallel requests are planned to be to 64-bit pools in the GRS address space. This is intended to increase the amount of data that can be processed by concurrent system-wide GQSCAN and ISGQUERY requests.

Enhanced contention analysis (ECA) for latches

Enhanced contention analysis (ECA) for GRS latches is designed to provide improved analysis for latch contention. A new latch identity service can be used by exploiters to provide information for display commands so that the holder of a latch is easier to identify. A new DISPLAY GRS,ANALYZE command, similar to the one available for enqueues, is available for latches. Latch identification information and better real time diagnostic data are planned to be displayed. In z/OS V1R11, System Logger and Resource Recovery Services (RRS) are planned to provide identification information about their latches. The new functions for analyzing latch waiters, blockers, and dependency contention analyses are intended to make it easier to identify deadlocks and to find the root cause of latch contention.

GRS contention notification exit

This support is designed to provide GRS ENQ contention notification (ENF 51) enhancements that will allow listeners to be notified for specific resources, including those for conditional requests that are not satisfied because resources are not available. The support allows ENQ monitors to report on unsatisfied conditional requests and for ENQ exploiters to be able to track and react to contention for their resources.

ENQ support

In z/OS V1R11, GRS is enhanced to provide additional dynamic exit support, which allows IBM and vendor products to gather ENQ contention information under an ENQ client's context. A new ISGNQXITQUEUED2 is provided at the control point where all local and global ENQs have been queued for processing under the caller's unit of work, and the existing

ISGCNFXITSYSTEM and ISGCNFXITSYSPLEX are modified to provide additional information. These exit enhancements add to the set of GRS OEM related exits.

1.8.9 64-bit architecture enhancements

In z/OS V1R11, support is added for data tables contained in load modules and program objects to be placed above 2 GB. A new ADDR64 parameter on the LOAD macro supports loading load modules and program objects containing only data tables above the bar. This function is intended to help relieve virtual storage constraints by allowing data tables to be loaded above the bar.

1.8.10 SYSREXX enhancements

Application development items are intended to be used by programmers to write z/OS-based applications. These items either make application development easier or add capabilities for new and extended applications on z/OS.

SYSREXX improvements

In z/OS V1R11, these new functions have been added to System REXX:

- REXXLIB concatenation: In addition to execs stored in the AXRREXX data set, System REXX will search a new REXXLIB concatenation for execs to be run.
- In addition to AXR00, parmlib members with other two-character suffixes are supported, and can be specified in IEASYSxx parmlib members. Also, multiple AXRxx members can be specified.
- System REXX is now designed to acquire JES affinity for AXRnn address spaces after the primary JES subsystem starts. Every TSO=YES AXREXX invocation will have a JES jobid and joblog associated with it. This is intended to allow the joblog to be browsed or parsed by REXX code. Also, additional capabilities related to JES spool might now be leveraged by REXX code running under System REXX, including the ability to write JCL to an internal reader, retrieve output, and use the TSO/E TRANSMIT, RECEIVE and SEND commands.
- Support for z/OS UNIX callable services (syscalls) for TSO=YES AXREXX requests. This is designed to allow you to take full advantage of the available syscall commands.
- Support for two new built-in functions: AXRINFO is designed to return the name of the subsystem under which the exec is running (JES2, JES3, or MSTR), and AXRWAIT is designed to provide a wait function within a REXX exec.
- Support for the TSO/E STORAGE function in read-only mode.

Note: These functions are also planned to be made available on z/OS V1R9 and z/OS V1R10 with the PTFs for APAR OA26802.

1.8.11 Dump management

The dump management roadmap is intended to keep pace with the growth in diagnostic data that results from larger systems and larger programs using ever-increasing amounts of memory. These improvements are meant to help you keep dumping time and dump transmission time under control.

Dump exit - JES3

JES3 now causes the system to dump additional address spaces when a failure occurs in an address space that is using certain JES3 services. The additional address spaces include JES3, JES3AUX, and JESXCF. This new function is intended to provide improved first failure data capture for other address spaces that use the subsystem interface (SSI) for JES3 services.

SDUMP resource management

SVC Dump processing uses virtual storage to contain a copy of the data being dumped until it can be written to a dump data set; this is intended to help minimize the time required to collect the data so that processing can resume quickly. The system prevents dump processing from consuming enough virtual storage to exhaust all available auxiliary storage. You can also specify a maximum amount of virtual storage using the MAXSPACE option of the CHGDUMP command to make it less likely that an auxiliary storage shortage will cause a system-wide impact during dump processing.

In z/OS V1R11, the AUXMGMT option is planned to be added to CHGDUMP. This allows you to specify that the system is to manage the amount of virtual storage used for dump processing based on available auxiliary storage. This is intended to allow the system to react to changing auxiliary storage capacity and utilization automatically and make it unnecessary to monitor and tune your specification for MAXSPACE.

SDUMP enhancements

As systems become capable of supporting larger amounts of storage, the exploitation of large real and virtual can increase the size of SVC dumps, and the time it takes to collect the data. For a portion of the dump processing time, other work on the system is suspended to prevent critical data from being changed before it can be copied. In z/OS V1R11, dump processing design has been changed to detect excessive periods of system-wide non-dispatchability and allow other work to resume if dump data collection is taking too much time. This is intended to reduce the impact of collecting diagnostic data.

Also, a new message, IEA045I, is issued at the start of dump processing to complement the existing IEA611I message issued when dump processing ends. This is designed to make it easier to find the duration of dump processing.

1.9 IBM z/OS Management Facility

IBM z/OS Management Facility (z/OSMF) is a separate no-charge product that will simplify, optimize and modernize the z/OS system programmer experience. z/OSMF will deliver solutions in a task-oriented, Web browser-based user interface with integrated user assistance, enabling both new and experienced system programmers to more easily manage the day-to-day operations and administration of the mainframe z/OS systems. The initial release will facilitate problem data management tasks for new and less skilled system programmers, as well as provide the experienced system programmers and administrators with procedural advantages through an event log summary and detail views of z/OS incidents. When a problem is encountered on a z/OS system today, the system programmer has to do the following:

- Take many manual, time-consuming steps to collect diagnostic data such as dumps.
- Examine appropriate excerpts from logs and sometimes send them to IBM.
- Get a consolidated list of the abend-related problems across a sysplex.

This new problem management task in the z/OS Management Facility provides simple interfaces to get a consolidated list of problems, as identified in SVC dumps, along with the properties and diagnostic data captured and saved with each abend dump. It also provides an easy-to-use interface to facilitate sending the diagnostic data for further diagnosis. This will require less time and skills to manage the problems and diagnostic data in a sysplex.

1.10 Language Environment

In z/OS V1R11, the following Language Environment enhancements are provided.

Language Environment diagnostics

In z/OS V1R11, a new function in the IPCS LEDATA verb exit is designed to provide a new high-level trace mechanism for C FILE I/O that can help reduce the time it takes to understand and debug complicated I/O problems. Also, the IPCS function, in combination with additional settings on the HEAPCHK option, is intended to improve formatting of heap pools, control blocks and extents to make it easier to find damaged cells.

Language Environment exploitation of large pages

In z/OS V1R11, Language Environment® now allows AMODE 64 applications to request large page-backed memory objects using the __moservices() interface. Appropriate use of large (1 MB) pages can help reduce memory management overhead and increase translation lookaside buffer (TLB) hit ratios for exploiting programs.

Language Environment AMODE 64 enhancement

In z/OS V1R11, Language Environment is enhanced to allow AMODE 64 applications to use the preinitialization routine (CELQPIPI) to specify that they will do their own storage management using GETSTORE and FREESTORE, and have their own message service routines (MSGRTN). This function is already available for AMODE 31 applications using CEEPIPI. Also, Language Environment will be designed to enhance existing support for IEEE Decimal Floating Point (DFP) for C/C++ programs.

Language Environment CEEFETCH

In z/OS V1R11, new support for Language Environment is intended to make it easier to write programs that can be ported between operating systems, such as z/OS and z/VSE™. This support includes:

- A CEEGLOB assembler macro, RMODE and AMODE support in the CEEENTRY assembler macro
- A new FTCHINFO keyword on the CEEFETCH assembler macro that allows for the load of both Language Environment and non-Language Environment modules, and SERVICE keyword support in CEEPPA assembler macro.

Swap context and make context

In z/OS V1R11, Language Environment is designed to provide a consistent implementation of the getcontext(), makecontext(), setcontext(), and swapcontext() functions between AMODE 31 and AMODE 64 applications. This is intended to allow applications to be common for AMODE 31 and AMODE 64 and make it easier to enable applications for 64-bit programming. This item satisfies FITS requirement MR1106061850.

Language Environment CICS AFP support

Current System z hardware and software support the IEEE standard for binary and decimal floating point arithmetic. However, CICS Transaction Server does not support application programs that make use of the additional floating point registers. The consequence of not supporting the additional floating point registers is that CICS cannot be sure of maintaining integrity for programs that use the non-volatile floating point registers. Programs can be any user application, IBM product code, or vendor product code. Problems have been reported in applications using XL C/C++ and Enterprise PL/I compilers because they support usage of the additional floating point registers as a default. The current solution for the compilers is to provide a new option AFP(VOLATILE) that specifies that all of the AFP registers are volatile. This option will cause the compilers to generate extra code to save and restore these registers, but at a cost to performance.

A service FLASH was created in 2005 explaining the problem and the action needed by applications to work around this problem. The Enterprise COBOL compiler has already implemented a similar solution as the default processing. CICS TS V4.1 is planned for 2009 and is being designed to include full support for the extended MVS linkage convention. This includes preserving the non-volatile floating point registers (FPRs), the floating point control (FPC) register, access registers (ARs), and the 64-bit general registers (GRs). Language Environment is also making changes in support of the CICS design so that IEEE binary and decimal floating point can be exploited from Language Environment-conforming languages in the CICS environment.

1.11 z/OS UNIX System Services

In z/OS V1R11, the following z/OS UNIX System Services enhancements are provided.

z/OS UNIX file system remount support

In z/OS V1R11, a new REMOUNT function is implemented for z/OS UNIX System Services. Supported on the ISHELL panel for working with mounted file systems, by new operands on the TSO/E **unmount**, z/OS UNIX CHMOUNT, and REXX **unmount** commands, and supported by callable service and an osi_service interface, this new function is designed to remount an already-mounted file system in the same mode. This is expected to help you recover from problems involving file systems that have been disabled for write.

zFS sysplex support

This support provides changes to the zFS XCF protocol used by zFS sysplex administration functions. Such changes will allow zFS to recover from failures dynamically, without operator intervention, thus improving zFS reliability and availability.

In z/OS V1R11, zFS design is changed for shared read/write file systems in a sysplex to send file requests directly to the physical file system (PFS) layer, avoiding the z/OS UNIX file system owner. This is intended to improve the performance of read/write zFS file systems shared across systems in a sysplex.

z/OS UNIX availability

z/OS UNIX has been changed to provide a new option that you can use to specify that all syscalls made during a kernel shutdown be failed for applications. Also, additional documentation and information will be provided for security system calls that end in error to help you diagnose security problems for z/OS UNIX applications.

z/OS UNIX syscall tracing

In z/OS V1R11, z/OS UNIX System Services allows you to capture system call (syscall) trace for a specific application or set of applications. This is intended to allow you to gather more information about program processing history to facilitate application debugging. You will be able to specify that syscall tracing be turned on or off using a new SIGTRACE signal and BPXTRACE command.

z/OS UNIX file system enhancements

In z/OS V1R11, improvements are made to z/OS UNIX file system processing design. z/OS UNIX automount processing design is intended to discontinue automated UNMOUNT attempts when unmount failures occur. A new BPXF251I message is issued when unowned file systems are recovered and made active in shared file system configurations. The DISPLAY OMVS,WAITERS command provides date and time information, and the DISPLAY OMVS,PFS display to include physical file system status information; the address space name for colony file systems including zFS, NFS and TFS; and the user, date, time, system, and master configuration file for the AUTOMOUNT command.

New ALTROOT parmlib statement

A new ALTROOT parmlib statement is added to specify an alternate root if the current root becomes unavailable (unowned), at which time z/OS UNIX System Services switch to the alternate root. This makes use of the NEWROOT function with modifications to the front end to obtain mountpoint information from the CDS MPT and TAB subrecords, because no operations can be done on the old root.

Demand debug load - dbx

In z/OS V1R11, the dbx debugger is designed to allow you to load a combined set of debug information (in a .mdbg file) to help improve dbx performance during startup and variable evaluation processing.

FTP access to z/OS UNIX named pipes

z/OS FTP support now transfers files to and from z/OS UNIX System Services named pipes. z/OS applications that support reading from named pipes can process data transferred into a named pipe while FTP is still writing data into that pipe. Likewise, FTP can initiate transfer of data written to a named pipe while an application is still writing to that pipe. This support is intended to help reduce overall latency when transferring data to and from z/OS systems in combination with pre- or post-processing by other z/OS applications or subsystems.

Alternate default UID support

This support automatically “on demand” generates unique UIDs and GIDs for users and groups that do not have an OMVS segment defined when a z/OS UNIX System Service is attempted. Such generation of unique IDs simplifies user identification and group administration, allowing users to be dubbed without explicit UID assignment or defaulting UID/GID, which can create accountability problems.

oedit and obrowse shell commands

In z/OS V1R11, z/OS UNIX System Services shell commands **`oedit`** and **`obrowse`** are changed to use ISPF Edit and Browse. This makes editing and browsing z/OS UNIX files work consistently in both environments and adds support for ASCII data when these commands are used from the shell.

euro currency new locale

In z/OS V1R11, new support is provided to align with the recent changes in currency and locale information. The new locale to be supported is Serbia.

1.12 System Logger

In z/OS V1R11, the following System Logger enhancements are provided.

IXGWRITE asynchronous processing management

In z/OS V1R11, System Logger design is changed to help protect the system from SQA shortages that can result from write requests for a log stream being issued faster than they can be processed. Thresholds are created for outstanding asynchronous IXGWRITE requests. When they are exceeded, requests are failed with a new return code to indicate to callers that too many outstanding requests for the specified log stream exist. An event notification facility (ENF) signal will indicate that the backlog has been reduced enough for new requests to be accepted. This change is intended to help improve system availability.

Restart time improvements

In z/OS V1R11, design changes are implemented for System Logger and the I/O Supervisor (IOS) to reduce IPL and restart time. The changes to System Logger are intended to reduce concurrent log stream connection time involving staging data set allocations, and IOS design is intended to streamline the channel programs used during initialization. These changes are intended to help improve IPL and Logger restart times.

1.13 DFSMS enhancements

In z/OS V1R11, the following DFSMS enhancements are provided.

DFSMS IDCAMS delete masking

The IDCAMS DELETE command can be used to delete multiple entries by using a wildcard character as part of the entry name. In z/OS V1R11, IDCAMS now provides more selective criteria on the DELETE command: A new MASK keyword now allows you to specify data set name selection criteria using a mask entry-name, or key filter, with the new keyword.

Data set separation by volume

In z/OS V1R11, SMS supports the allocation of critical data sets (such as DB2 partitions) on separate volumes to help reduce I/O contention. This new function is designed to expand the existing data set separation function, to allow you to specify that critical SMS-managed data sets be separated across extent pools and volumes that are not used by other data sets specified in the separation group.

Demand tape library allocation

In z/OS V1R11, there is a new JCL keyword on the DD statement, SMSHONOR. It is intended to be used to specify that allocation is to honor the device number or esoteric device type specified by UNIT. This change is intended to help improve serviceability by allowing you to specify a specific device within a tape library when gathering diagnostic information, such as when using GTF tracing.

Dynamic trace table

The new MODIFY DFS command is designed to allow you to format the DFS trace table, restart the trace, display the size and other information about the trace table, and change its size dynamically while the SMB server runs. These changes are intended to help improve serviceability for the SMB server.

DFSMS EAV: DADSM/CVAF RAS improvement

In z/OS V1R11, DFSMSdfp processing is changed to write an end-of-file (EOF) during the allocation of new non-SMS-managed data sets and those with an undefined data set organization on DASD. This change is intended to make it unnecessary to OPEN data sets for the sole purpose of writing an EOF and to avoid reading old data if the data set is read immediately after being allocated.

NFS client message globalization (NLS)

In z/OS V1.R11 the z/OS NFS client optionally issues console messages in Japanese, in addition to the existing support for English. This function is intended to use the infrastructure changes made for the message and CTrace globalization support implemented for the NFS server in release V1R10.

NFS server mount symlink support

This enhancement removes the symlink restriction from NFS version 4 mounts and provides improved interoperability with NFS V4 protocol clients.

More meaningful NFS client reason codes

Currently, various errors are returned to the z/OS UNIX System Services file system caller with reason codes that are the NFS “file-and-line” type, requiring that analysis be done by NFS development to identify the code point reporting the error. To make it easier to identify the root causes for these kinds of errors, NFS design is changed in z/OSV1R11 to convert many of these notifications to published reason codes you can use to discover the reason for the error.

Convert NFS client MemMgmt 0C1 Abend to 0F4 or 801 Abend

In z/OS V1.11, NFS client memory management provides a more meaningful diagnostic information, including the module and offset of the calling function when an error occurs.

NFS V4 server delegation support

The NFS V4 protocol provides the ability for an NFS server to temporarily delegate management of a file's resources to an NFS client. With this release, an NFS client will be able to request NFS V4 delegation authority for MVS data sets from the z/OS NFS server. Prior to this release the delegation operations were not supported.

The key purpose of the delegation is to provide improved performance by eliminating communications with the NFS server. The actual performance improvement obtained will vary greatly depending on the NFS requests issued and the data volumes being transferred.

DFSMS RAS enhancements

In z/OS V1R11, DFSMS volume selection for striped data sets is changed to be more consistent with volume selection for unstriped data sets. This new design is intended to treat quiesced and enabled volumes the same way, assure that volumes remain below the high threshold defined for a storage group after a stripe is allocated, treat normal and overflow storage groups identically, and ignore the multi-tiered storage group attribute and volume preference attributes.

1.13.1 DFSMShsm enhancements

In z/OS V1R11, the following DFSMShsm enhancements are provided.

DFSMShsm enhancement for backup copy retention period

The existing DFSMShsm (H)BACKDS command allows you to create backup versions of specified data sets. In z/OS V1.11, function is added to allow you to specify a retention period on the data set backup command. The specified retention period is intended to keep an individual backup copy for either a shorter or longer than normal period of time.

DFSMShsm fast replication enhancements

In z/OS V1R8, DFSMShsm introduced support to recover individual data sets from copy pool backup versions. DFSMShsm provided support for preallocated, cataloged data sets residing on the same volumes they were on when backed up. Deleted or moved data sets had to be reallocated and cataloged to the same volumes on which they resided when backed up in order to recover them. In z/OS V1R11, DFSMShsm now has a design to capture catalog information for the data sets within a copy pool at the time of the backup, and use that information to recover the data set.

DFSMShsm ML1 volume enhancements

Currently, DFSMShsm data sets (migration and backup) that are allocated on ML1 volumes are physical sequential data sets. These data sets are allocated in the DFSMShsm ML1 volume pool in such a way that the volume pool fills up evenly. Total free space, without regard to fragmentation, is used for ML1 volume selection. These characteristics result in the following shortcomings:

- Data sets whose expected size after compaction (if active and used) is greater than 64K tracks cannot be migrated or backed up to disk.
- At any given time, there might not be enough free space on any ML1 volume for larger data sets because all of the volumes are filled up evenly.

In z/OS V1R11, DFSMShsm addresses these issues and enables ML1 OVERFLOW volumes to be selected for migration processing in addition to their current use for data set backup processing. DFSMShsm enables these ML1 OVERFLOW volumes to be selected for migration/backup of large data sets and the determining value is externalized through a new SETSYS command. Additionally, DFSMShsm will allocate migration/backup copies for large data sets as a large format physical sequential (LFS) data set.

A coexistence APAR is required to enable downlevel DFSMShsm to tolerate LFS format DFSMShsm data sets. Downlevel DFSMShsm installations (pre V1R11) will be able to recall and recover data sets from a V1R11 DFSMShsm LFS migration and backup copies whether they come from NOOVERFLOW or OVERFLOW volumes. ARECOVER's of ML1 data sets that are LFS will fail on the recovery to ML1 volumes on downlevel systems.

1.13.2 DFSMSrmm enhancements

In z/OS V1R11, the following DFSMSrmm enhancements are provided.

DFSMSrmm simplified monitoring and management

In z/OS V1R11, DFSMSrmm provides enhancements intended for simplified monitoring and management as follows:

- The DFSMSrmm CIM agent is updated to support CIM level 2.17, and the CIM agent and providers are updated to run with OpenPegasus 2.8.1.

- The DFSMSrmm TSO command support for extended searching for volumes is intended to provide more flexibility and scope for querying a wider range of volume attributes such as dates, actions, options, and flag settings.
- Extensive changes to the DFSMSrmm report generator are intended to improve usability, enable more report customization, and simplify the way selection information can be specified. The changes are intended to further improve the reporting available for DFSMSrmm, DFSMShsm, and other DFSMS components.
- A new interface to the DFSMSrmm API provides a function call based interface. The new interface supports associated error messages in addition to error codes.

DFSMSrmm RAS enhancement

In z/OS V1R11, a new design intended to simplify DFSMSrmm vital record specifications (VRS) handling allows location definitions to be changed after a VRS has been defined.

DFSMSrmm CDS updates

In z/OS V1R11, DFSMSrmm provides a new utility to enable CDS updates made in test or recovery environments to be repeated against the original CDS. This is intended to enable a complete and up-to-date record of tape volume contents and status to be retained in a single CDS in a production RMMplex.

DFSMSrmm usability

In z/OS V1R11, the following DFSMSrmm usability enhancements are available.

- Increased flexibility in the DFSMSrmm ISPF dialog for specifying volume type, storage group name, and creation date and time when adding new volumes, and improved navigation from data set information to the actual policies used for the data set.
- Improved DFSMSrmm TSO command line parsing with consistent syntax across the various subcommands (such as ADDVOLUME, CHANGEVOLUME, GETVOLUME) and parmlib options.
- A new capability in the EDGINERS utility, to support reading and cross-verification of tape label information with the DFSMSrmm control data set, intended to allow you to cross-check the tape label details with the tape management system.
- New DFSMSrmm parmlib options intended to provide flexibility in how tape generation data sets are managed for cyclic retention and help avoid the need for a usermod for duplicate generation retention.
- An enhanced sample volume installation exit, CBRUXVNL, includes additional checks and customization designed to determine whether a volume is a tape and whether it is system-managed. Also, exit customization is planned to be enhanced to make it easy for installations where there are no external non-library drives. These customization options are designed to require little or no assembler language knowledge, and to make the sample exit installed and active by default.

DFSMSrmm scalability and performance

In z/OS V1R11, DFSMSrmm provides several scalability and performance enhancements.

- Scalability and performance of the DFSMSrmm TSO command line interface and API are improved by enabling API callers to request multiple resources returned in a single API call, reducing the overhead of API and command processing, setting REXX variables to null instead of blank, and reducing the number of REXX stem variables set with the count of entries returned.
- DFSMSrmm now supports multiple installation exit routines for each of the DFSMSrmm installation exits by exploiting the z/OS dynamic exit facility for all exits. Today the

installation exits created for use with DFSMSrmm are enabled for dynamic change, easily refreshed without impacting tape processing. However, they do not support multiple instances of the exit, so clients must merge together code from multiple sources into a single exit routine. The MVS Dynamic Exits Facility provides a way to enable multiple instances of an installation exit, and will be used for all DFSMSrmm exit points.

1.13.3 DFSMSoam enhancements

In z/OS V1R11, the following DFSMSoam enhancements are provided.

DFSMS OAM archive retention enhancements

With z/OS V1R11, OAM design is changed to add these retention-related enhancements:

- Deletion-protection, intended to prevent object deletion prior to an object's expiration date. This design will allow you to specify that deletion-protection be turned on and off with no restrictions on changing expirations.
- Deletion-hold prevents object deletion until the object is RELEASED. This function can be used to ensure that objects are not deleted during periods when the regulatory retention period has lapsed but other legal requirements mandate that the records continue to be maintained.
- Event-based-retention provides object expiration date dependent on external event notification. Traditionally, an OAM object can expire automatically based on its age, its usage, or a specific date (derived from its management class or a management-class approved object-specific retention period, if provided). Event-based-retention provides an alternate mechanism whereby retention time for an object can be initiated by an external event.
- Retention-protection prevents object deletion prior to an object's expiration date, and does not allow an expiration date to be changed to an earlier date. The objective of detention protection is to prevent deliberate or accidental deletion of data until its specified retention criterion is met.

These new enhancements are intended to enhance protection against inadvertent object deletion.

DFSMSoam 2000 MB object support

In z/OS V1R10, OAM implemented 2 GB object support to enable applications to store objects up to 2000 MB (2097152000 bytes) in size using DB2 on direct access storage. In z/OS V1R11, OAM is designed to extend that support to the tape tier of the OAM storage hierarchy. This support is intended to provide full support for objects up to 2000 MB in size on both DASD and tape and is expected to reduce the need to separate large binary strings into multiple objects and simplify the application interface by eliminating the need for applications to materialize entire objects before storing them.

1.13.4 DFSMS EAV support

In z/OS V1R11, the following DFSMS EAV support is provided.

DFSMS EAV VTOC refresh after DVE

z/OS V1R11 now refreshes the volume table of contents (VTOC) information related to volume size following Dynamic Volume Expansion. This new function is intended to allow you to use new space on volumes immediately without the need for manually refreshing the VTOC.

DFSMS EAV volumes

In z/OS V1R11, support is added to several components to allow extended format sequential data sets to be placed in the extended addressing space on extended address volumes (EAVs) in addition to VSAM data sets, for which support was introduced in z/OS V1R10. This support is planned to help ease space constraints by allowing you to move additional data to larger volumes and help simplify storage management by allowing you to store your data on a smaller number of volumes.

These enhancements provide the following new functional capabilities:

- Support for a new data set level attribute, EATTR, allows you to control the migration of non-VSAM data sets to EAS by making it possible to specify whether data sets can be partly or entirely located in EAS. This attribute is supported for both VSAM and non-VSAM data sets, as follows:
  - NO indicates no extended attributes and ineligibility to reside in EAS. This default is for non-VSAM data sets.
  - OPT indicates that extended attributes are allowed, and that a data set is eligible for placement in EAS. This default is for VSAM data sets.
- Support for a new JCL keyword for EATTR.
- Support for setting and displaying the EATTR attribute for data sets that can reside on Extended Address Volumes (EAV).
- Language Environment XL C/C++ runtime support for processing extended format sequential data sets that have extended attributes, including those data sets that reside in the extended addressing space.
- RACF support for discrete profiles for non-VSAM data sets that have extended attributes, including those that reside in EAS, in addition to the support available now for VSAM data sets. However, the RACF database must not be allocated in EAS.
- AMASPZAP support for SYSIN and SYSPRINT data sets that have extended attributes, including those data sets that reside in the extended addressing space.
- SVC dump support for dump data sets with extended attributes, including those that reside in EAS, including SYSMDUMPs and transaction dumps. Also, SNAP and ABDUMP services support the placement of dump data sets having extended attributes, including those data sets that reside in the extended addressing space.
- z/OS FTP support for reading from and writing to sequential extended format data sets in the (EAS) extended address space of an EAV.
- Program Management Binder support for input and SYSPRINT data sets having extended attributes, including those data sets that reside in the extended addressing space.
- JES2 support for external writers and spool offload to use output data sets having extended attributes, including those data sets that reside in the extended addressing space.
- TSO/E support for data sets having extended attributes, including those that reside in EAS, in the ALLOCATE, TRANSMIT, RECEIVE, SUBMIT, RACONVRT, and LISTDSI commands.
- Support by the direct access device storage manager component of DFSMS (DADSM) for single extents to span track- and cylinder-managed space for data sets with extended attributes. This will be intended to make it possible for an EAV to be entirely occupied by a single data set when the data set's maximum size is equal to or larger than the volume size.

1.13.5 DFSMS NFS enhancements

In z/OS V1R11, the following DFSMS NFS enhancements are provided.

NFS dynamic detect and report external function request delay

In z/OS V1R11, a new NFS design is intended to detect that an object access call has been outstanding for an inordinate amount of time and issue an operator message that includes the operation and the path name for the object being delayed. This design applies to both z/OS UNIX System Services and MVS data access calls.

NFS eliminates mvslogin/mvslogout for RPCSEC_GSS requests

A new NFS design in z/OS V1R11 is intended to improve ease of use and interoperability by eliminating the mvslogin and mvslogout requirement by NFS Client users when communicating with a SAF z/OS NFS server. The RPCSEC_GSS framework is intended to provide strong authentication with cryptographic protection and a means for the servers to validate the identities being claimed by client users, thus bypassing additional calls made by the z/OS NFS server to RACF services. Client users or applications that initiate RPCSEC_GSS workloads to an SAF z/OS NFS server can exploit this functionality.

NFS completion messages for operator commands

In z/OS V1R11, NFS provides completion messages for NFS server operator commands.

DFSMS OCE performance and RAS

In z/OS V1R11, changes to the Open/Close/End-of-Volume (OCE) component of DFSMS include the following enhancements:

- Fast positioning of labeled (SL) and unlabeled (NL) cartridges on 3590 and later technology tape using Position Relative commands. This is intended to provide faster data access to tape files during EOV tape concatenation.
- Avoiding the reprocessing of the same volume sequences when duplicate volume serial numbers are detected during multi-volume tape processing, and an installation-wide abend option through the LABAN (Label Anomaly) exit for multi-volume tape conditions that generate messages IEC709I, IEC710, IEC711I, and IEC7012I.
- Enhancing DASD data set output expiration date processing to externalize never-expire expiration dates in message IEC507D. This is intended to provide an installation automation option to enforce never-expiring dates using automation.

1.13.6 DFSMS VSAM

DFSMS VSAM data trap

In this release, the VSAM component of DFSMS provides a KSDS data trap that checks the data component CIs before they are written to DASD. If a data component was corrupted, a SVC dump is generated and a console message is issued to notify the user that the trap has encountered a problem with a data set. This data trap will improve serviceability for VSAM record management.

DFSMS VSAM trace enhancement

In this release, the VSAM component of DFSMS improves the VSAM record management trace and VSAM internal trace by expanding the usability. Changes include adding new Hook points throughout the VSAM record management processing, improving the serialization of the trace routine, and enhancing the trace IPCS-generated output. In addition, minor

enhancements are made to VSAM internal tracing (VSAM footstep) to improve the first failure data capture for VSAM.

Note: This support allows you to capture VSAM diagnostics data without having to re-IPL your system, simplifying VSAM problem diagnosis.

DFSMS VSAM SMB performance

The performance of VSAM system-managed buffer (SMB) direct optimized access bias can be impacted by too few index buffers being allocated when opening a data set that is small at first but grows as it is processed. In z/OS V1R11, the system's use of the SMBVSP parameter is enhanced to make the amount specified apply to both data and index buffer space.

1.14 Sysplex availability with GDPS

Enhanced GDPS sysplex timer toleration for k-system resilience

These changes are designed to allow the GDPS controlling system (K-system) image to continue running for a period of time, even after losing its sysplex time source (Sysplex Timer or STP), so that it can continue to fulfill its role as the driver of GDPS's sysplex recovery actions, such as coordinating Metro Mirror volume failover and HyperSwap, removing failed systems from the sysplex, and restarting failed systems and workloads at the recovery site. This is intended to help improve the resiliency of GDPS processing, including HyperSwap.

1.15 zAAP and zIIP enhancements

In z/OS V1R11, the following zAAP and zIIP enhancements are provided.

HiperDispatch improvements for zIIPs

The HiperDispatch design is changed to improve the performance of large scale z/OS systems that include zIIP processors. These changes are intended to improve system performance for large LPARs with very busy zIIPs and multiple zIIPs.

zAAP on zIIP

In z/OS V1R11, support is added to enable workloads that are processed on a System z Application Assist Processor (zAAP) to be processed on a System z Integrated Information processor (zIIP). In the case when no zAAPs are installed and there are zIIPs installed, a zAAP workload can be dispatched on a zIIP specialty engine. Dispatching workloads on specialty engines is intended to improve price performance. And enabling zAAP workloads to run on zIIPs is intended to allow installations with smaller specialty engine needs to utilize the cost benefits.

1.16 WLM enhancements

ITDS WLM exploitation

In z/OS V1R11, IBM Tivoli Directory Server for z/OS exploits workload manager (WLM) functions. This enhancement is designed to allow performance goals to be set for work running in IBM Tivoli Directory Server for z/OS and to take advantage of WLM's health notification service. This support allows classification of particular IP addresses or

distinguished names (DNS) into exception enclaves so their system resource consumption can be throttled. Use of the health notification service is expected to help prevent work from being directed to malfunctioning servers when their response time is very short.

Remove limit of 1,000 tasks

Currently a WLM server address space is able to use at a maximum 1000 tasks that can select work from a WLM service class. As applications grow, this can be a constraint, preventing more tasks from being started. Increasing this limit will allow applications, such as WebSphere Application Server, to grow beyond the current constraint.

1.17 RACF enhancements

The need for strong platform security is a core attribute and one of the basic values of the z/OS platform. The mainframe is an ideal security hub for the enterprise. The IBM commitment to z/OS system integrity coupled with the latest security enhancements can help your business protect users, applications, and data, which can ultimately help manage risk and meet compliance guidelines. Centralized definition, application, maintenance, and management of security policies help simplify security infrastructures as well.

Logon statistics suppression

If you have a large number of RACF users who log on to various applications many times every day, the recording of logon statistics can cause considerable RACF database I/O activity. In z/OS V1R11, RACF now allows you to specify that logon statistics be recorded only one time per day for certain applications. This is intended to limit the I/O and serialization activity for the RACF database and help improve RACF performance.

RACF - R_admin resource support

In z/OS V1R11, IBM extends the existing ITDS interface to allow you to perform RACF administration tasks for general resource profiles using the SDBM backend in addition to user and group profiles. The RACF R_admin callable service is extended so that general resource profile data can be extracted in a high performance manner from the RACF database, and the SDBM backend of the IBM Tivoli Directory Server for z/OS server is enhanced to manage RACF Resource profiles.

This is intended to allow you to use LDAP operations to create, modify, delete, and display discrete and generic resource profiles and access lists. Also planned are enhancements to allow setting RACF options using an SDBM modify command and enhancements to the change logging extended operation interface to provide change log entries for these profiles in addition to those for users and groups. This support is intended to allow you to manage these profiles by LDAP, using the same familiar operations that are currently used to manage RACF user, group, and connect profiles.

In addition, RACF is providing a REXX interface to the extract functions of the R_admin callable service (IRRSEQ00). This allows a caller to easily obtain information from any RACF profile (excluding DATASET profiles) that the caller's identity is authorized to see, including SETROPTS data.

RACF and z/OS identity propagation

This support provides the ability for z/OS transactional subsystems to associate a user's distributed identity with a RACF user ID under z/OS security (RACF) control, meanwhile maintaining the user's original identity information for audit purposes. This improves

interoperability between platforms and provides differentiated value for both host-centric and heterogeneous application environments.

RACF program signature generation and verification

In RACF, the program management binder and program management loader are now designed to add generalized functions that enable program objects residing in PDSEs to be digitally signed and verified. These functions are intended to allow you to sign and verify applications you have written, to enable software vendors to sign their applications, and for the system to verify signatures of these program objects. This includes program management binder support for storing digital signatures in program objects, and loader support to verify program object signatures before using them when you specify that signatures are to be verified. This is intended to provide another capability that can help you ensure that change control processes and procedures are enforced.

RACF SMF unload support for WebSphere Application Server and TKLM

RACF provides an SMF unload utility for clients to unload security-relevant SMF records into both a flat file format suitable for loading into DB2 for query/reporting and into XML format. WebSphere Application Server V7 introduces an audit infrastructure that enables the WebSphere runtime to produce audit records for security-relevant events. Similarly, the Tivoli key life cycle manager (TKLM) will also provide SMF audit records on z/OS. The RACF SMF unload utility is planned to be updated to support both the SMF type 83 subtype 5 records written by WebSphere Application Server and the type 83 subtype 6 records written by TKLM.

RACF zSecure request: Profile name in authorization exits

In z/OS V1R11, RACF provides the name of the profile used for an authorization check to the post-processing authorization exits. The profile name can be separate from the resource name sent as input to the authorization check, because the profile used for the authorization check can be a generic profile. This support can be used by an installation or a vendor product that might want to track RACF profile names that are used for authorization checks.

1.18 Security

In z/OS V1R11, the following security enhancements are provided.

ITDS enhanced replication

IBM Tivoli Directory Server for z/OS is designed to provide enhanced replication function equivalent to that currently provided by the IBM Tivoli Directory Server. This will include the following: Replication of subtrees of the Directory Information Tree (DIT) to a specific server, multi-tier topology referred to as cascading replication, assignment of server role (master or replica) by a subtree, filtered replication, support of gateway replication, schema replication, enhanced conflict resolution, extended operations, and command line utilities.

The enhanced replication functions are intended to allow easier migration of LDAP server workloads to z/OS.

RACDCERT and PKI Services enhancements

In z/OS V1R11, RACF RACDCERT and PKI Services are enhanced. RACDCERT services are updated to extend the current multi-byte character support. The RACDCERT support is intended to allow installation, retrieval, and authentication functions on certificates to specify multi-byte characters outside of the 1047 code page, allowing characters used in additional languages to be used in the Subject Distinguished Name. PKI Services is now designed to

support the “SHA256 with RSA encryption” signature algorithm. This enhanced support is intended to allow the SHA256 signature algorithm to be used for signing certificates, for certificate and authority revocation lists (CRL/ARL), and for OCSP responses.

PKI Services certificate key generation extensions

In z/OS V1R11, new designs for key generation and key archival/recovery capabilities are introduced for PKI Services. This new support is intended to give certificate requesters the option to generate public/private key pairs themselves as they can today or to have PKI Services generate key pairs. This new design is intended to allow a requester to retrieve a certificate for which the key pair was generated by PKI Services using its private key.

Support for NTLMv2

This support allows modification of the SMB login processing to support the NTLMv2 authentication protocol. Currently, SMB uses the authentication protocol NTLM. This item is being used to implement NTLMv2 as the authentication protocol (second generation NT LanMan authentication utilizing HMAC-MD5). NTLMv2 is the Windows Vista default authentication protocol.

1.18.1 Network security

The components discussed here provide network accessibility to z/OS security services, leveraging the core security and crypto components of the platform in an enterprise-wide role, thereby amplifying the value of the z/OS core infrastructure. Components such as z/OS PKI services have been maturing to the point where significant value and savings can be realized, helping improve overall cost of ownership and cement z/OS as a core part of IT in these enterprises.

Kerberos - TCP/IP resolver location setting update

In z/OS V1R11, z/OS Communications Server design for Kerberos in a sysplex is changed by updating its interaction with TCP/IP resolver settings. The objective of this new support is to utilize Language Environment API calls so system administrators do not need to provide the location of the TCP/IP Resolver configuration file to Kerberos.

Kerberos - Keytab merge utility

This facility allows for the merging of two keytabs into one, enabling easier consolidation of KDCs.

SSL - TLS V1.1

In z/OS V1R11, System SSL design is updated to support the Transport Layer Security (TLS) version 1.1 protocol as defined in RFC4346. This support is intended to allow System SSL applications to exploit the protocol as well as ensuring continued interoperability with other SSL implementations that support the TLS V1.1 protocol.

SSL - X.509 V3 certificate support

In z/OS V1R11, System SSL is updated to support X.509 v3 certificates and X.509 v2 Certificate Revocation Lists (CRL) at the RFC3280 level. Currently, z/OS System SSL is coded to create and validate X.509 certificates according to RFC2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile. This support is expected to allow System SSL applications to create and utilize certificates based on RFC3280, but also to retain backward compatibility with existing applications of z/OS System SSL that might be using X.509 certificates created according to RFC2459.

ICSF - IPv6

The U.S. government has defined a new protection profile for IPv6 that takes advantage of the AES symmetric encryption algorithm. ICSF will provide two new services that support the AES-based AES-XCBC-MAC-96 and AES-XCBC-PRF-128 algorithms. This is intended to meet the government standard.

Eliminate UID(0) requirement for SMB server

In z/OS V1R11, new support allows the SMB server process to run under a user ID that has been permitted to the BPX.SUPERUSER profile in the RACF Facility class. This is planned to eliminate the current requirement for the SMB server user ID to have UID(0).

SMB server client security permissions

In z/OS V1R11, the SMB server is designed to dynamically recognize changes to the SMB client's security permissions.

SSL - TLS Extensions

In z/OS V1R11, System SSL is updated with support for TLS Extensions (RFC3546). TLS Extensions are designed to be used to provide additional functionality for the Transport Layer Security (TLS) protocols by allowing TLS clients and servers to exchange supplementary information during the SSL handshake. This additional functionality can be useful for customizing SSL implementations subject to specific environmental or circumstantial constraints, such as limitations on bandwidth or memory. TLS Extensions are utilized as part of the CLIENT-HELLO and SERVER-HELLO messages in the TLS handshake. Support is being provided for the server name indication, maximum fragment length negotiation, and truncated HMAC functions.

System SSL FIPS 140-2

The National Institute of Standards and Technology (NIST) is the U.S. federal technology agency that works with industry to develop and apply technology, measurements, and standards. One of the standards published by NIST is the Federal Information Processing Standard Security requirements for cryptographic modules referred to as FIPS 140-2. In z/OS V1.11, System SSL is providing a mode of operation designed to meet the NIST FIPS 140-2 Level 1 criteria. This mode is intended to restrict a System SSL application to using FIPS approved algorithms, key sizes, and SSL protocols.

SMB Windows Vista support

The SMB server is enhanced to support the Windows Vista business and Windows Vista Enterprise operating systems running as SMB clients.

AT-TLS enhancements

z/OS Communications Server application transparent SSL/TLS support (AT-TLS) is updated to support System SSL functions that have been added since z/OS V1R7, including support for:

- TLSv1.1
- Using RFC3280 to validate a certificate
- Negotiation and use of a truncated HMAC
- Negotiation and use of a maximum SSL fragment size
- Negotiation and use of handshake server name indication
- Setting the CRL LDAP server access security level

This support is intended to combine the simplicity of using AT-TLS with the ability to use the latest system SSL capabilities.

Additionally, z/OS Communications Server Application transparent SSL/TLS support (AT-TLS) is updated to support FIPS-140-2 compliance for applications that use ATTLS to provide secure connections.

IPSec enhancements

Management of IPSec is improved through enhancements to the IPsec command, the Network Management Interface (NMI), and system monitoring facility (SMF) records. This provides improved usability of z/OS Communications Server IPsec for network security.

1.18.2 Enhanced platform security

ICSF query function

Clients expect top performance and a suite of crypto functions from System z. The decision on hardware or software implementations is made by the middleware software based on knowing what hardware options the client has installed and the crypto software functions supported by the installed software at execution time. Java, System SSL, and PKCS #11 need a single service that summarizes the crypto algorithms available on the system so a decision can be made to use the system services or provide their own implementations. ICSF is providing a service, ICSF query algorithms, to provide that information in a convenient form.

ICSF 14-digit PAN support

ICSF provides services for major credit card vendors, such as VISA and MasterCard, to generate and verify the verification values. The verification values are used in the process of validating the authenticity of the cards. In z/OS V1R11, the VISA Card verification value (CVV) service generate (CSNBCSG) and VISA Card verification value (CVV) service verify (CSNBCSV) callable services are extended to support 14-digit Diner's Club primary account numbers (PANs). Support for additional lengths (15-digit, 17-digit, and 18-digit PANs) is planned to be included for future applications.

ICSF secure key AES

The advanced encryption standard (AES) is a National Institute of Standards and Technology specification for the encryption of electronic data. It is expected to become the accepted means of encrypting digital information, including financial, telecommunications, and government data. AES is the symmetric algorithm of choice, instead of Data Encryption Standard (DES) or Triple-DES, for the encryption and decryption of data. ICSF will support the AES encryption algorithm with secure (encrypted) keys of 128, 192, and 256 bits.

Key store policy controls with ICSF

ICSF is introducing the concept of a key store policy that provides a set of policy controls to allow installations the ability to further limit application control of key material and centralize the point of control to security administrators. The new set of policy controls extends ICSF's use of the z/OS security manager and provides additional protections for key material defined in the CKDS and PKDS based on policies defined by the security administrator.

The new set of policy controls includes:

- Key token authorization checking
- Default key label checking
- Duplicate key label checking

- Granular key label access control
- Symmetric key label export control

NFS RPCSEC_GSS V4 client security

The NFS V4 protocol facilitates the usage of multiple security mechanisms and the ability to administer the server name space with the deployment of separate security policies. The protocol provides for an NFS client to ask what security the server requires for a given file object, and this communication between the client and the server is achieved by the use of SECINFO operation of the NFS V4 protocol. For improved interoperability with the NFS server, z/OS support allows the NFS client to have the ability to negotiate security flavors with the server through the SECINFO operation.

1.19 z/OS XL C/C++ compiler

C/C++ z10 improvements

In z/OS V1R11, the XL C/C++ includes additional z10 hardware exploitation in the compiler through the addition of the PREFETCH option. This option is designed to enable heuristics to automatically generate z10 prefetch instructions, as appropriate. This is expected to help reduce the effects of memory latency by beginning to fetch data before it is thought to be needed. Additionally, optimization and tuning improvements are planned for the compiler. These changes are intended to improve the performance of generated code without the need for changes to its source.

METAL C enhancements

New function is available for METAL C, including the addition of the strtof(), strtod(), and strtold() functions, enhancement of the sprintf and sscanf family functions in the runtime library with additional format specifiers that will allow the use of floating-point numbers, and the creation of a static library that users can bind with directly instead of using the system vector. Also, support is included for the “extern” and “all” keywords to the Metal C PROLOG and EPILOG compiler options. This is intended to allow you to specify whether the specified PROLOG and EPILOG apply to external functions only or to both internal (static) and external functions within the source file, and to make it easier to integrate C code with assembler code.

C/C++ improved debug capability

In z/OS V1R11, XL C/C++ has improvements intended to make it easier to debug C/C++ code on z/OS. Specifically, we have added support for:

- Captured Source

  This is intended to allow programmers to specify that the compiler is to imbed source code information into the debug side file for future debugging. This is designed to make it possible to step through the program even in situations where the original source file is not available at debugging time, for example, debugging on a production system when the source resides on another system.

- CDADBGLD utility

  This is intended to provide those without access to z/OS UNIX System Services with support equivalent to the z/OS UNIX System Services dbgld utility; that is, the ability to create a module map for debugging. This is designed to make programmers more productive through improvements in debugger startup time.

C/C++ client requests

In z/OS V1R11 XL C/C++, new designs intended to satisfy client requirements include UNICODE Literal Support, which is designed to add support for the char16_t and char32_t types through the use of typedefs in C and as native types in C++. This is designed to make it easier to port code exploiting these types to z/OS. Also included is a new SKIPSRC option intended to allow you to specify that the compiler exclude source from the listing that has been #ifdef-ed out. This is designed to improve programmer productivity by improving readability of the compiler listings.

C/C++ TR1 libraries support

In z/OS V1R11, XL C/C++ has upgraded support for the “ISO C++ technical report on Library Extensions”. This is designed to make it easier to port C++ code from other platforms to z/OS.

C++0x language standard support

In z/OS VR11, XL C/C++ design is extended to support parts of the ISO/IEC JTC1/SC22/WG21 draft C++0x language standard. This support is intended to make it easier to port C++ code from other platforms to z/OS.

C/C++ GCC source compatibility subset

In z/OS V1R11 XL C/C++, IBM plans to expand support for source code compatibility with the GNU Compiler Collection (GCC), including C support for the statement expression construct to allow programmers to use loops, switches, and local variables within an expression, and C and C++ support for the zero extent array construct, which can be used as the last element of a structure, as a header for a variable length object. These features are designed to make it easier to port C/C++ code using these constructs on other platforms to z/OS XL C/C++.

1.20 z/OS Unicode

JIS-X0213 support on z/OS Unicode Services

New z/OS Unicode services support is implemented for the new conversion table between CCSID 1390/1399 and Unicode for JIS X0213. This support is added for the 2002 version of the 1390/1399 CCSIDs, which contain JIS X0213 characters. The X0213 standard promoted by the Japanese government is expected to conform with the family registration law for names of people that has been established by the Ministry of Justice and the Kanji glyph style standard specified by the Ministry of Education, Culture, Sports, Science and Technology.

PKI Web pages

In z/OS V1R11, the PKI Services Web pages that are provided to help request certificates and perform certificate administration tasks are enhanced. The Web pages are provided in Java in addition to REXX CGI. This change is intended to make it easier to integrate PKI Services Web pages with other Web-based applications you might have by allowing you to define them in XML files with XML schemas and then customize them using modifiable Java server pages (JSP) files.

1.21 Dynamic channel path management

I/O configuration management comprises a complex set of tasks. Configurations must be defined with various factors in mind, including single point of failure avoidance and expected bandwidth needs. After a configuration has been defined, its I/O bandwidth requirements

often change. This means you must monitor the system and respond to changes to assure your workloads meet their response time goals.

FICON DCM

Dynamic channel path management (DCM) was introduced for ESCON channels with z/OS V1R1 on z900 servers. DCM is intended to help simplify I/O configuration definition, maximize the utilization of installed hardware, and enhance RAS. In z/OS V1R11, IBM extends DCM to manage FICON channels in addition to ESCON channels. This is expected to help improve performance in most environments, because the system can react to changing bandwidth requirements much more quickly than people can.

Dynamic CHPID management does this by allowing you to identify channels to be managed. Managed channels are not assigned to a specific control unit but instead are dynamically assigned to control. This makes it possible for the system to respond to I/O bandwidth requirements, and is expected to make it easier to define and manage I/O configurations because you no longer need to maintain configurations that address workload variations.

FCX measurement facility support - WLM

This item is intended to support the reporting of performance data for the new “High Performance FICON” channels.

1.22 XML enhancements

XML VSCR improvement

XML code page dependent tables that are used for z/OS XML System Services processing are moved above the bar, which frees up common storage below the bar that is currently used.

2

z/OS V1R11 installation overview

This chapter provides information about the current release of z/OS regarding installation, coexistence, and fallback considerations. Although coexistence and fallback, which play an important part in planning for migration to the latest release, might at first seem like separate topics, in fact they are quite related in that both deal with an earlier level of a system being able to tolerate changes made by a later level, as explained here.

The chapter also describes IBM policy regarding the releases that are supported for coexistence, fallback, and migration, and indicates which specific releases are supported.

The following topics are covered:

- Elements, features and FMID changes in z/OS V1R11
- z/OS ordering and deliverable key dates
- Driving system requirements for z/OS V1R11
- Coexistence considerations for z/OS V1R11
- Fallback considerations for z/OS V1R11
- DASD storage requirements for z/OS V1R11
- Using your existing JES2, JES3, and SDSF with z/OS V1R11
- Functions withdrawn in z/OS V1R11

2.1 Elements, features, and FMID changes in z/OS V1R11

There are no new elements added in z/OS V1R11, and no elements have been removed.

The Integrated Security Services LDAP Server, FMID HRSL3AA, has been withdrawn in z/OS V1R11; therefore, the FMID is not included in z/OS V1R11. z/OS V1R10 is the last release to provide Integrated Security Services LDAP Server FMID HRSL3AA.

When you order z/OS V1R11, you can also order program products. See *z/OS Planning for Installation*, GA22-7504, to identify the program products that are required if you plan to use certain functions provided by the elements in z/OS V1R11.

Hardware requirements

The minimal hardware requirements for z/OS, as well as additional hardware needed by specific z/OS elements and features, are documented in *z/OS Planning for Installation*, GA22-7504.

z/OS V1R11 runs only in z/Architecture® mode. It must be IPLed in z/Architecture mode, and only on the following servers:

- IBM System z10™ Enterprise Class (z10 EC)
- IBM System z10 Business Class (z10 BC)
- IBM System z9 Enterprise Class (z9 EC), formerly the IBM System z9 109 (z9-109)
- IBM System z9 Business Class (z9 BC)
- IBM eServer™ zSeries 990 (z990)
- IBM eServer zSeries 890 (z890)
- IBM eServer zSeries 900 (z900)
- IBM eServer zSeries 800 (z800)

2.1.1 z/OS ordering and deliverable key dates

The program number for z/OS Version 1 Release 11 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 V1R11, there are only two export controlled unpriced features:

- z/OS Security Level 3
- Communications Server Security Level 3

  z/OS Security Level 3 contains the sub-elements:
  - IBM Tivoli Directory Server Security Level 3
  - Network Authentication Service Level 3
  - OCSF Security Level 3
  - System Secure Sockets Layer (SSL) Security Level 3

ServerPac and SystemPac

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: z/OS V1R10 ServerPac is no longer orderable after October 2009. SystemPac (a fee deliverable) is still available.

z/OS V1R11 delivery dates

Note the following key dates for z/OS V1R11:

- September 11, 2009

  This is the first date for ordering z/OS V1R11 ServerPac, SystemPac, CBPDO using CFSW configuration support, or ShopzSeries, the Internet ordering tool.

  The program number for z/OS Version 1 Release 11 is 5694-A01.

  Note: When ordering this program number, remember to order all the optional features that you were licensed for in previous releases of z/OS.

- September 25, 2009

  z/OS V1R11 general availability through ServerPac, CBPDO, and SystemPac.

- October 2009

  z/OS V1R10 ServerPac planned ordering ends.

2.2 Products related to z/OS V1R11

When you order z/OS V1R11, you can also order program products. *z/OS Planning for Installation*, GA22-7504, identifies the program products that are required if you plan to use certain functions provided by the elements in z/OS V1R11.

Following are several products to consider ordering in conjunction with z/OS V1R11.

SMP/E V3R5

SMP/E for z/OS V3R5 (5655-G44) is the current version. 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 clients 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.

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

IBM Ported Tools for z/OS Version 1 Release 1 (5655-M23)

IBM Ported Tools for z/OS is an unpriced program product that can be ordered with z/OS. It can run on z/OS V1R4 and higher. It has been generally available since May 28, 2004.

Additional information about IBM Ported Tools for z/OS is available from the following URL:

http://www-03.ibm.com/servers/eserver/zseries/zos/unix/ported/

IBM Ported Tools for z/OS: Perl for z/OS Feature (5655-M23)

The IBM Ported Tools for z/OS: Perl for z/OS Feature provides a port of the Perl (Version 5.8.7) scripting language to the z/OS UNIX System Services platform. To order, Perl for z/OS is an unpriced feature of the IBM Ported Tools for z/OS, an unpriced licensed program. It has been generally available since June 16, 2006.

The program numbers are IBM Ported Tools for z/OS (5655-M23) and IBM Ported Tools for z/OS Software and Subscription (5655-M29).

IBM Ported Tools for z/OS: PHP for z/OS Feature (5655-M23)

The IBM Ported Tools for z/OS: PHP for z/OS Feature (PHP for z/OS) provides a port of the PHP (Version 5.1.2) scripting language to the z/OS UNIX System Services platform.

To order, PHP for z/OS is an unpriced feature of the IBM Ported Tools for z/OS, which is an unpriced licensed program product. It has been generally available since May 25, 2007. The program numbers are IBM Ported Tools for z/OS (5655-M23) and IBM Ported Tools for z/OS Software and Subscription (5655-M29).

The IBM Ported Tools for z/OS: Supplementary Toolkit for z/OS Feature provides miscellaneous utilities that are commonly used on other platforms, and now on the z platform. For z/OS V1R11, bzip2 is provided. To order, the Supplementary Toolkit for z/OS is an unpriced feature of the IBM Ported Tools for z/OS, which is an unpriced licensed program.

IBM Ported Tools for z/OS: IBM HTTP Server V7.0 (5655-M23)

The IBM Ported Tools for z/OS: IBM HTTP Server V7.0 feature has the same functionality found in the WebSphere Application Server for z/OS V7.0 product.

To order, IBM HTTP Server V7.0 is an unpriced feature of the IBM Ported Tools for z/OS unpriced licensed program. When IBM Ported Tools for z/OS: IBM HTTP Server V7.0 is ordered, it will be sent with IBM Ported Tools for z/OS (which contains the OpenSSH function). Note that an order of IBM Ported Tools for z/OS will *not* include IBM HTTP Server V7.0. OpenSSH and IBM HTTP Server V7.0 functions can be run independently.

XML Toolkit for z/OS V1R10 (5655-J51) and V1R9

The IBM XML Toolkit for z/OS, V1R10 continues to provide enhanced support for the XML Parser, C++ Edition and the XSLT Processor, C++ Edition. For a description, visit the following URL:

http://www.ibm.com/zseries/software/xml/

To order, XML Toolkit for z/OS V1R10 became available in December, 2008. The program number for the XML Toolkit for z/OS is 5655-J51.

IBM 64-bit SDK for z/OS, Java Technology Edition Version 6 (5655-R32)

The IBM 64-bit SDK for z/OS, Java Technology Edition Version 6 provides a Java software development kit (SDK) at the SDK 6 level for the z/OS platform.

To order, the IBM 64-bit SDK for z/OS, Java Technology Edition Version 6 became generally available in December 2007. The program number is 5655-R32.

IBM 31-bit SDK for z/OS, Java Technology Edition Version 6 (5655-R31)

This product is the z/OS 31-bit Java product that supplies the Java SDK 6 APIs.

To order, the IBM 31-bit SDK for z/OS, Java Technology Edition Version 6 became generally available in December 2007. The program number is 5655-R31. For the most current information about the IBM 31-bit SDK for z/OS, Java Technology Edition Version 6 and IBM 64-bit SDK for z/OS, Java Technology Edition Version 6, see the zSeries Java Web site:

http://www.ibm.com/servers/eserver/zseries/software/java/

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

The IBM 64-bit SDK for z/OS, Java 2 Technology Edition, Version 5 is the z/OS 64-bit Java product that supplies the Java SDK 5 APIs.

To order, the IBM 64-bit SDK for z/OS, Java 2 Technology Edition Version 5 became generally available in December 2005. The program number is 5655-N99.

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

The IBM 31-bit SDK for z/OS, Java 2 Technology Edition Version 5 (5655-N98) product is the z/OS 31-bit Java product that supplies the Java SDK 5 APIs. To order, this product became generally available in December 2005. The program number is 5655-N98.

For the most current information about 64-bit SDK for z/OS, Java 2 Technology Edition Version 5 or 31-bit SDK for z/OS, Java 2 Technology Edition Version 5, see the zSeries Java Web site:

http://www.ibm.com/servers/eserver/zseries/software/java/

2.3 Driving system requirements for z/OS V1R11

The *driving system* is the system image (hardware and software) that you use to install the target system. The *target system* is the system software libraries and other data sets that you are installing. You log on to the driving system and run jobs there to create or update the target system. After the target system is built, it can be IPLed on the same hardware (same LPAR or same processor) or on hardware other than that used for the driving system. If your driving system will share resources with your target system after the target system has been IPLed, be sure to install applicable coexistence service on the driving system before you IPL the target system.

Requirements for the driving system

The driving system requirements for installing z/OS by way of ServerPac or dump-by-data set SystemPac are:

- For the operating system, you can use any of the following:
  - z/OS V1R9 or z/OS V1R10
- The Customized Offerings Driver V3 (5751-COD)

2.3.1 Other system requirements

A TSO/E session on the IPLed system must be established using a locally-attached or network-attached terminal.

This system requires the following:

- Proper authority by using the RACFDRV installation job as a sample of the security system definitions required so that you can perform the installation tasks.
- Proper security to install the UNIX files requires the following:
  - The user ID you use must be a superuser (UID=0) or have read access to the BPX.SUPERUSER resource in the RACF FACILITY class.
  - The user ID you use must have read access to FACILITY class resources BPX.FILEATTR.APF, BPX.FILEATTR.PROGCTL, and BPX.FILEATTR.SHARELIB (or BPX.FILEATTR.* if you choose to use a generic name for these resources). The commands to define these FACILITY class resources are in SYS1.SAMPLIB member BPXISEC1.
  - Group IDs uucpg and TTY, and user ID uucp, must be defined in your security database. These IDs must contain OMVS segments with a GID value for each group and a UID value for the user ID. (For ease of use and manageability, define the names in uppercase.)

    Note: The group ID and user ID values assigned to these IDs cannot be used by any other IDs. They must be unique. You must duplicate the required user ID and group names in each security database, including the same user ID and group ID values in the OMVS segment. This makes it easier to transport the z/OS UNIX file systems (HFS or zFS) from test systems to production systems. For example, the group name TTY on System 1 must have the same group ID value on System 2 and System 3.

    If it is not possible to synchronize your databases, then you will need to continue running the FOMISCHO job against each system after z/OS UNIX is installed.

Minimum requirements to service the new target system

The following requirements must be met to service the new z/OS V1R11 system:

- z/OS V1R11 program management binder

  The program management binder is changed in z/OS V1R11.
- SMP/E and HLASM

  SMP/E and HLASM elements are the same in z/OS V1R10 and V1R11.

Note: If your ServerPac order contains a product that uses ++JARUPD support, the driving system requires IBM SDK for z/OS, Java 2 Technology Edition, V1 (5655-I56) at SDK 1.4 or later.

2.4 Coexistence considerations for z/OS V1R11

Coexistence occurs when two or more systems at different software levels share resources. The resources can be shared at the same time by different systems in a multi-system configuration, or they can be shared over a period of time by the same system in a single-system configuration. Examples of coexistence include:

- 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)

z/OS systems can coexist with specific prior releases; see Figure 2-1. This is important because it provides you with the flexibility to migrate systems in a multi-system configuration using rolling IPLs rather than requiring a systems-wide IPL.

The way in which you make it possible for earlier-level systems to coexist with z/OS is to install coexistence service (PTFs) on the earlier-level systems. Complete the migration of all earlier-level coexisting systems as soon as you can.

Note: Coexistence PTFs allow existing functions to continue to be used on the earlier-level systems when run in a mixed environment that contains later-level systems.

Coexistence PTFs are not aimed at allowing new functions provided in later releases to work on earlier-level systems.

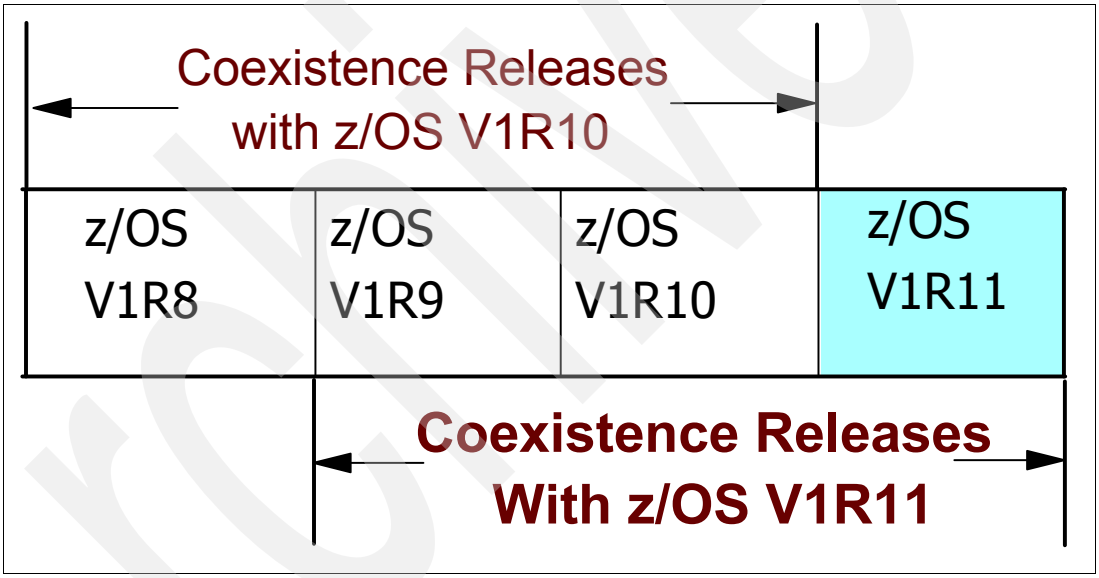


Figure 2-1 Coexistence levels

2.4.1 Fallback considerations for z/OS V1R11

Fallback is a backout or return to the prior level of a system. Fallback can be appropriate if you migrate to z/OS V1R11 and, during testing, encounter severe problems that can be resolved by backing out the new release. By applying fallback PTFs to the prior release system before you migrate, the prior release system can tolerate changes that were made by the new system during testing.

Note: Remember that new functions can require that all systems be at z/OS V1R11 level before the new functions can be used. Therefore, be careful not to exploit new functions until you are fairly confident that you will not need to back out your z/OS V1R11 systems, because fallback maintenance is not available in these cases. Consult the appropriate element or feature documentation to determine the requirements for using a particular new function.

Fallback PTFs

Fallback is relevant in all types of configurations, that is, single-system or multi-system, with or without resource sharing. As an example of fallback, consider a single system that shares data or data structures, such as user catalogs, as you shift the system image from production (on the “old” release) to test (on the new release) and back again (to the old release). The later-level test release might make changes that are incompatible with the earlier-level production release. Fallback PTFs on the earlier-level release can allow it to tolerate changes made by the later-level release. This means that:

- Coexistence of a z/OS V1R11 system with a z/OS V1R9 or z/OS V1R10 system is supported.
- Fallback from a z/OS V1R11 system to a z/OS V1R9 or z/OS V1R10 system is supported.
- Migration to a z/OS V1R11 system from a z/OS V1R9 or z/OS V1R10 system is supported.

As a general reminder, always plan to have a backout path when installing new software by identifying and installing any service required to support backout.

Fallback is at a system level, rather than an element or feature level, except for JES2, SDSF, and JES3. That is, except for JES2, SDSF, and JES3, you cannot back out an element or feature; you can only back out the entire product. JES2, SDSF, and JES3 fallback can be done separately as long as the level of JES2, SDSF, or JES3 is supported with the release of z/OS and any necessary fallback PTFs are installed.

2.4.2 DASD storage requirements for z/OS V1R11

If you are migrating to z/OS V1R11 from z/OS V1R10, or if you will have another product set than your previous release, you can see increased need for DASD. 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.

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

DASD storage requirements

For z/OS V1R11, installation requirements are as follows:

- The total storage required for all the target data sets is approximately 6400 cylinders on a 3390 device.
- The total storage required for all the distribution data sets listed in the space table is 9200 cylinders on a 3390 device.
- The total file system storage is approximately 3100 cylinders on a 3390 device for the ROOT file system.
- 50 cylinders for the `/etc` file system.
- 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 V1R11 that are both cross-zone and use CALLLIBs, therefore the SMPLTS is not needed for permanent storage).

2.4.3 Using your existing JES2, JES3, and SDSF with z/OS V1R11

Migrate to the JES2 or JES3 that comes with z/OS V1R11 at the same time you migrate to the rest of z/OS V1R11, or as soon as possible thereafter. In this way, you benefit directly from the new functions in the z/OS V1R11 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 V1R11 so that you can stage your migration to z/OS V1R11 (that is, migrate your JES2 or JES3 later).

Allowable BCP, JES2, and SDSF combinations

With z/OS, the JES levels supported by a given release are the same as the JES levels that may coexist in the same multi-access spool (MAS) or multisystem complex with the JES delivered in that z/OS release. In addition, starting with z/OS V1R9, the SDSF release level must be the same as the JES2 release level. Table 2-1 lists the allowable JES2 and SDSF combinations.

Table 2-1 JES2 and SDSF allowable combinations

BCP release	JES2 release allowed	SDSF release allowed
z/OSV1R11	z/OSV1R9	z/OSV1R9
	z/OSV1R10	z/OSV1R10
	z/OSV1R11	z/OSV1R11

Note: With z/OS V1R11, the SDSF release level must be the same as the JES2 release level.

With z/OS V1R10 and later releases, SDSF is supported with JES3. Prior to z/OS V1R10, SDSF was supported only with JES2. The JES3 and SDSF releases allowed with z/OS V1R11 BCP are shown in Figure 2-2.

Table 2-2 JES3 and SDSF allowable combinations

BCP release	JES3 release allowed	SDSF release allowed
z/OSV1R11	z/OSV1R9	Not allowed with JES3
	z/OSV1R10	z/OSV1R10
	z/OSV1R11	z/OSV1R11

Service policy

Currently, IBM policy is to provide maintenance (service) for each release of z/OS for three years following its general availability (GA) date. However, service on the last release of a version might be extended beyond the intended three-year period. Prior to withdrawing service for any version or release of z/OS, IBM intends to provide notice at least 12 months in advance. The service policy for z/OS also applies to any enhancements (including but not limited to Web deliverables), as shown in Table 2-3.

Table 2-3 Service policy dates

Version and Release	General Availability (GA)	End of Service (EOS)
z/OS V1R8	29 September 2006	September 2009 (announced)
z/OS V1R9	28 September 2007	September 2010 (planned)
z/OS V1R10	26 September 2008	September 2011 (planned)
z/OS V1R11	25 September 2009	September 2012 (planned)

The following URL provides the latest information about end-of-service dates.

http://www.ibm.com/servers/eserver/zseries/zos/support/zos_eos_dates.html

2.5 Functions withdrawn in z/OS V1R11

The functions listed here are withdrawn in z/OS V1R11. z/OS V1R10 is the last release to provide these functions, so take this fact into account as you plan your migration to z/OS V1R11.

- The removal of these functions can have migration actions which you can perform now, in preparation for z/OS V1R11.

  See *z/OS Migration*, GA22-7499,for the migration action details.

- Communications Server support for Network Database (NDB) function has been withdrawn in z/OS V1R11.

  If you were using this function, use the distributed data facility (DDF) provided by DB2 for z/OS and the DB2 Runtime Client.

- Communications Server support for Bind DNS 4.9.3 function has been withdrawn in z/OS V1R11.

  Starting with z/OS V1R11, you must use BIND DNS 9.2.0.

- Communications Server support for Boot Information Negotiation Layer (BINL) has been withdrawn in z/OS V1R11.

  Use another product such as IBM Tivoli Provisioning Manager for OS Deployment V5 (5724-Q99) for network-based operating system installation services.

- Communications Server support for Dynamic Host Configuration Protocol (DHCP) server function has been withdrawn in z/OS V1R11.

  Use a DHCP server on a system other than z/OS.

- Integrated Security Services LDAP Server is not supported in z/OS V1R11.

  Use IBM Tivoli Directory Server in z/OS V1R11.

- Interactive problem control system (IPCS) problem management subcommands were functionally stabilized in 1981 but left in IPCS. Since then, clients have been advised to migrate to other problem management tools. z/OS V1R10 was the last release that included the subcommands.

  As of z/OS V1R11, the subcommands have been removed from z/OS. In place of the subcommands, consider using IBM Tivoli Information Management for z/OS V7 (5698-A08) or a similar product.

Note: There are IBM Migration Health Checks that you can use to determine whether the Communications Server functions that are withdrawn in z/OS V1R11 are being used on earlier z/OS releases. See *z/OS Migration*, GA22-7499, for additional information.

3

BCP contents supervisor enhancements

This chapter discusses new BCP contents supervisor enhancements for z/OS V1R11. Dynamic LPA services allow modules to be added to or deleted from LPA after IPL. Because any product can own LPA modules and have an interest in updating their control structures, an exit (CSVDYLPA) is provided from dynamic LPA services. Providing lists of modules added to or deleted from LPA, the exit is intended to be used as a notification mechanism so that products can update internal control structures with the new module addresses.

The following new functions and enhancements are covered:

- MapMVS - dynamic LPA support for a HFS path name
- RMODE 64 data tables
- JPQ search after a find for a HFS load

3.1 MapMVS - dynamic LPA support for HFS path name

MapMVS is a dynamic LPA enhancement used to store additional information for module or program objects loaded in LPA through a “directed load.” MAPMVS needs to understand where a module came from so that it can re-read the module from its source and use binder services to determine the CSECT arrangement so that a performance analysis can target specific CSECTs within a module. Unfortunately, directed loads conflict with this goal, because no control structures are maintained. That cannot be changed, so we need to provide alternatives to using Directed Load. For common storage, and dynamic LPA is one such alternative. However, dynamic LPA does not support loading from HFS. It *does* support being given an address and being told about it.

This new support enhances dynamic LPA to do the following:

- Take the full path name and file name when the module is passed using the ByAddr parameter.
- Save the path name as part of the information associated with the entity so that MapMVS can get at it when it uses CSVINFO or CSVQUERY.

  Also, for ByAddr cases, to allow the user to provide the diagnostic data needed to identify the data set by its UCB address and CCHH when the load comes from a PDS(E).

Prior to this support, information for modules dynamically loaded into LPA by the user through a Directed Load was limited to:

- Entry point
- Load point
- Module length

MapMVS support

New support called MapMVS provides additional information to be associated with directly loaded LPA modules. This new support does the following:

- It allows MapMVS to return an HFS path name through CSVINFO or CSVQUERY.
- It also allows the user to provide the diagnostic data needed to identify the data set (UCB address and CCHH) when the load came from a PDS(E).

Note: Prior to the support provided by the MAPMVS feature, it was not possible to save information about program objects located in an HFS. The infrastructure for storing information about an HFS was not initially put in place because CSVDYLPA cannot itself load from an HFS.

This support will particularly assist WebSphere. Prior to this support, WebSphere must perform a “directed load from UNIX System Services” and then a dynamic LPA ByAddr operation to be able to get its executables into common storage and into LPA.

3.1.1 MapMVS support for CSVDYLPA

CSVDYLPA allows you to request dynamic LPA services. Be aware, however, that changes to LPA itself are not actually performed. This set of services truly only lets you add modules to, and delete modules from, common storage. When searching by module name, the system locates the copy of a module added by dynamic LPA services even if it was present in PLPA, MLPA, or FLPA.

With CSVDYLPA, you can request services to:

- Add one or more modules to common storage (REQUEST=ADD).
- Delete one or more modules that were previously added using dynamic LPA services (REQUEST=DELETE).
- Query information about support for LPA services (REQUEST=QUERYDYN).

Currently, CSVDYLPA ADD supports a BYADDR=YES parameter. The user first loads the module into common storage through a directed load and then calls CSVDYLPA BYADDR=YES with entry point, load point, and module length to store the information into LPA.

With MapMVS support, CSVDLPA BYADDR=YES can now pass and store additional information into the LPA structures, as follows:

- HFS path name for a program object located in an HFS
- UCB CCHH information for a PDS(E)

This information is passed through a new parameter, BYPATH, when specifying BYADDR=YES. The new parameters supported by MapMVS have the syntax shown in Figure 3-1.

```
CSVDYPLA REQUEST=ADD,MODINFOTYPE=MEMBERLIST,BYADDR=YES,
BYPATH=YES,PATHNAME=pathname,PATHNAMELEN=pathnamelen
BYPATH=NO,[UCBADDR=ucbaddr,CCHH=cchh]
```

Figure 3-1 New parameter for dynamic LPA services

The CSVDYLPA optional parameters are explained here:

PATHNAME This parameter is required when BYPATH=YES is specified. This is a string from 1 to 1023 characters. It is the path name from which the module was fetched. This must be a fully qualified path name including the leading forward slash mark (/).

PATHNAMELEN This parameter is required when BYADDR=YES, BYPATH=YES is specified. This is a fullword integer from 1 to 1023. It is the length of the path name provided by the PATHNAME parameter.

UCBADDR This parameter is optional when BYADDR=YES, BYPATH=NO is specified. It is the address of the UCB for the volume on which the first extent of the data set containing the module exists. Normally, this field is found from field DEBUCBA mapped by IEZDEB in the DEB associated with the opened DCB used to load the module. The UCB can be a “captured UCB” and therefore, the address from DEBUCBA must be passed into the IOSCAPF service and the output from that service used for the CSVDYLPA UCBADDR parameter. If you are able to determine that this is not a captured UCB, you can use its address directly.

CCHH This parameter is required when UCBADDR is specified. It consists of the CC and HH values associated with the first extent of the data set containing the module. Normally, this field comes from fields DEBSTRCC and DEBSTRHH mapped by IEZDEB in the DEB associated with the opened DCB used to load the module, for the DEB entry that corresponds to the UCB address provided by the UCBADDR parameter.

The CCHH parameter denotes a 32-bit track address consisting of a cylinder number and a track number. However, keep in mind that the

internal format of the CCHH field is not relevant to the CSVDYLPA specification. The exact number of bits that represent the cylinder number or track number is device-dependent, so even though the parameter is called CCHH, this does not mean that it has to represent a 16-bit cylinder and 16-bit track value.

CSVDYLPA service

Figure 3-2 shows an example of the new parameters when using the CSVDYLPA service.

```
CSVDYLPA REQUEST=ADD,
            BYADDR=YES,
            BYPATH=YES,
            PATHNAMELEN=MYPATHNAMELEN,
            PATHNAME=MYPATHNAMEIN,
            MODINFO=MYLPMEA,
            MODINFOTYPE=MEMBERLIST,
            NUMMOD=1,
            REQUESTOR=MYREQUESTOR,
            RETCODE=RC,
            RSNCODE=RSN,
            MF=(E,L$DYLPA)
```

Figure 3-2 CSVDYLPA ADD request using BYPATH=YES

When a dynamic LPA module is loaded using the BYADDR=YES and BYPATH=YES parameters, and is processed by CSVINFO, the area mapped by CSVMODI will include:

- MODI_NOMODI_DynlpaPathname. This is a one-bit field which is set if MODI_DynlpaPathnameLen and MODI_DynlpaPathnameAddr cannot be determined.
- MODI_DynlpaPathnameLen. This is 15-bit field which contains the length of the path name.
- MODI_DynlpaPathnameAddr. This is a four-byte pointer which contains the address of an area that contains the path name (up to the length of MODI_DynlpaPathnameLen). When MODI_DynlpaPathnameLen is 0, the value of MODI_DynlpaPathnameAddr is undefined.

Note: A module that is not dynamic LPA or is dynamic LPA but not BYADDR=YES and BYPATH=YES will have 0 for MODI_DynlpaPathnameLen.

3.1.2 MapMVS support for CSVQUERY

CSVQUERY supports PATHNAME by returning the HFS path name in the OUTPATHNAME field. It returns 1026 bytes, where the first two bytes are the length of the path name. Also returned is the UCB/CCHH information using the OUTDIAG parameter.

```
CSVQUERY OUTLENGTH=OUTLENGTH,
            OUTLENGTH64=OUTLENGTH64,
            OUTEPA=OUTEPA,
            OUTEPA64=OUTEPA64,
            OUTLOADPT=OUTLOADPT,
            OUTLOADPT64=OUTLOADPT64,
            OUTDIAG=OUTDIAG,
            OUTPATHNAME=OUTPATHNAME,
            PLISTVER=MAX,
            MF=(E,L$QUERY,NOCHECK)
```

Figure 3-3 CSVQUERY request that returns HFS path name information

3.1.3 MapMVS support for CSVINFO

CSVINFO is called with a pointer to a user routine (which so named because it is written by the user of the function). CSVINFOM fills in the MODI fields and then calls the user routine. The MODI fields exist for the convenience of the user routine.

Note: CSVINFO and its usage is described in detail in the z/OS MVS Assembler Services Guide. A utility called CSVDLPAU is shipped in SYS1.LINKLIB. This utility lists the contents of LPA with entry points, load points, and length.

SYS1.SAMPLIB(CSVSMIPR) shows an example of how to code a CSVINFO user routine (called an MIPR) in Assembler.

CSVINFO and CSVQUERY macros

Both the CSVINFO and CSVQUERY macros return information about loaded modules. A *loaded module* is a load module that has been loaded into storage. Use CSVQUERY if you need information about a particular loaded module or if your program is running in access register (AR) mode. Use CSVINFO to obtain information about a group of loaded modules or when you want information about the loaded module associated with a particular program request block (PRB) or information that the CSVQUERY macro does not provide.

IPCS support

You can write a VERBX executable that uses CSVINFO under an IPCS environment.

3.2 RMODE 64 data tables

As applications require more and more data, storage below 2 GB becomes inadequate. For tables that are loaded from data sets, there is no functionality available that can both get the table placed into storage with relocation performed. The RMODE 64 new function has been added to z/OS V1R11 to allow the user of “directed load” (currently, using the LOAD macro with the ADDR64 or ADRNAPF64 parameters) to identify an area above 2 GB as the target of the load. This provides virtual storage constraint relief.

Note: z/OS XML is exploiting this function in z/OS V1R11.

LOAD macro

The LOAD macro brings the load module containing the specified entry name into virtual storage, if a usable copy is not available in virtual storage. Control is not passed to the load module; instead, the load module's entry point address is returned in GPR 0. Load services places the load module in storage above or below 16 megabytes depending on the RMODE of the module.

The responsibility count for the load module is increased by one. The load module remains in virtual storage until the responsibility count is reduced to 0 through task terminations or until the effects of all outstanding LOAD requests for the module have been canceled (using the DELETE macro), and there is no other requirement for the module.

LOAD macro with z/OS V1R11

The LOAD macro supports new keywords with z/OS V1R11 to identify an area above 2 GB, as explained here.

- **ADDR64** and **ADRNAPF64**

  These specify that the module is to be loaded beginning at the designated address. The address must begin on a doubleword boundary. Storage for the module must have been previously allocated in the key of the eventual user. The system does not search for the module and does not maintain a record of the module when it is loaded. If you code ADDR64 or ADRNAPF64, you must also code the DCB parameter (not DCB=0), and you must not code GLOBAL or LOADPT. ADDR64 and ADRNAPF64 are only allowed with PLISTVER=1 or PLISTVER=MAX. The SYSSTATE ARCHLVL value (by specifying SYSSTATE ARCHLVL=n) must be greater than 1.

  Note: The system assumes that the RMODE of the module is consistent with this address, but the system does not check. The data set from which an ADDR64 or ADRNAPF64 request is met must *not* be a VIO data set.

  To code, specify the AD-type address, or address in 64-bit register (2)-(12), of an 8-byte field. If your program requires that the module is in an APF-authorized library, use ADDR64; otherwise, use ADRNAPF64.

- **PLISTVER**

  This is an optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

  – IMPLIED_VERSION, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

These new parameters are analogous to the existing 31-bit parameters and use 8-byte rather than 4-byte fields. ADDR64 and ADRNAPF64 may be specified on the LOAD macro and are available to any application that desires to use them, and that meets the environmental requirements. If a module is cached in VLF, LLA passes the 8-byte address to VLF. VLF populates the storage and then LLA performs relocation. If a module is not cached in VLF, CSV passes the 8-byte address to the loader.

Note: Because VIO processing in DFSMS does not handle 64-bit targets, a VIO target of a LOAD with ADDR64 is rejected.

Figure 3-4 lists examples showing where these new parameters are used.

```
       LOAD EP=MYMODULE,ADDR64=addrabove2G
       LOAD EP=MYMODULE,ADRNAPF64=addrabove2G
MYLIST LOAD MF=L,PLISTVER=1
       LOAD EP=MYMODULE,ADDR64=addrabove2G,
            MF=(E,MYLIST)
```

Figure 3-4 Examples using ADDR64 and ADRNAPF64

PLISTVER=0 is the earlier form of VER. PLISTVER=1 is the new form of VER. Using PLISTVER=MAX uses the maximum available VER.

Note: Removing the 64s works today with 31-bit addresses.

The user is responsible for making sure that the module will work if placed above 2 GB. The system does not verify that the module truly can exist above 2 GB (for example, suppose it has a 4-byte address constant to another part of itself). It is up to the exploiter to ensure this.

This is consistent with the current LOAD macro with ADDR behavior. The system does not verify that the module is truly non-executable. If an attempt is made to transfer the flow of execution to the module, the results will be unpredictable, just as with any executable code that is placed above 2 GB by any mechanism. This does not imply that support for executable modules above 2 GB is forthcoming.

3.3 JPQ search after find for HFS load

Prior to z/OS V1R11, the flow for an HFS load consisted of a call to search the job pack queue (JPQ) for usable copy of the load module. If not found in the JPQ, then a “find” was issued to locate the directory entry and continue with the fetch of the load module. This exposed a timing window in which a second copy of a reentrant module can be loaded.

In z/OS V1R11, a new protocol is used for an HFS load to avoid this timing window and ensure that only one copy of a reentrant module is loaded.

The new protocol issues a call to search the JPQ for a usable copy. If it is not found, then a “find” is issued to locate the directory entry and then re-search the JPQ and continue with a fetch only if it is not found.

4

Extended address volumes

Rapid data growth on the z/OS platform is leading to a critical problem for various clients. For many, the 4-digit device number limit (no more UCBs) is becoming a real constraint to growing data on z/OS. In addition business resilience solutions (GDPS, HyperSwap, and PPRC) that provide continuous availability are also driving this constraint because those functions use more device numbers. The IBM solution to this problem is to provide larger volumes (by about four times in R10) by increasing the number of cylinders beyond 65,520 cylinders. These new volumes have a larger number of cylinders (hundreds of millions) by implementing a new track addressing architecture. This method is discussed in this chapter.

This relief builds upon prior technologies that were implemented in part to help reduce the pressure on running out of device numbers. These include PAV and HyperPAV. PAV alias UCBs can be placed in an alternate sub-channel set. HyperPAV reduces the number of alias UCBs over traditional PAVs and provides the I/O throughput required. The benefit of this support is that the amount of z/OS addressable disk storage is significantly increased. This provides relief for clients that 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. This support is provided in z/OS V1R10.

This chapter describes a further enhancement to extended address volumes introduced with z/OS V1R11. z/OS V1R10 introduced the extended address volume (EAV), which allowed DASD storage volumes to be larger than 65,520 cylinders. The space above the first 65,520 cylinders is referred to as cylinder-managed space. Tracks in cylinder-managed space use extended addressing space (EAS) techniques to access these tracks. Data sets that are able to use cylinder-managed space are referred to as being EAS-eligible. The following EAV functions and enhancements provided by z/OS V1R11 support in DFSMS are covered:

- EAV support with z/OS V1R10
- EAV support enhancements with z/OS V1R11
- Coexistence support with pre-z/OS V1R11 systems
- Migration to EAVs
- EAV eligible data sets
- TSO/E support
- FTP EAV support
- Service aid support

4.1 EAV overview

Prior to z/OS V1R10, DASD storage was limited to 65,520 cylinders per volume. To satisfy growing DASD storage requirements, z/OS V1R10 supports extended address volume. An extended address volume (EAV) is by definition 65,521 cylinders or larger. In releases starting with z/OS V1R10, the maximum size is 262,668 cylinders.

The extra space on an extended address volume (the cylinders whose addresses are equal to or greater than 65536) is referred to as the *extended addressing space* (EAS). These cylinder addresses are represented by 28-bit cylinder numbers. On an extended address volume, the cylinders whose addresses are below 65536 are referred to as the *base addressing space*. These cylinder addresses are represented by 16-bit cylinder numbers or by 28-bit cylinder numbers whose high order 12 bits are zero (0).

The extended address volume (EAV) is the next step in providing larger volumes for z/OS environments. This support, introduced 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 grew 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, thus causing issues for the 4-digit device number limit. With EAV in z/OS V1R10, a new architecture was implemented that provides capacities of hundreds of terabytes (TBs) for a single volume and includes the following:

- An architectural maximum of 268,434,453 cylinders.
- The first releases of EAV support in z/OS V1R10 and z/OS V1R11 is limited to a volume with 223 GB or 262,688 cylinders.

This section describes the following aspects of the EAV support for both releases:

- zArchitecture data scalability
- ESS, DS8000, and PAV
- Recall of extended address volume (EAV) as introduced in z/OS V1R10
- Specific enhancements in z/OS V1R11

4.2 zArchitecture data scalability

As processing power has dramatically increased, great care has been taken and appropriate solutions deployed to ensure 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 4-1 on page 65.

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

Serialization granularity

Beginning in the 1960s, shared DASD was serialized through a sequence of RESERVE/RELEASE CCWs that are today under the control of GRS, as illustrated in Figure 4-1. This was a useful mechanism as long as the volume of data that was serialized (the granularity) in this way was not too great. However, whenever such a device grew to contain too much data, bottlenecks became an issue.

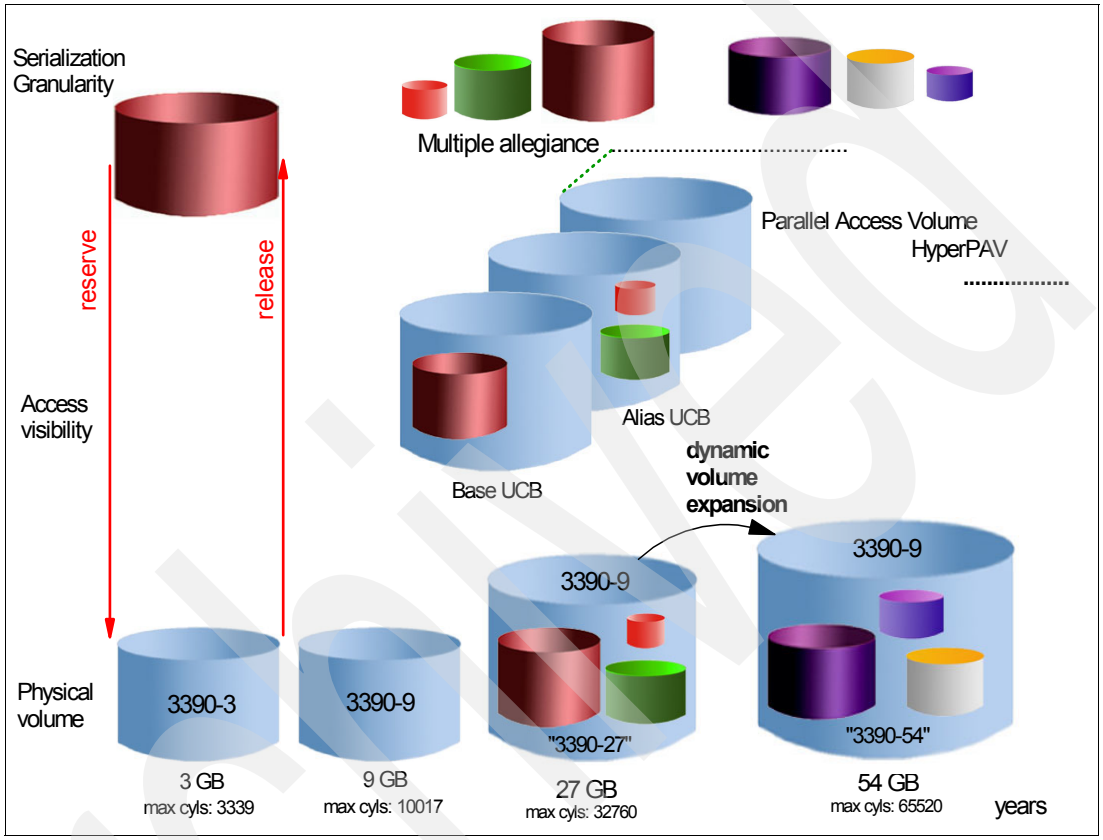


Figure 4-1 zArchitecture data scalability

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)

This relief builds upon prior technologies that were implemented in part to help reduce the pressure on running out of device numbers. These include PAV and HyperPAV. PAV alias UCBs can be placed in an alternate sub-channel set (z9 multiple subchannel support). HyperPAV reduces the number of alias UCBs over traditional PAVs and provides the I/O throughput required.

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 provides 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 the 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 generates a busy indication for the volume whenever one of the systems issues a request against the volume. This causes 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.

Channel subsystem and subchannel sets

To alleviate the addressing schema constrained by 4-x digit device addresses, significant changes in the I/O subsystem have been made in the past years. XMP allows several channel subsystems to be embodied in a CEC, with each of those channel subsystems being in charge of up to 256 CHPIDs and up to 15 LPARs. Subchannel sets allowed the possibility to directly alleviate the 4-digit constraint by defining for each LPAR (and thus, for each z/OS image), a secondary subchannel set known as subchannel set-1.

Note the following points:

- Subchannel set 0 remains the z/OS-visibly defined 4-digit addressing schema (shown in blue in Figure 4-2 on page 67). Only 65,280 devices of this 4-digit schema are visible by z/OS. There are 256 devices of subchannel set 0 that remain reserved and invisible.
- Subchannel set 1 is a 5-x digit set of addresses (shown in brown in Figure 4-2 on page 67). They are only accessible to specially-tailored functions such as PAV alias devices, PPRC second devices, and other special devices.

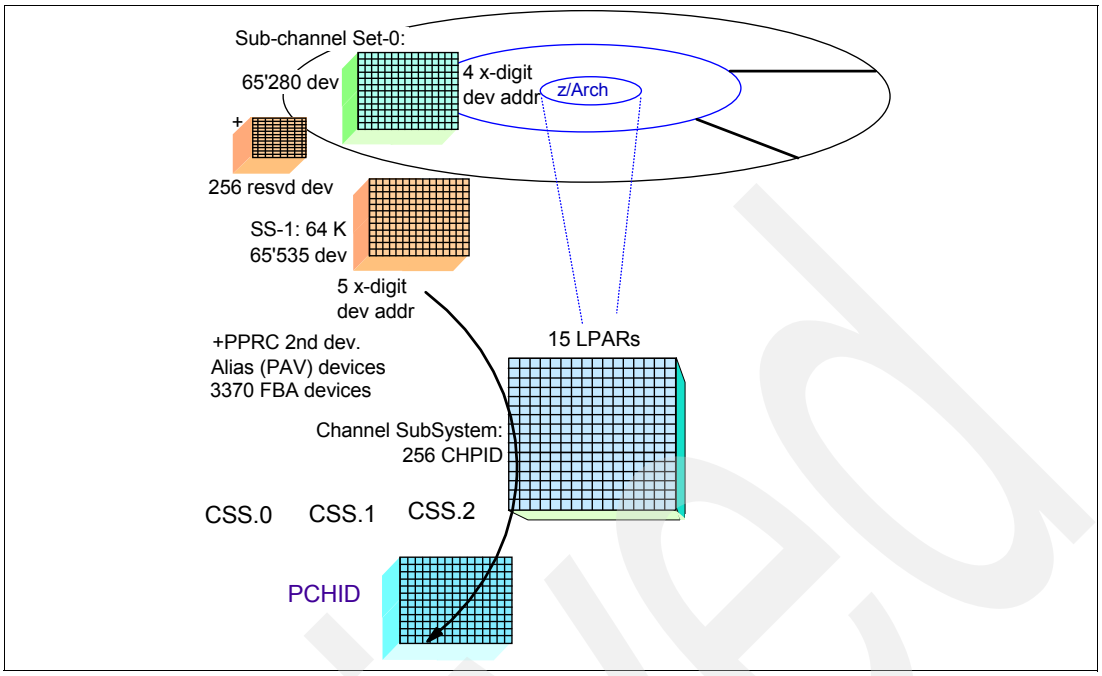


Figure 4-2 Multiple subchannel sets

Note: The maximum supported volume size today is 65,520 cylinders, which is near the 16-bit theoretical limit of 65,535 cylinders.

4.3 ESS, DS8000, PAV, and HyperPAV

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*. With alias addresses, instead of one UCB per logical volume, an MVS host can now use several UCBs for the same logical volume. 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. With ESS, alias device numbers can be defined 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.

4.3.1 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 with the following benefits:

- It eliminates I/O operation queue time for SMS-managed data sets.
- It improves performance for certain 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 that is 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 requires WLM to monitor workload and goals. However, it takes time for WLM to detect an I/O bottleneck. Then WLM has to coordinate the reassignment of alias addresses within the sysplex and the DS8000. If the workload fluctuates or has a burst character, the job that caused the overload of one volume can end before WLM reacts. In such cases the IOSQ time was not eliminated completely.

4.3.2 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, 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 queues 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 using the same (or fewer) operating system resources.

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.
- The 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 4-1 on page 65.

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.

4.4 Extended address volume

As mentioned, an extended address volume (EAV) is a volume with more than 65,520 cylinders. An EAV increases the amount of addressable DASD storage per volume beyond 65,520 cylinders by changing how tracks on ECKD volumes are addressed.

With z/OS V1R10, the IBM solution to this problem has been to provide larger volumes (by about four times) by increasing the number of cylinders beyond 65,520 cylinders. Volumes

are grown with a larger number of cylinders (hundreds of millions) by implementing a new track addressing architecture.

Dynamic volume expansion

More recently, dynamic volume expansion is a function that allows you to increase the size of a given existing volume. Dynamic volume expansion increases the capacity of open systems and System z volumes, while the volume remains connected to a host system. This capability simplifies data growth by allowing volume expansion without taking volumes offline.

EAV benefit

The benefit of this support is that the amount of z/OS addressable disk storage is further significantly increased. This provides relief for clients that 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. This support is provided in z/OS V1R10 and enhanced in z/OS V1R11.

The extended address volume is the next step in providing larger volumes for z/OS. This support is provided starting 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 65,520 cylinders or approximately 54 GB. Access to the volumes includes the use of PAV, HyperPAV, and FlashCopy SE (space-efficient FlashCopy), as previously mentioned.

4.5 Extended address volume - 3390 Model A

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. Figure 4-3 on page 71 illustrates the 3390 device types.

Important: The EAV support is provided in the DS8000 4.0 version.

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 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.

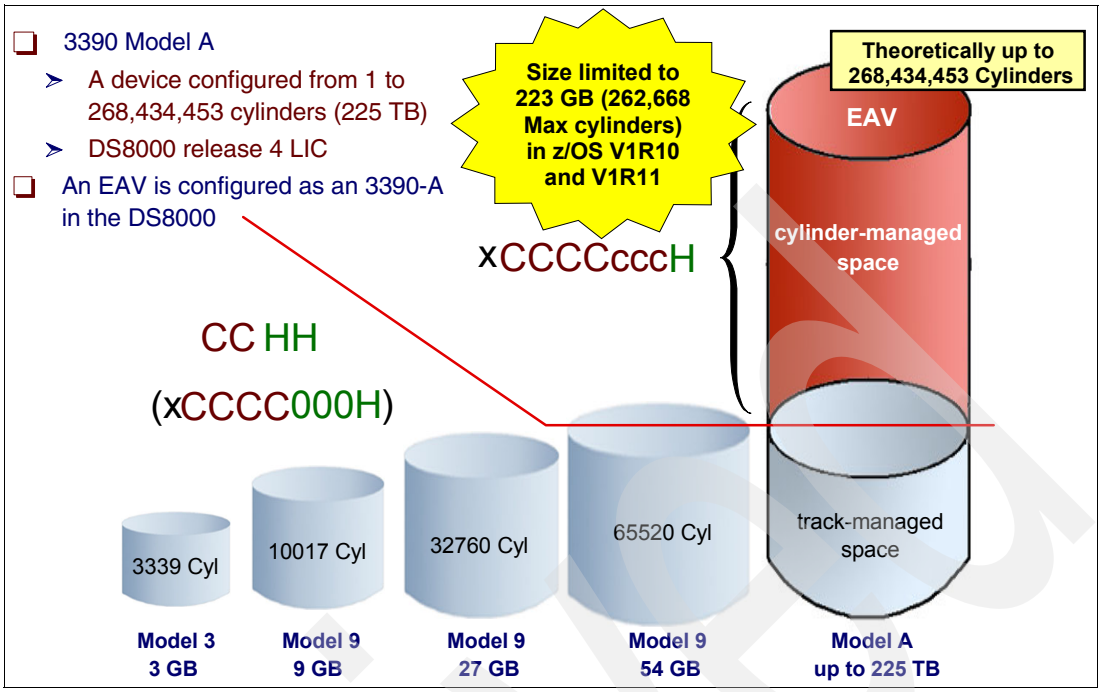


Figure 4-3 DS8000 support for device type 3390

EAV basic definitions

Note the following points regarding extended address volumes.

- Only 3390 Model A devices can be EAV.
- EAV is supported starting in z/OS V1R10 and further releases.
- The size is limited to 223 GB (262,668 cylinders) in z/OS V1R10 and V1R11.

Important: As mentioned, 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 current z/OS releases, this maximum size is currently not supported.

4.6 EAV terminology

Note: Two sets of terms are used to reference an EAV. One set is used to describe how space is managed. The other set is used to describe how the disk is addressed. The context of what is being described dictates which terminology to use.

extended address volume (EAV) This refers to a volume with more than 65,520 cylinders. Only 3390 Model A devices can be an EAV.

track address This refers to a 32-bit number that identifies each track within a volume. It is in the format hexadecimal CCCCcccH, where CCCC is the low order 16 bits of the cylinder number, ccc is the high order 12 bits of the cylinder number, and H is the four-bit track number. For compatibility with older programs, the ccc portion

	is hexadecimal 000 for tracks in the base addressing space.
extended addressing space (EAS)	On an extended address volume, this refers to cylinders with addresses that are equal to or greater than 65,536. These cylinder addresses are represented by 28-bit cylinder numbers.
base addressing space	On an extended address volume, this refers to cylinders with addresses below 65536. These cylinder addresses are represented by 16-bit cylinder numbers or by 28-bit cylinder numbers whose high order 12 bits are zero (0).
multicylinder unit	This refers to a fixed unit of disk space that is larger than a cylinder. Currently, on an EAV, a multicylinder unit is 21 cylinders and the number of the first cylinder in each multicylinder unit is a multiple of 21.
cylinder-managed space	This refers to the space on the volume that is managed only in multicylinder units. 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 EAVs.
track-managed space	This refers to the space on a volume that is managed in tracks and cylinders. Track-managed space ends at cylinder address 65519. Each data set occupies an integral multiple of tracks. Track-managed space also exists on all non-EAVs.
breakpoint value (BPV)	When a disk space request is this size or more, the system prefers to use the cylinder-managed space for that extent. This applies to each request for primary or secondary space for data sets that are eligible for the cylinder-managed space. If not enough cylinder-managed space is available, then the system will use the track-managed space or will use both areas. The breakpoint value is expressed in cylinders. When the size of a disk space request is less than the breakpoint value, the system prefers to use the track-managed area. If not enough space is available there, then the system will use the cylinder-managed space, or will use both areas.

4.6.1 EAV key design points

An important EAV design point is that IBM maintains its commitment to clients 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, as shown in Figure 4-4.

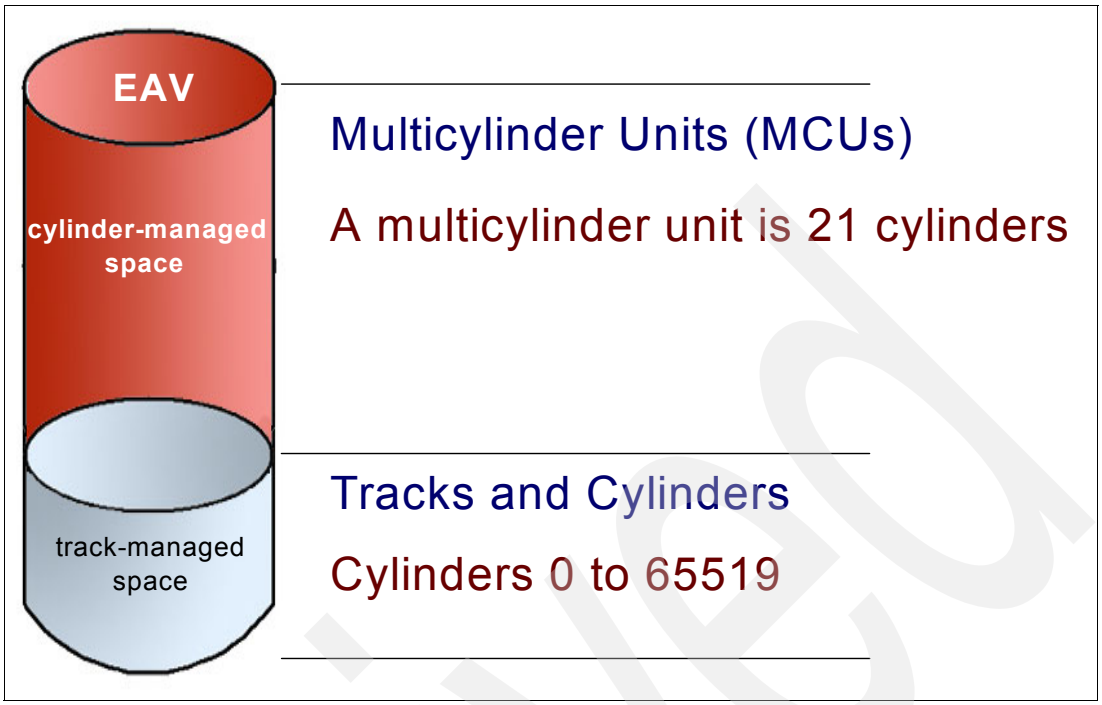


Figure 4-4 EAV and multicylinder units

Track-managed space

The track-managed space on a volume is managed in track and cylinder increments. All volumes today have track-managed space. The 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. Physical copies can be performed from a non-EAV to an EAV and have those data sets accessible.

Cylinder-managed space

The cylinder-managed space on a volume 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 is rounded up to the next multicylinder unit. The cylinder-managed space exists only on EAVs. 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, a multicylinder unit is 21 cylinders and the number of the first cylinder in each multicylinder unit is a multiple of 21; see Figure 4-4.

The 21-cylinder value for the MCU 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), 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.

4.6.2 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.

“extended address volume (EAV) This refers to a volume with more than 65,520 cylinders. Only 3390 Model A devices can be an EAV.” on page 71 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 4-5 on page 75. 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 4-5 on page 75, 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; see “New track address for extended address volume” on page 76.

  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 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 4.7.6, “New format DSCBs for EAVs” on page 87.

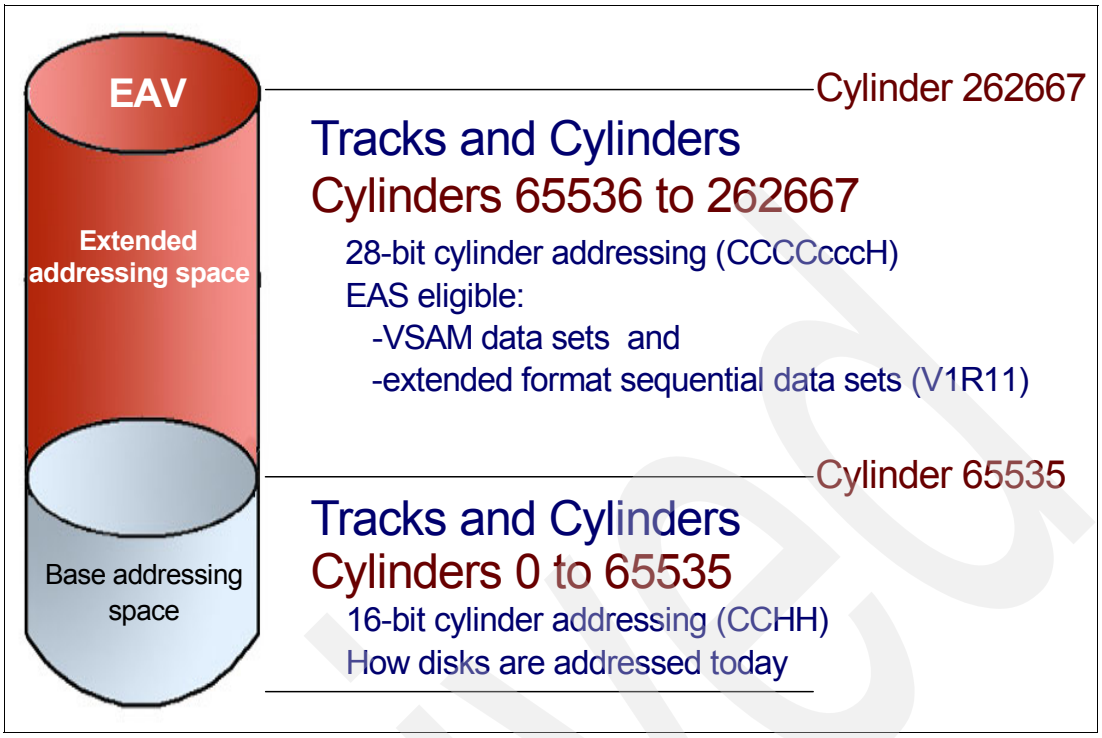


Figure 4-5 New track addressing for EAVs

Old track address

The base addressing space is the area on an EAV located within the first 65536 cylinders. As shown in Figure 4-6 on page 76, 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 a track address can be shown 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. Thus to handle cylinder numbers greater than 65,520, a new format for the track address is required.

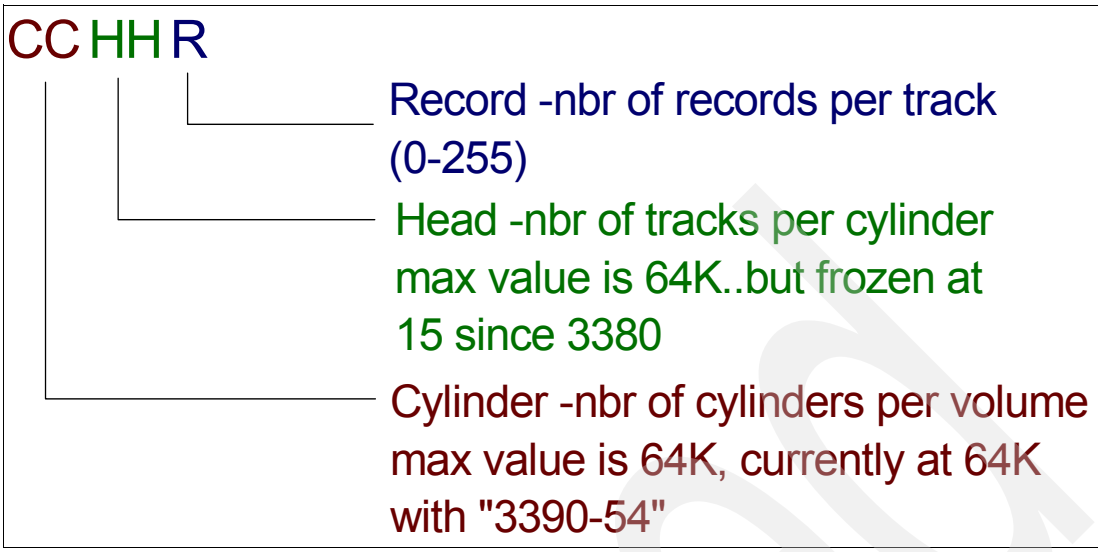


Figure 4-6 Old track address

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 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 4.7.6, “New format DSCBs for EAVs” on page 87.

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 previous track addresses.

Note: For compatibility reasons, the 32 bits in each track address on an EAV is in the following 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 4-7 on page 77.

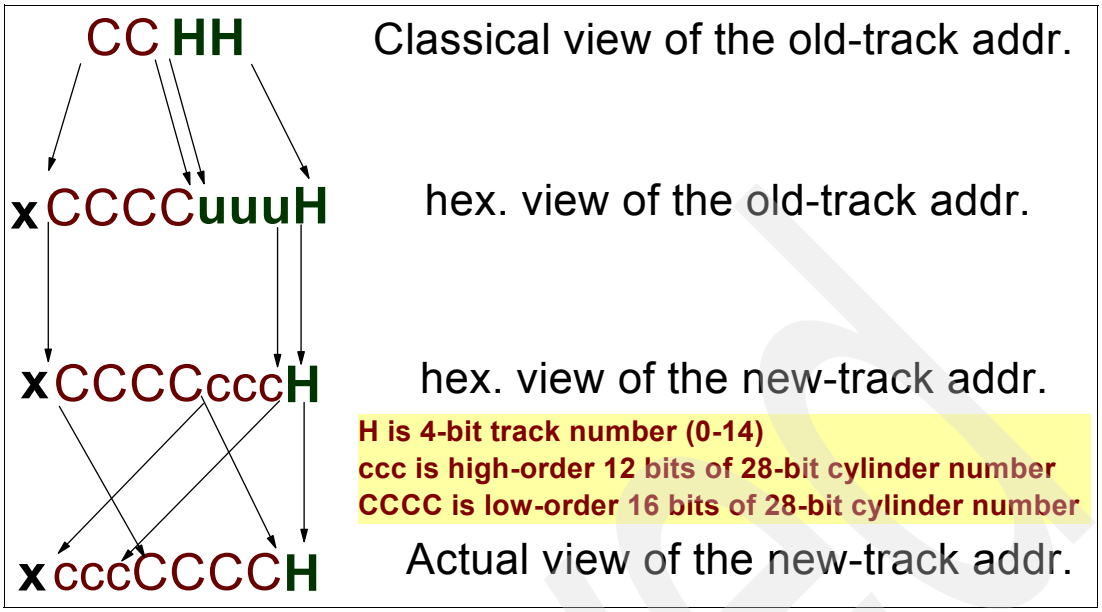


Figure 4-7 From old track address to new track address

Key points

The cylinder number is in a non-contiguous form. Reading this new track address, the hex digits must be rearranged, as shown in Figure 4-8. 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 is 28 bits and the track number is four bits. For compatibility reasons, the 32 bits in each track address on an EAV are in this format: CCCCcccH.

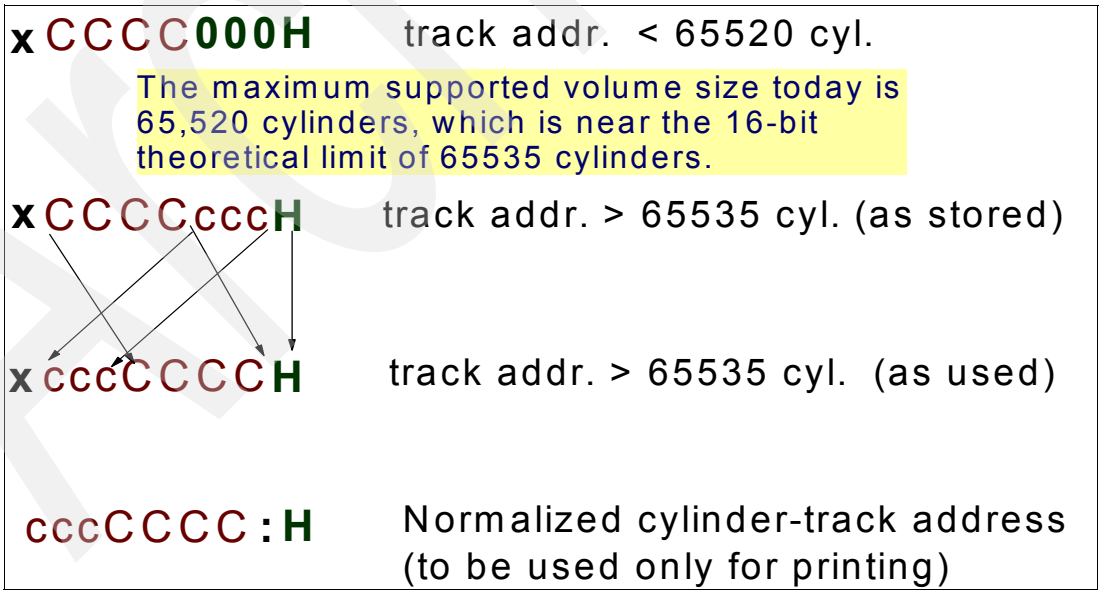


Figure 4-8 EAV: the two types of track address

Extended address volume attributes

In z/OS V1R10, each extended address volume, as shown in Figure 4-9 on page 78, 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 is to be tested independently. An exception is that a volume with more than 65520 cylinders requires format-8 and format-9 DSCBs. A non-extended address volume has none of these attributes. An extended address volume has all of the following three attributes:

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

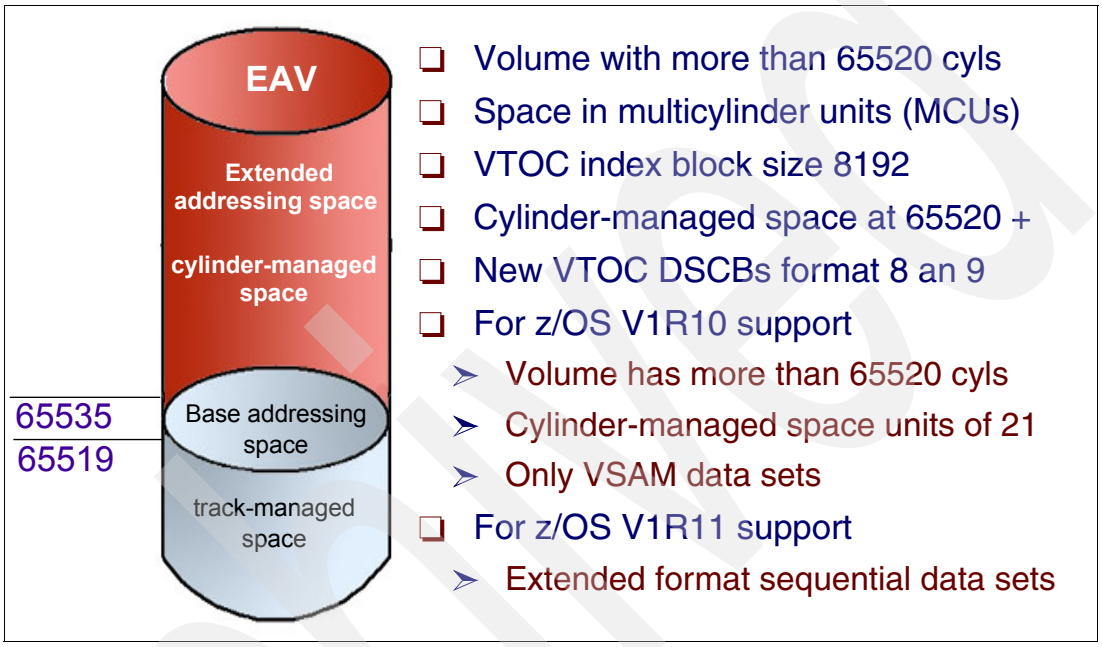


Figure 4-9 EAV attribute summary

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

4.6.3 EAS-eligible data sets

EAS-eligible data sets are 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 might 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

- **With z/OS V1R11:** Extended-format sequential data sets that are SMS-managed can be allocated in the base addressing space of an EAV.

Note that 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.

4.6.4 EAS non-eligible data sets

An EAS-ineligible data set may exist on an EAV, but is not eligible to have extents (through Create or Extend) in the cylinder-managed space.

Note the following exceptions to EAS eligibility:

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

- **With z/OS V1R11:** Non-VSAM data sets, except extended-format sequential data sets that are SMS-managed, are EAS non-eligible.

- VSAM data sets with incompatible CA sizes

Note: In a future release, several of these data sets might 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.

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.

z/OS V1R11 enhancement

z/OS V1R11 removes a number of restrictions that applied to striping volume selection, to make that function as close as possible to conventional volume selection. The following new functions are added:

- Enabled volumes are preferred over quiesced volumes.

- Volumes that do not have sufficient space below the “high threshold” to contain the stripe are eligible for selection, and not rejected outright.
- Normal storage groups are preferred over overflow storage groups.
- The storage group sequence order as specified in the ACS storage group selection routines is supported when multi-tiered storage group is requested in the storage class.
- Data set separation is supported.
- Volumes preference attributes, such as availability, accessibility, and PAV capability are supported.
- Fast Volume Selection is supported, regardless of the current specification of the FAST_VOLSEL parameter. SMS rejects the candidate volumes that do not have sufficient free space for the stripe when 100 volumes have already been rejected by DADSM for insufficient space. This is to prevent the striping allocation from overusing the system resources, because an iteration of volume reselection may consume significant overhead when there are a large number of candidate volumes.

4.7 Data migration to EAVs

The prerequisites for the implementation of extended address volumes in production is the z/OS V1R10 and z/OS V1R11, DFSMS V1R10 and DFSMS V1R11, 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 EAVs, consider the following required criteria:

- 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:
  - DS CLI to define the new 3390 Model A, as explained in “Command-line interface” on page 81
  - DS8000 Storage Manager, which is a Web browser GUI interface as explained in “Using the IBM System Storage DS8000 Storage Manager” on page 82

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.

4.7.1 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 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); see “Command-line interface” on page 81 for more information.
- Use a Web browser GUI; see “Web browser GUI” on page 82 for more information.

Note: All systems must be at the z/OS V1R10 level or above 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.

4.7.2 Command-line interface

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 4-10 shows the change of the device to a 3390 Mod 9, which is displayed using the Web browser GUI interface.

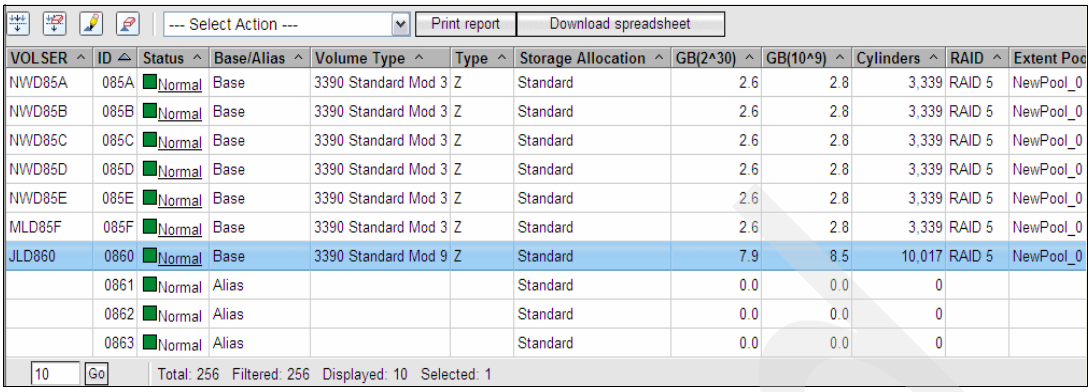


Figure 4-10 Device change from a 3390 Mod 3 to a 3390 Mod 9

4.7.3 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 beyond 65,520 cylinders is specified. The number of cylinders when defining an EAV can be defined as low as 1 cylinder and up to 262,668 cylinders.

In addition, expanding a 3390 type of volume that is below 65,520 cylinders to be larger than 65,520 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 4-11 on page 82) and then using the Select Action pull-down menu to select **Increase Capacity**.

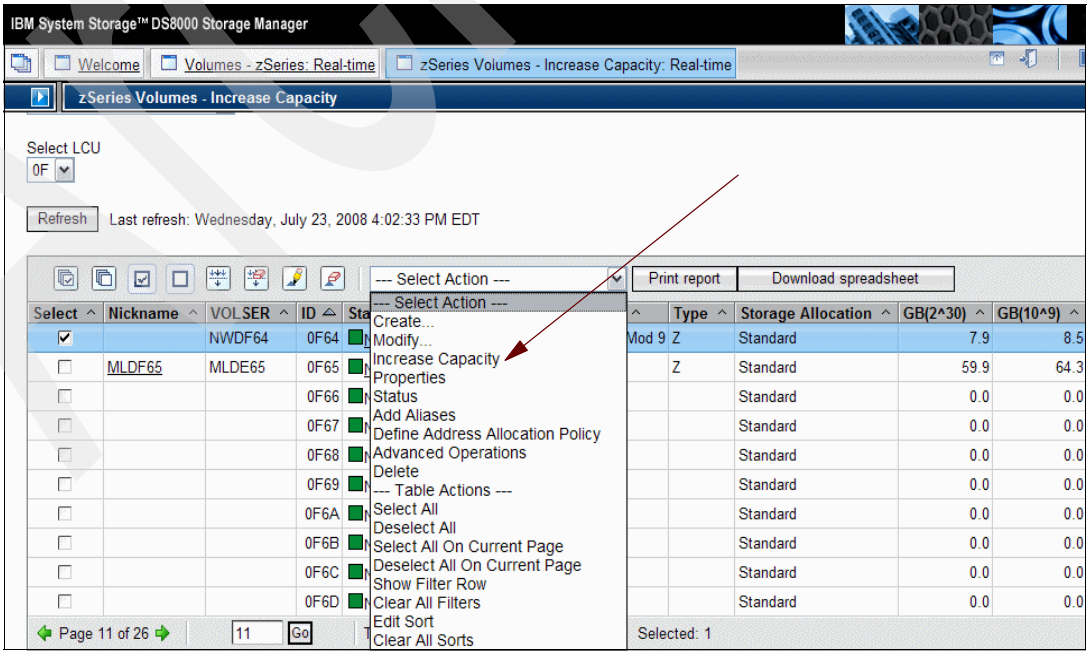


Figure 4-11 Screen to increase the size of a volume

Figure 4-12 on page 83 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.

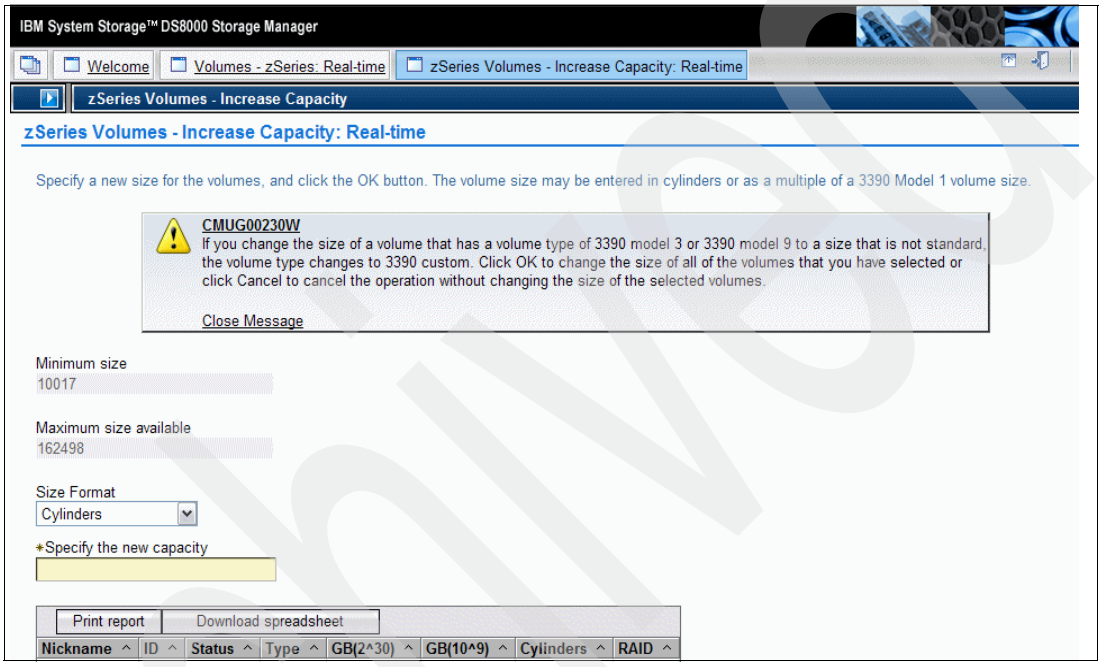


Figure 4-12 Warning message issued regarding the volume type change

After you close the message, the panel in Figure 4-13 is displayed. Here 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.

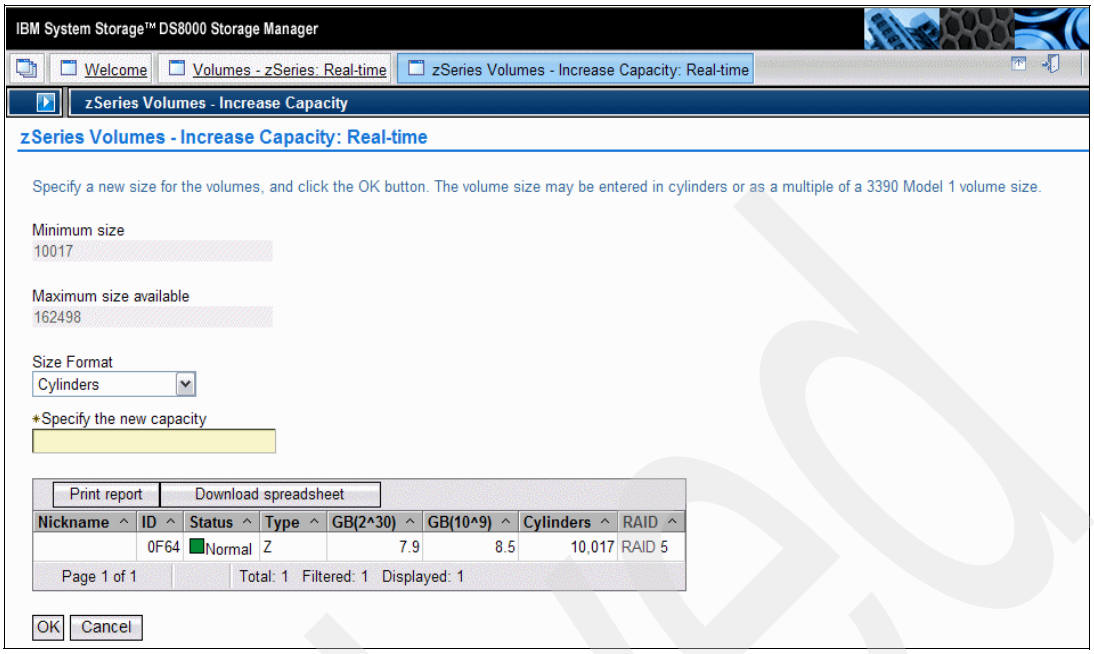


Figure 4-13 Panel to specify the new volume capacity

Thus, if 75000 cylinders is specified on the panel shown in Figure 4-13, the message displayed in Figure 4-14 on page 84 is issued when you click **OK**.

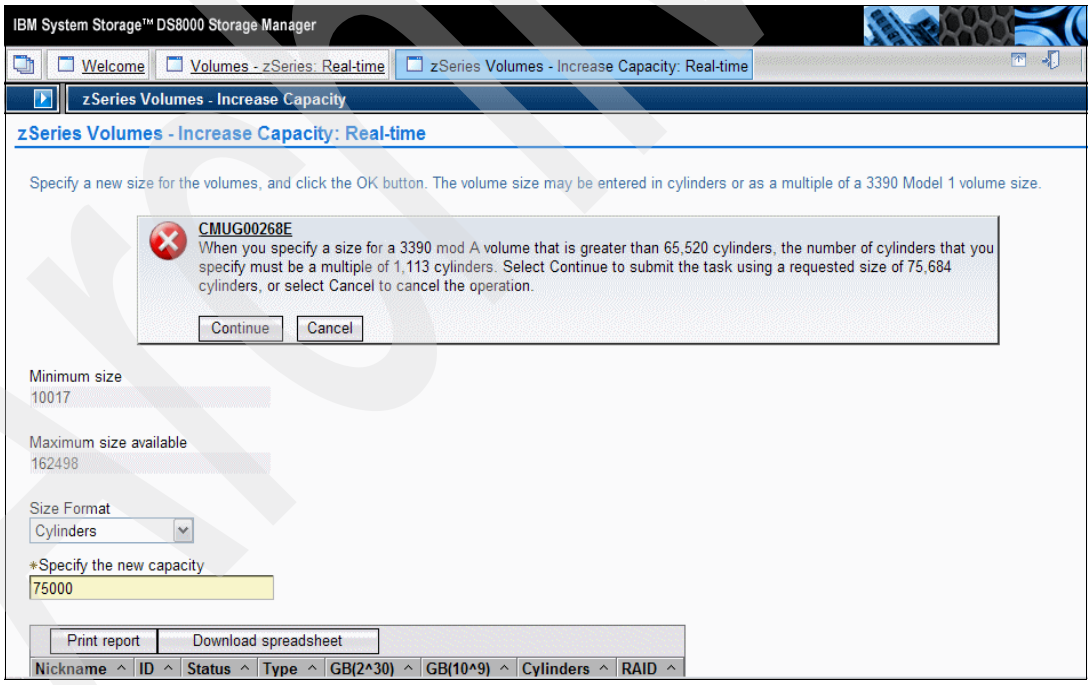


Figure 4-14 Message showing the actual number of cylinders required

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.

4.7.4 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.

With z/OS V1R10: 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.

VTOC index changes

The index block size increased from 2048 bytes to 8192 bytes for devices with cylinder-managed space, as follows:

- Contents of index map records increased proportionally (more bits per record, offsets changed)
  - Extended header in VIXM for free space statistics.
  - VPSM, track-managed space. Small (tracks and cylinders) and large (cylinders) unit map. Same but more bits.
  - VPSM, 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.
- Block size is in format-1 DSCB for the VTOC index and returned by DEVTYPE macro.
- Default index size computed by ICKDSF; it might not be 15 tracks.

This new block size is recorded in the format-1 DSCB for the index. It is necessary to allow for scaling to the largest-sized volumes.

The VTOC index space 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, as follows:
  - LSPACE like free space statistics
- The VIXM 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.

Note: If the VTOC index size is omitted when formatting a volume with ICKDSF and does not preallocate the index, the default before this release has been 15 tracks.

In EAV Release 1, that is, starting with z/OS V1R10, the default size for EAV and non-EAVs is calculated and might be different from earlier releases.

The small unit map (for tracks) describes only the tracks in the first 65520 cylinders (the track-managed space).

There are two large unit maps (for cylinders):

- The existing one is for the first 65520 cylinders.
- A new one is for cylinders after the first 65,520 cylinders (the cylinder-managed space). In the new map, each bit represents 21 cylinders instead of each bit representing one cylinder.
- A new field defines 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.

4.7.5 VTOC rebuild with z/OS V1R11

With z/OS V1R11, when a volume is increased in size, this is detected by the system which then performs an automatic VTOC and index rebuild.

- The system is informed by state change interrupts (SCIs), as controlled with new the DEVSUPxx parmlib member options as follows:

  REFVTOC=ENABLE This enables the automatic REFVTOC function of the device manager. With the REFVTOC function enabled, when a volume expansion is detected, the Device Manager causes the volume VTOC to be rebuilt. This allows the newly added space on the volume to be used by the system.

  REFVTOC=DISABLE This is the default value. It disables the automatic REFVTOC function of the device manager. With the REFVTOC function disabled, when a volume expansion is detected, the following message is issued:

  ```
  IEA019I dev, volser, VOLUME CAPACITY CHANGE,OLD=xxxxxxxx,NEW=yyyyyyyy.
  ```

  - The VTOC is not rebuilt. An ICKDSF batch job must be submitted to rebuild the VTOC before the newly added space on the volume can be used, as is the case for z/OS V1R10. Invoke ICKDSF with REFORMAT/REFVTOC to update the VTOC and index to reflect the real device capacity.

Note: The refresh of the index occurs under protection of an exclusive SYSTEMS ENQ macro for major name SYSZDMO, minor name DMO.REFVTOC.VOLSER.volser.

4.7.6 New format DSCBs for EAVs

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-9 DSCB

The format-9 DSCB is new as of V1R10. It has all the information z/OS needs to record the attributes of a data set in the EAS of an EAV. The format-9 DSCB can point to one or more format-3 DSCBs, as depicted in Figure 4-15 on page 89.

Thus, we see that format-8 DSCB exists strictly to highlight the fact that a format-9 DSCB with all the EAS information has been inserted between the format-1 (also known as format-8) DSCB and the format-3 DSCB chain.

Notes:

The logical DSCB chain today for a data set today is a format-1 and up to 10 possible format-3 DSCBs.

The logical DSCB chain for an EAS-eligible data set on an EAV is a format-8 and one or more format-9s, and up to 10 possible format-3 DSCBs.

In both cases the chain pointer in each DSCB points to the next, if one exists.

The format-9 DSCB is a place for additional attribute information. It contains direct pointers to each possible format-3 DSCB. With this new service in the system, (OBTAIN, CVAFDIR) can read the entire logical DSCB chain for a data set in one call. There are no more loops to read DSCBs until the chain pointer is zero (0).

With z/OS V1R11: Data set attributes are recorded in the format-9 DSCB. These fields indicate the job and step name and time since midnight that the data set was created.

In future releases, additional format-9 DSCBs might be chained between the subtype 1 and any format-3 DSCBs.

The format 9 DSCB exists only for EAS-eligible data sets (VSAM (V1R10), and extended-format sequential data sets (V1R11)). It contains the following EAV information:

- The format identifier is x'F9'.
- A subtype field.
- In the first EAV release, the subtype is 1.
- In future releases, additional subtypes might be added.
- Track addresses which point directly to up to ten format-3 DSCBs.
- All the format-3 DSCBs can be read with one channel program.
- A 20-byte field, DS9ATRV1, that IBM is reserving for vendors. IBM will not specify or monitor its content.
- A "next DSCB" address points to a possible format-3 DSCB. The format-3 DSCBs continue to be chained.

Format-4 DSCB

The format-4 DSCB contains the following new information for EAVs:

- The number of cylinders on a volume.
- The existing 2-byte field DS4DSCYL in the format-4 DSCB contains 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 contains the number of cylinders on the volume.
- A new allocation unit for cylinders above 65,520.
- A new 2-byte field DS4LCYL contains a code value of x'0010' to indicate that the cylinder-managed space after the first 65,520 cylinders must be allocated in units that are larger than one cylinder.
- This value represents 65,520 cylinders divided by 4095. For a non-EAV, this will be zero (0).
- The new field DS4MCU (minimum allocation unit) contains the number of cylinders that each extent in the cylinder-managed area must be a multiple of. For an EAV, this is 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) is 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 use a new track address format (that is, they may contain 28-bit cylinder numbers).

- The “next DSCB” address (DS1PTRDS) always points to a format-9 DSCB (a new type of DSCB), instead of to a possible first format-3 DSCB.

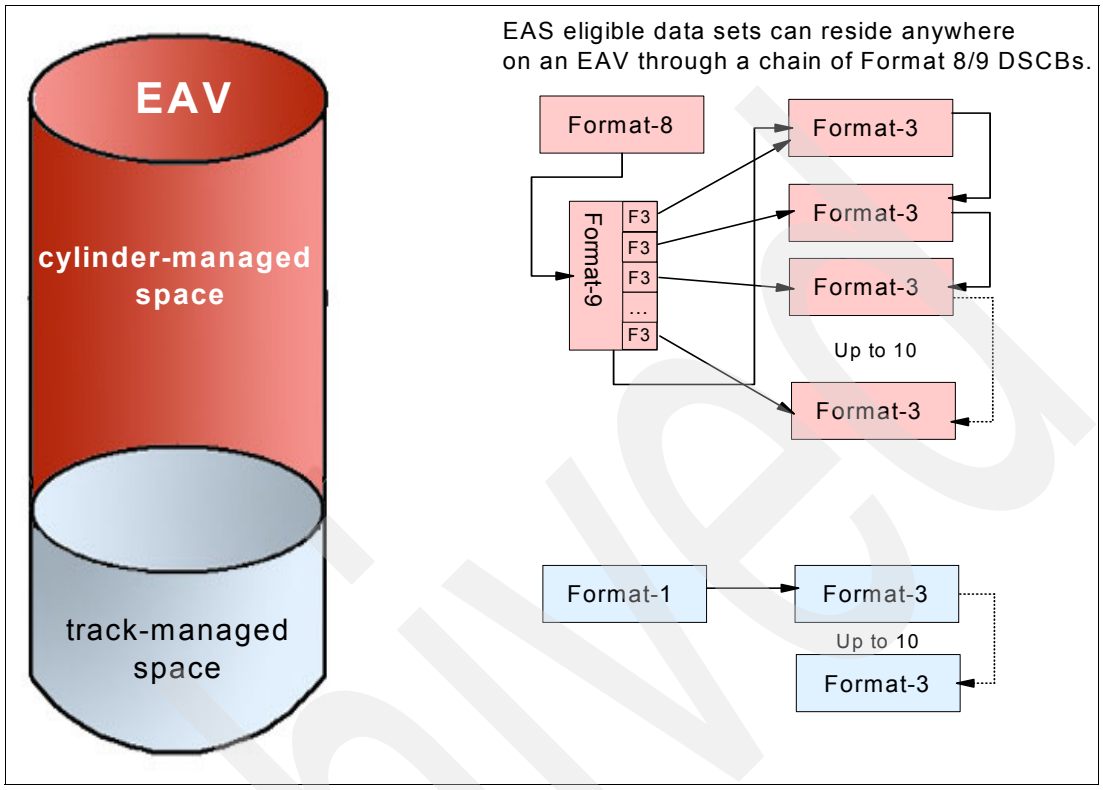


Figure 4-15 Access to EAS-eligible data sets on EAV

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 the EADSCB=OK keyword, the invoking program is indicating to the system service that it understands extended attribute DSCBs and the 28-bit cylinder numbers that can be present in the data set’s extent descriptors.

EADSCB=OK keyword

This keyword specifies whether this program supports data sets with format-8 and format-9 DSCBs. Such data sets can appear on extended address volumes.

Attention: Code EADSCB=NOTOK when your program does not support data sets that have format-8 and format-9 DSCBs. The extent descriptors in DSCBs for a data set described with these formats may have track addresses that contain cylinder addresses 65,520 or larger.

EADSCB=OK is accepted for data sets described by all DSCB types, including format-1 DSCBs, regardless of the volume size where the data set resides. Your program can also run on an older level of the system that does not support this keyword.

In these cases, EADSCB=OK is ignored. EADSCB=OK sets byte 2 bit 4 in the OBTAIN parameter list to ON.

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.

Attention: Not specifying the EADSCB=OK keyword 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 an application supports EADSCB=OK. Specify 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=OK but can be assembled with EADSCB=OK on z/OS V1R10 and above systems and run on downlevel releases, where EADSCB=OK has 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, various vendor products used fields in the format-1 DSCB that IBM reserved for IBM use.

A 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 enforce content rules here. For this field, use the format illustrated in Figure 4-16 on page 90.

```
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 4-16 Format-9 DSCB vendor fields

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

4.8 z/OS V1R11 enhancements for EAVs

EAS-eligible data sets that are added in z/OS V1R11 include extended-format sequential data sets that are SMS-managed. Extended-format sequential data sets, for most purposes, have the same characteristics as sequential data sets. However, records are not necessarily stored in the same format or order as they appear. You can refer to an extended-format data set as a “striped data set” if its data is interleaved across multiple volumes. This is called *sequential data striping*.

Large data sets with high I/O activity are the best candidates for striped data sets. Data sets defined as extended-format sequential must be accessed using BSAM or QSAM, and not EXCP or BDAM.

Extended-format sequential data sets

The following characteristics describe extended-format sequential data sets:

- Extended-format sequential data sets have a maximum of 123 extents on each volume. (Sequential data sets have a maximum of 16 extents on each volume.)
- Each extended-format sequential data set can have a maximum of 59 volumes. Therefore, an extended-format sequential data set can have a maximum of 7257 extents (123 times 59).
- An extended-format data set can occupy any number of tracks. On a volume that has more than 65,535 tracks, a sequential data set cannot occupy more than 65,535 tracks.
- An extended-format, striped sequential data set can contain up to 4 GB blocks. The maximum size of each block is 32,760 bytes.

This new support for extended format sequential data sets includes the following:

- For all data set types a new data set attribute, EATTR, has been added to allow a user to control whether a data set can have extended attribute DSCBs and thus control whether it can be allocated in the EAS. This data set attribute is described in more detail in “New EATTR attribute in z/OS V1R11” on page 91 and “IEHLIST LISTVTOC support” on page 95.
  - There are many ways to define this attribute as described in “Other methods for EATTR specification” on page 92.
- Allocated extents can start in track-managed space and end in cylinder-managed space. One large extent can describe an entire EAV except the VTOC and index. Allocation may occur with fewer extents than before and may allow space allocation where it would not have succeeded before.
- Coexistence support in z/OS V1R10 is provided.
- New format-9 DSCB attributes include the following changes:
  - Job and step name used to create data set
  - Time since midnight (in microseconds) that the data set was created
  - Maximum EAV size remains 223 GB (where 1 GB=1,000,000,000 bytes)

4.8.1 New EATTR attribute in z/OS V1R11

EAS-eligible data sets are defined to be those that can be allocated in the extended addressing space and have extended attributes. This is sometimes referred to as *cylinder-managed space*.

For all EAS-eligible data sets a new data set attribute, EATTR, has been added in z/OS V1R11 to allow a user to control whether a data set can have extended attribute DSCBs and thus control whether it can be allocated in EAS.

DFSMShsm checks the data set level attribute EATTR when performing non-SMS volume selection. The EATTR data set level attribute specifies whether a data set can have extended attributes (Format 8 and 9 DSCBs) and optionally reside in EAS on an extended address volume (EAV). Valid value for the EATTR are NO and OPT.

For more information about the EATTR attribute, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

4.8.2 Defining the EATTR attribute

Use the EATTR parameter to indicate whether the data set can support extended attributes (format-8 and format-9 DSCBs) or not. To create such data sets, you can include extended address volumes (EAVs) in specific storage groups or specify an EAV on the request or direct the allocation to an esoteric containing EAV devices. By definition, a data set with extended attributes can reside in the extended address space (EAS) on an extended address volume (EAV). The EATTR parameter can be used as follows:

- This parameter can be specified for non-VSAM data sets as well as for VSAM data sets.
- The EATTR value has no affect for DISP=OLD processing, even for programs that might open a data set for OUTPUT, INOUT, or OUTIN processing.
- The value on the EATTR parameter is used for requests when the data set is newly created.

Subparameter definitions in JCL

Use the EATTR JCL keyword, EATTR=OPT, to specify that the data set can have extended attribute DSCBs and can optionally reside in EAS.

EATTR = OPT Extended attributes are optional. The data set can have extended attributes and reside in EAS. This is the default value for VSAM data sets if EATTR(OPT) is not specified.

EATTR = NO No extended attributes. The data set cannot have extended attributes (format-8 and format-9 DSCBs) or reside in EAS. This is the default value for non-VSAM data sets. This is equivalent to specifying EATTR=NO on the JCL and is applicable to all data set types.

The DD statement, shown in Figure 4-17, defines a new partitioned data set. The system allocates 10000 cylinders to the data set, of which one hundred 256-byte records are for a directory. When the CONTIG sub-parameter is coded, the system allocates 10 contiguous cylinders on the volume.

```
//DD2 DD DSNAME=PDS12,DISP=(,KEEP),UNIT=SYSALLDA,
//       VOLUME=SER=25143,SPACE=(CYL,(10000,,100),,CONTIG),
//       EATTR=OPT
```

Figure 4-17 JCL example with the EATTR parameter

4.8.3 Other methods for EATTR specification

Different BCP components have to be adapted to with the EATTR attribute introduced with z/OS V1R11. This attribute is specifiable using any of the following methods:

- **AMS DEFINE CLUSTER** and **ALLOCATE**

  On the **ALLOCATE** or **DEFINE CLUSTER** commands, allocate new data sets with parameter EATTR(OPT) to specify that the data set can have extended attribute DSCBs (format-8 and format-9) and can optionally reside in EAS. This is the default behavior for VSAM data sets if EATTR(OPT) is not specified. For non-VSAM data sets, the default is that the data set cannot have extended attribute DSCBs and optionally reside in EAS. This is equivalent to specifying EATTR(NO) on the command.

- ISMF data class panel

  Use the EATTR attribute, on the ISMF Data Class Define panel, to define a data set that can have extended attribute DSCBs and optionally reside in EAS.

  Note: The EATTR specification is recorded in the format-1 or format-8 DSCBs for all data set types and volume types and is recorded in the VVDS for VSAM cluster names. EATTR is listed by IEHLIST, ISPF, ISMF, LISTCAT, and the catalog search interface (CSI).

- TSO/E support

  TSO/E support for data sets that reside in the extended addressing space, including those that reside in EAS, in the **ALLOCATE**, **TRANSMIT**, **RECEIVE**, **SUBMIT**, **RACONVRT**, and **LISTDSI** commands. This support is planned to help ease space constraints by allowing you to move additional data to larger volumes and help simplify storage management by allowing you to store data on a smaller number of volumes. For more information, see 4.11, “TSO/E support for extended address volumes” on page 115.

- Language Environment

  Language Environment XL C/C++ Run-Time Library support for processing extended-format sequential data sets that have extended attributes, including those data sets that reside in the extended addressing space.

- RACF support

  RACF support for discrete profiles for non-VSAM data sets that have extended attributes, including those data sets that reside in the extended addressing space, in addition to the support available now for VSAM data sets.

  - The **ADDSD** and **ALTDSD** commands have been updated to support data sets in the EAS.
  - The RACF database is not supported if allocated in the EAS.
  - RACF utilities do not support allocation of database data sets in the EAS. EATTR=NO (default) is to be used on allocations for the following data sets:
    - IRRUT200 SYSUT1 DD
    - IRRMIN00 SYSRACF DD
    - IRRUT400 OUTDD DD

- Communications Server

  z/OS Communications Server FTP support for reading from and writing to sequential extended-format data sets in the extended addressing space (EAS) of an extended address volume (EAV) and the **QDISK** command.

- Binder support

  Binder support for input and SYSPRINT data sets having extended attributes, including those that reside in EAS.

- ISPF EATTR specification

  There is new support in ISPF for displaying and setting the EATTR attribute for data sets that can reside on extended address volumes (EAVs). ISPF has added support for the display of extended address volumes (EAV) with the specification of a data set level attribute, EATTR, using the allocation panel, Option 3.2, as shown in Figure 4-18.

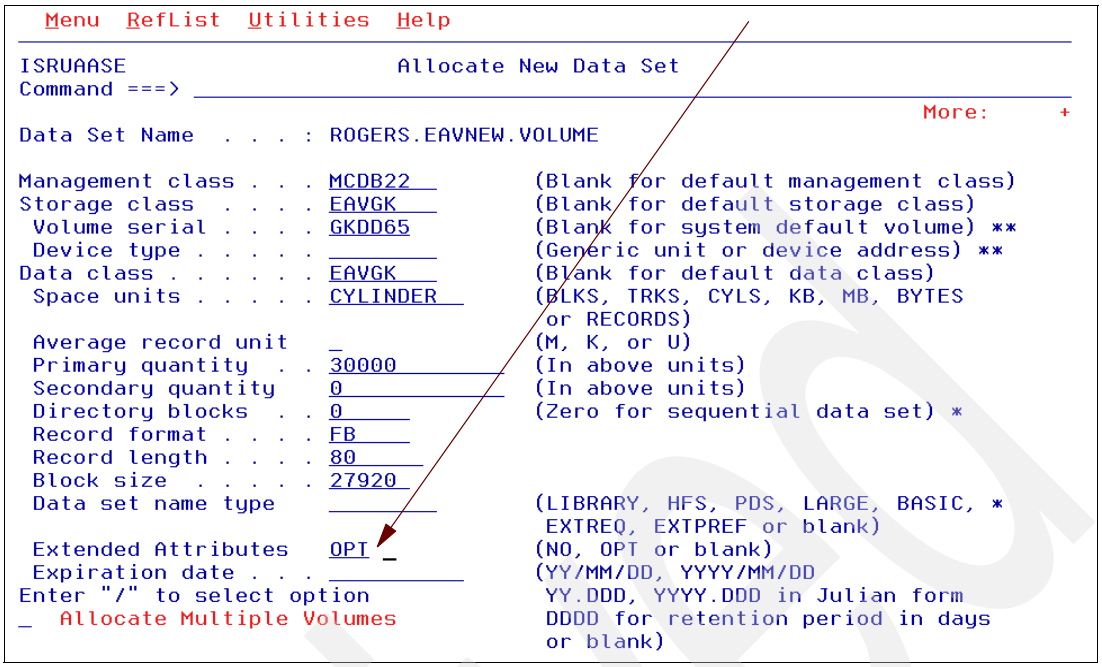


Figure 4-18 ISPF Option 3.2 allocation using EATTR

- Service aids enhancements

  As detailed in 9.6, “Service aids support of extended address volumes” on page 229, service aids have been enhanced for z/OS V1R11, such as:

  – AMASPZAP support for SYSIN and SYSPRINT data sets that reside in the extended addressing space, including those that reside in EAS.

  – SVC dump support for dump data sets that reside in the extended addressing space, including SYSMDUMPs and transaction dumps. Also, IPCS, SNAP and ABDUMP services support is planned for the placement of dump data sets having extended attributes, including those that reside in EAS.

4.8.4 Determine use of EATTR

EATTR use is determined in the JCL interface by merging EATTR from the JCL, LIKE=, or data class, in that order.

Note: The EATTR value recorded for a data set type that is not supported as being EAS-eligible will have no effect until a future time when the system might begin supporting that data set type for EAS.

Volume selection and EATTR

Volume selection is based on EATTR values:

- SMS volume selection
- DFSMShsm non-SMS volume selection

Programs can read DSCBs by issuing any of the following; OBTAIN, CVAFDIR or CVAFFILT macros or by reading a VTOC. In a format-1 or format-8 DSCB, the EATTR value is recorded in two bits at offset 6,1 as shown in Figure 4-19.

```
DS1FLAG1
.... ..00  EATTR not specified.
.... ..01  DS1EATTR_NO   EATTR=NO.
.... ..10  DS1EATTR_OPT  EATTR=OPT.
```

Figure 4-19 EATTR in a format-1 or format-8 DSCB

4.8.5 IEHLIST LISTVTOC support

The IEHLIST LISTVTOC report has changed, as shown in Figure 4-20 on page 95. Update programs or procedures that depend on the LISTVTOC output. The report has the following changes:

- For data sets that support extended attribute DSCBs, the extent descriptions are adjusted to support larger cylinder numbers.
- For volumes that support cylinder-managed space, the format-4 DSCB display identifies the size of the volume, the location of the address of the cylinder where cylinder-managed space begins, and the allocation size of multiple cylinder unit that the system uses to allocate space in cylinder-managed space.
- At the end of the report there is a listing of the number of empty cylinders and additional empty tracks for the track-managed space of volumes that support cylinder-managed space.
- In the VPSM report of free extents, the TRK ADDR and FULL CYLS columns are adjusted to support larger numbers for the track address and full cylinders when the volume is an EAV.

```
 CONTENTS OF VTOC ON VOL 1P9802 <THIS IS AN SMS MANAGED VOLUME>
 FORMAT 4 DSCB NO AVAIL/MAX DSCB /MAX DIRECT  NO AVAIL  NEXT ALT   FORMAT 6  LAST FMT 1     VTOC EXTENT        THIS DSCB
           VI   DSCBS   PER TRK  BLK PER TRK  ALT TRK   TRK(C-H)   (C-H-R)   DSCB(C-H-R)/LOW(C-H) HIGH(C-H)    (C-H-R)
           81   65499       50         45         0         0  0          1279  14  50    0   1  1279  14     0   1   1
          NUMBER OF   MULTICYLINDER UNITS
          CYLINDERS   FIRST CYL ADDR  SPACE
             262668          65520      21
---------------DATA SET NAME----------------  SER NO  SEQNO  DATE.CRE  DATE.EXP  DATE.REF  EXT DSORG RECFM OPTCD BLKSIZE
BRS8AM02.HANDLIN.V1P9802.NVSAM.FILL.CB1       1P9802      1  2008.176    00.000    00.000    1  PS   FB     00     6320
SMS.IND   LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC    EXTEND       LAST BLK(T-R-L)  DIR.REM  F2 OR F3(C-H-R)  DSCB(C-H-R)
S           80           TRKS           1                    0   0 58786                                  0   2  29
            EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                      0 50901  9    50901 14
                                ----ON THE ABOVE DATA SET,THERE ARE          6 EMPTY TRACK(S)
---------------DATA SET NAME----------------  SER NO  SEQNO  DATE.CRE  DATE.EXP  DATE.REF  EXT DSORG RECFM OPTCD BLKSIZE
BRS8AM02.HANDLIN.V1P9802.VSAM.FILL.D1.DATA    1P9802      1  2008.176    00.000    00.000    1  VS   U      80     4096
SMS.IND   LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC    EXTEND       LAST BLK(T-R-L)  DIR.REM  PTR TO F3(C-H-R)  DSCB(C-H-R
S           0            CYLS           0                                                                  2   3  13
                                                      VENDOR
EATTR   JOB        STEP      CREATE TIME      CODE  DATA
OPT     JOB1GALA   STEP0001  01:28:58.673275     1  X'11223344'    2  X'556677'  255  X'88'
EXTENTS  NO       LOW(C-H)      HIGH(C-H)
          0     193704  0      193724 14
                                ----UNABLE TO CALCULATE EMPTY SPACE.
VPSM              A = NUMBER OF TRKS IN ADDITION TO FULL CYLS IN THE EXTENT
            TRK       FULL             TRK       FULL             TRK       FULL             TRK       FULL
           ADDR       CYLS  A          ADDR       CYLS  A          ADDR       CYLS  A          ADDR       CYLS  A
         3784725         42  0       3785670         63  0       3786930         21  0       3787560         42  0
         3788505         63  0       3789765         21  0       3790395         42  0       3791340         63  0

 THERE ARE     52290 EMPTY CYLINDERS PLUS  4590 EMPTY TRACKS ON THIS VOLUME
 THERE ARE 12852 EMPTY CYLINDERS PLUS  4590  EMPTY TRACKS FROM THE TRACK-MANAGED SPACE
 THERE ARE  957221 BLANK DSCBS IN THE VTOC ON THIS VOLUME
New fields and extent descriptors for data and free space adjusted to support larger cylinder numbers
```

Figure 4-20 IEHLIST LISTVTOC report

Note: See *z/OS DFSMSdfp Utilities*, SC26-7414, for complete examples of IEHLIST VTOC listings.

FORMAT option

With the FORMAT option for data sets that support extended attribute DSCBs, the extent descriptions are adjusted to support larger cylinder numbers. With the DUMP option, format-9 DSCBs are displayed.

LISTCAT support

Similarly, LISTCAT reports have been enhanced to reflect the presence of the EATTR attribute; see Figure 4-21.

```
LISTCAT ENT(CFDUPLEX.AA.CARDHOLD) ALL
CLUSTER ------- CFDUPLEX.AA.CARDHOLD
     IN-CAT --- W93UCAT.TVS
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------2009.058
       RELEASE----------------2     EXPIRATION------0000.000
     SMSDATA
       STORAGECLASS ---MLCLAS01     MANAGEMENTCLASS---(NULL)
       DATACLASS ------EATTROPT     LBACKUP ---0000.000.0000
      EATTR--------------(OPT)
       BWO STATUS------00000000     BWO TIMESTAMP---00000 00:00:00.0
       BWO--------------------NO
```

Figure 4-21 EATTR in LISTCAT

Figure 4-22 displays a LISTCAT report with EXTENDED format.

```
 LISTC ENT('ROGERS.TEST.EAV') ALL
NONVSAM ------- ROGERS.TEST.EAV
     IN-CAT --- UCAT.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------2009.156
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
     SMSDATA
       STORAGECLASS ------EAVGK     MANAGEMENTCLASS---MCDB22
       DATACLASS ---------EAVGK     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------GKDD65     DEVTYPE------X'3010200F'     FSEQN------------------0
ASSOCIATIONS--------(NULL)
     ATTRIBUTES
       STRIPE-COUNT-----------1
       EXTENDED
```

Figure 4-22 LISTCAT showing EXTENDED format data set

Figure 4-23 on page 97 shows an EAV enabled on all systems.

```
D SMS,SG(EAVGK),LISTVOL
IGD002I 13:29:19 DISPLAY SMS 593

STORGRP  TYPE    SYSTEM= 1 2 3 4
EAVGK    POOL            + + + +

VOLUME   UNIT    SYSTEM= 1 2 3 4                        STORGRP NAME
GKDD65   DD65            + + + +                         EAVGK
***************************** LEGEND ******************************
. THE STORAGE GROUP OR VOLUME IS NOT DEFINED TO THE SYSTEM
+ THE STORAGE GROUP OR VOLUME IS ENABLED
- THE STORAGE GROUP OR VOLUME IS DISABLED
* THE STORAGE GROUP OR VOLUME IS QUIESCED
D THE STORAGE GROUP OR VOLUME IS DISABLED FOR NEW ALLOCATIONS ONLY
Q THE STORAGE GROUP OR VOLUME IS QUIESCED FOR NEW ALLOCATIONS ONLY
> THE VOLSER IN UCB IS DIFFERENT FROM THE VOLSER IN CONFIGURATION
SYSTEM  1 = SC63       SYSTEM  2 = SC64       SYSTEM  3 = SC65
SYSTEM  4 = SC70
```

Figure 4-23 Storage group display showing an EAV enabled on all systems

4.8.6 Miscellaneous difference between V1R10 and V1R11

In z/OS V1R10, there is a restriction that a given extent cannot start in track-managed space and end in cylinder-managed space.

In z/OS V1R11, a given extent is now allowed to start in track-managed space and end in cylinder-managed space.

Important: Coexistence support in z/OS V1R10 is required to recognize straddled extents. However, z/OS V1R10 does not support creating them.

If space is fragmented, this might allow space allocation in cases where it was not successful before, for example when a user coded CONTIG, or where six extents are required but only five are allowed.

If the ALX (all extents) value is coded or the MXIG (maximum single extent) value is used in the SPACE parameter on V1R11, the same amount of space can be obtained as in V1R10, but it might have more or fewer extents than before.

4.9 EAV migration assistance tracker

The EAV migration assistance tracker can help you find programs that you might need to change if you want to support extended address volumes (EAV). The EAV migration assistance tracker is an extension of the console ID tracking facility. Programs identified in this phase of migration assistance tracking will continue to fail if the system service is issued for an EAV, if you do not specify the EADSCB=OK keyword for them.

DFSMS provides 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 perform the following tasks:

- 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 is to be changed.

Error detection by the tracker

Each category of errors is explained here:

- Identify interfaces with access to the VTOC that are to 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 might 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

4.9.1 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, by using a SYS1.PARMLIB member, to specify preventing an instance from being recorded. Instances already verified then do not show up in the tracker. The exclusion list filters are the registered tracking information, job name, and program name. Wild-carding (? and *) is allowed for these fields in the exclusion list.

In addition to these services, the tracker allows you, by using 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 client can provide instances of programs recorded in the tracker to IBM.

4.9.2 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 4-26 on page 101.

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 by using 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 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 4-24 on page 100 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 4-24 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 4-25 on page 101.

```
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 4-25 Sample tracker exclusion list in CNIDTR7T parmlib member

Tracking command example

In Figure 4-26, 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 occurs 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 4-26 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 only reports new events.

4.10 Migrating to EAVs

The following migration configurations are described in this section:

- Migrating to an EAV-capable system - VSAM CA size
- Migrating to EAV - SMF records
- Migrating to EAV - finding programs requiring migration
- Migrating to EAV - VVDS
- Migrating to EAV - accessing VTOC
- Migrating to EAV - LSPACE macro usage
- Migration to EAV - DEVTYPE macro usage
- Migrating from EAV - to EAV with z/OS V1R11 (EATTR attribute)
- Indications of EAS-eligible data sets

4.10.1 Migrating to an EAV-capable system - VSAM CA size

All VSAM data sets created in z/OS V1R10 may have different CA sizes from what was 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. The list of CA sizes allows any VSAM data set to be EAS-eligible that otherwise were not eligible. 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 additional volumes that can support cylinder-managed space. Also note that VSAM data sets physically copied from a non-EAV to an EAV can have an incompatible CA and thus are not EAS-eligible. This means that extents for additional space would not use cylinder-managed space. The system can adjust the primary and secondary quantity to select a CA.

APAR II14458

The title of this APAR is “VSAM data set index cisize may change when defined or redefined”. When a VSAM data set or ICF catalog is defined or redefined on z/OS V1R10, the index cisize might be increased. This can occur if the data CA size was in tracks and the amount specified was not 1,3, 5, 7, 9 or 15 tracks.

For example, if the space requested was TRK(2,2), then the allocation amount will be TRK(3,3) on z/OS V1R10. Because the data CA size is now larger, the minimum index cisize might increase. For applications such as IMS or CICS that have (or might have) a static LSR pool definition, a change to the index cisize might cause a data set to no longer open after being redefined.

A new tool is now available to find data sets that have a larger CA size; see the following ftp site:

```
ftp.software.ibm.com
```

The tool is in the directory:

```
/servers/storage/support/software/dfsms/ as INDXCI1O.JCL.CNTL.TRSD.
```

Download this tool in binary mode to a data set with the following attributes:

```
LRECL=1O24,BLKSIZE=6144,RECFM=FB,DSORG=PS.
```

After downloading, un-terse the tool by using either AMATERSE or TRSMAIN. This tool consists of a REXX exec and JCL to execute in batch. Review the comments in the JCL for more information about the tool.

The tool is also provided in A.2, “JCL and documentation for the tool” on page 674 of this book.

4.10.2 Migrating to EAV - SMF records

There are two new SMF flags to indicate that the EADSCB=OK keyword on the DCBE macro might need to be specified only if this data set becomes 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. Upgraded programs continue to operate correctly on downlevel systems.

SMF record types 14 and 15 (non-VSAM data set EOV or close) have been modified as follows:

- A new flag, SMF14EADSCB, indicates whether a program specified the EADSCB=OK keyword on the DCBE macro and also includes the following:
  - A specific 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.
    - Upgrade the program to handle 28-bit cylinder numbers and code EADSCB=OK.
- A new flag, SMF14EXCPBAM, indicates that a program used a non-EXCP OPEN DCB (BSAM, QSAM, BPAM) and issued EXCP or XDAP.
  - When SMF14EADSCB is off, it identifies programs that might need to be upgraded to handle 28-bit cylinder numbers and have EADSCB=OK coded.

Important: The VTOC cannot be in cylinder-managed space. Opening a VTOC on a volume that supports extended attribute DSCBs requires the EADSCB=OK keyword because the VTOC may contain format-8 and format-9 DSCBs.

Note the following points:

- If the new flag SMF14EXCPBAM is *not* set, then an EXCP or XDAP was not issued when the program was opened for BAM processing. No changes are required in this case because BAM will handle the 28-bit cylinder numbers internally in a future release.
- If this new flag is set, then an EXCP or XDAP was specified when the program 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.

Other changed SMF records

Other SMF records have also been changed:

- SMF Type 19 record (volume statistics)
  - LSPACE statistics recorded support is expanded with track-managed free space statistics and total volume/track-managed space sizes.

- SMF Type 60, 61, 64, 65, 66 records (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 is saved.

4.10.3 Migrating to EAV - finding programs requiring migration

The simplest way to handle 28-bit cylinder numbers is to use the TRKADDR macro. It works equally well with track addresses that contain 28-bit and 16-bit cylinder numbers, and with any DASD data set. It also works correctly on systems prior to EAV support in z/OS V1R10 and z/OS V1R11.

Switching to use the TRKADDR macro is 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 when:

1. The access method is BSAM, BPAM or QSAM.
2. The data set has a format-8 DSCB.

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

Programs that issue EXCP, EXCPVR, or XDAP

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 can 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.

To facilitate migrating programs that build channel programs, SMF type 14 and 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.

Programs that do not use OPEN and issue EXCP, EXCPVR, or XDAP

Each of these macros requires 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. Instead, programmers must search source code and run tests to find these programs.

Programs that issue an OPEN and issue EXCP, EXCPVR, or XDAP

Two kinds of programs 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.

EXCP OPEN and issue EXCP, EXCPVR, or XDAP

To find instances of these EXCP DCBs, programmers can search source code and also examine SMF type 14 and 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 two-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 contains the data set name unless the OPEN is for a partitioned concatenation.

If the program issues an OPEN for an EXCP DCB, then OPEN can determine whether the program has 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 to open the VTOC and the EADSCB=OK keyword is not coded, a new ABEND and message IEC142I 113-44 is issued. In a future release where other data set types will be EAS-eligible, then this condition will occur for them also.

If the new bit 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 this program is expected to be used with a data set that has a format-8 DSCB in z/OS V1R10 or later, then upgrade the program to handle 28-bit cylinders by coding the EADSCB=OK keyword.

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. It is valid to issue EXCP, but not EXCPVR, in this case. An EXCPVR results in ABEND400.

Search source code for those instances of EXCP and XDAP. In addition, the following data in SMF is recorded:

- A new SMF14/15 flag, SMF14EXCPBAM, is defined to help find programs that issue a Non-EXCP OPEN and issue EXCP or XDAP. This flag is set on when the access method of BSAM, QSAM or BPAM was used and the user program issued 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 the EADSCB=OK keyword on the DCBE macro. Upgrade the program to handle 28-bit cylinders and code the EADSCB=OK keyword.

Changing programs that issue EXCP, EXCPVR or XDAP

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

In a future release, issuances of EXCP or XDAP are expected to be failed for a non-VSAM data set when the access method is BSAM, BPAM or QSAM, and the data set has a format-8 DSCB. Use the techniques described in the prior sections to find and upgrade these programs and add the EADSCB=OK keyword option to the DCBE macro. The upgraded program continues to operate correctly on downlevel systems.

4.10.4 Migrating to EAV - VVDS

The VVDS (VSAM volume data set), which is a catalog volume data set, contains extent descriptors for VSAM data sets. For data sets that are eligible to be allocated or extended on an EAV, its extent descriptors may contain 28-bit cylinder addresses. Prior to EAV support, these extents are stored in an internal VVDS structure called a VVR (VVDS record). Some programs may access the VVDS even though no interfaces are provided for this purpose.

Important: You need to review, and possibly modify, these programs that refer to the extents within the VVR to ensure that they can handle the 28-bit cylinder addresses.

VVDS description

A VVDS contains the following extent descriptors for VSAM data sets:

- The VVR (VVDS record) may contain 28-bit cylinder numbers.
- No interfaces are provided to access the VVDS directly. Programs that access the extents in the VVR record must ensure that they support 28-bit cylinder numbers.

  Use the TRKADDR macro or the IECTRKAD routine.

4.10.5 Migrating to EAV - finding affected programs

An existing product or program might be affected by EAV support if it issues any of the services described in this section. If a program opens an EAV VTOC, it must specify EADSCB=OK on the DCBE macro. This is the same as for EAV usage in V1R10, but now also applies to extended format sequential data sets. The following list, although not comprehensive, specifies the processing needed for data sets that are EAS-eligible with z/OS V1R11 enhancements:

- When issuing an OBTAIN macro to read a DSCB for an EAS-eligible data set or issuing a macro whose name begins with “CVAF” for an EAS-eligible data set, a new EADSCB=OK keyword option must be coded. For performance reasons, you can also exploit a new option to read all the DSCBs for a specified data set with one call.
- If a program bypasses OPEN, almost any channel program issued for an EAS-eligible data set is affected. This includes both building it and monitoring it.
- Use a track address for an EAS-eligible data set such as those in IOSEEK in the IOSB, IOBSEEK in the IOB, or DS1EXT1 in the DSCB. As in the first release of EAV, these control blocks contain 28-bit cylinder numbers. The TRKADDR macro can be used z/OS V1R10 to assist with these manipulations. Use the BBCCHHR field in the I/O error text returned by the SYNADAF macro. This is in EBCDIC form. It is similar to text returned by VSAM.
- Calculate the size of an EAS-eligible VSAM or non-VSAM data set from the cylinder and track numbers of its extents. These cylinder and track numbers might be from a VTOC or DEB, or from an access method internal control block. The TRKADDR macro or IECTRKAD routine can assist with this calculation.
- Examine programs that read VTOCs or DSCBs. In V1R10, programs designed to read an EAV VTOC expect to see format-8 DSCBs for only VSAM data sets. In V1R11, those programs can also see format-8 DSCBs for extended-format sequential data sets.

  If EAVs are shared between V1R10 and V1R11, avoid opening non-VSAM data sets that have format-8 DSCBs or preventthose data sets from being on those volumes.

  Programs can read VTOCs or individual DSCBs with BSAM, QSAM, EXCP, OBTAIN, CVAFDIR, and CVAFFILT.

- Examine those VTOC-reading programs to see whether they might be affected by seeing a format-8 DSCB when they expected a format-1 DSCB, or might be affected by seeing a format-9 DSCB when they expected a format-3 DSCB.
- Examine programs that calculate the size of a data set on a volume. The value might be larger than the program has ever seen. The TRKADDR macro can assist with these calculations.
- Although there are no intended programming interfaces for channel programs with extended-format data sets, any such programs must take the 28-bit cylinder numbers into account as in V1R10. 28-bit cylinder numbers might be in a DSCB, IOB, or channel program. The TRKADDR macro can assist with manipulating track addresses.

Important: If a non-VSAM data set is created on V1R11 on an EAV and you want to use it on an earlier release, avoid using EATTR=OPT for that data set. If the non-VSAM data set has a format-8 DSCB, it can be opened on a release before V1R11. If the program issues the OBTAIN, CVAFDIR, or CVAFFILT macro for a data set that has a format-8 DSCB, then the macro must have EADSCB=OK.

4.10.6 Migrating to EAV - accessing VTOC

With z/OS V1R11, access to VTOC records has been enhanced to allow a program to read the logical DSCB chain for a data set in one invocation of a service. The OBTAIN and CAMLST routines have a new option to specify the number of DSCBs that might be read.

You can use either DADSM or common VTOC access facility (CVAF) macros to access a VTOC and its index.

The DADSM macro includes:

- LSPACE, which provides information about 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, which reads one or more DSCBs from the VTOC.
- PARTREL, which releases unused space from a sequential or partitioned data set or a PDSE.
- REALLOC, which allocates DASD space.

If you specify a data set name using OBTAIN and the CAMLST SEARCH option, the OBTAIN routine reads the 96-byte data portion of the format-1 DSCB and the absolute track address of the DSCB into virtual storage. The absolute track address is a 5-byte field in the form CCHHR that contains zeros VIO data sets.

Accessing DSCBs on EAVs

To read one 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= keyword (which is a new keyword with z/OS V1R10 and z/OS V1R11) 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.

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 can have 28-bit cylinder track addresses. Use the TRKADDR macro or IECTRKAD service to manipulate 16-bit or 28-bit cylinder track addresses.

MULTIPLEDSCB parameter

Specifying MULTIPLEDSCB=YES indicates that the calling program requests to read and write multiple DSCBs to and from a buffer list that contains more than one buffer list entry.

- This flag resolves to a new indicator in the CVPL, CV4MULTD, to be set on.
- Multiple DSCB processing for reads and writes is requested by specifying the MULTIPLEDSCB=YES keyword and providing a buffer list that contains more than one buffer list entry (BFLHNOE>1).

Specifying MULTIPLEDSCB=NO indicates that the calling program requests that only one DSCB is to be processed. This is the default for MF=L and MF=I forms of the CVAFDIR macro

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

OBTAIN macro

OBTAIN search processing stores an additional two bytes in the caller's return area right after the 101 bytes that is set by the OBTAIN macro on prior releases. Ensure that programs provide the minimum 140-byte return area. These two bytes are set to the total number of consecutive 140-bytes areas that are needed to read all the DSCBs for the data set.

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 a work area called wkarea_relexp. The system treats a value of 0 as a 1.

Prior to z/OS V1R11, the system did not support a chain of more than 12 DSCBs for one data set, but it is valid to provide an area that is longer than currently needed. The system verifies that the provided area is valid. When an area is provided that is long enough to contain more than one DSCB, obtain processing returns DSCBs for the requested data set name in logical VTOC order until all the 140-byte return areas are used.

The logical VTOC order is a format-1 DSCB, followed by zero or more format-3 DSCBs; or a format-8 DSCB, followed by one or more format-9 DSCBs, followed by zero or more format-3 DSCBs. No absolute maximum number of DSCBs for a data set can be assumed.

For SEARCH requests, the actual number of DSCBs is returned in a field that is located in the first 140-byte return area. For SEEK requests where the target of the seek operation is not a format-1 or format-8 DSCB, the NUMBERDSCB value is treated as though it were 1 and only that single DSCB is returned.

Note: For programs run on an earlier level of the system that does not support this keyword, the NUMBERDSCB value is treated as though it were 1.

CVAFDIR macro

For an indexed or non-indexed VTOC, you can use the CVAFDIR macro to perform the following functions:

- Read or write one or more DSCBs by specifying the name of the data set they represent
- Read or write one or more DSCBs by specifying their addresses

CVAFDIR provides a new keyword, MULTIPLEDSCB=YES, to indicate to CVAFDIR processing to use the multiple buffers passed in the buffer list.

Multiple DCB reads

Note the following information about multiple DSCB when CVAFDIR reads:

- Only the first buffer list entry, seek or search, argument is used. This provides 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 is set in the format specified by the caller in each buffer list entry. The buffer list argument is indicated as updated with the flag, BFLEAUPD, set on.

  Reading the data set DSCBs is 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) is 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 and 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 is either Format-1 -> Format-3s, or Format-8 -> Format-9s -> Format-3s.
- Buffer list entries other than the first must provide a 140-byte buffer. The first buffer list entry buffer size follows the same rules as today. That is, with the seek option, provide either a 96-byte or 140-byte buffer and 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) 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.

Multiple DSCB writes

Note the following information about MULTIPLEDSCB CVAFDIR writes:

- The buffer list header must indicate the number of buffer list entries passed. Only buffer list entries without the skip flag on are 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 can be a 96-byte buffer if the DSCB to write is a format-1. The same holds 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 can be set for entries where a format-0 DSCB verify before a write is not needed (BFLENVER). This overrides the VERIFY=YES setting.

4.10.7 Migrating to EAV - LSPACE macro usage

The LSPACE macro can be used 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 sub-function, 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, the EXPMSG option can be used to create an expanded message return area that the LSPACE macro needs.

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

The expanded data return area (EXPDATA) returns binary data of free space information for volumes with more than 65,520 cylinders. LSPACE macro can return additional information such as the format-4 DSCB, the total number of free extents on the volume, or the fragmentation index.

LSPACE macro new keywords

Four new keywords are added to the LSPACE macro, namely XEXPMSG, EXPDATA, DATATYPE, and PLISTER. The LSPACE macro returns information about a DASD volume, as follows:

- Returned information can be in character or binary format.
- A new XEXPMSG= keyword does the following:
  - It 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 to obtain free space information in the message return area for volumes that are configured with more than 65,520 cylinders.

    The returned free space includes space for the total volume and space from the track-managed space on a volume. For volumes with 65,520 cylinders or less, both sets of free space information are returned but they are the same.
- A new keyword EXPDATA= keyword does the following:
  - It specifies the address of a caller-provided expanded data return area into which LSPACE returns expanded free space and volume information. Specify this keyword to obtain free space information in the LSPACE data return area for volumes that are configured with more than 65,520 cylinders.

    The returned free space includes space for the total volume and space from the track-managed space on a volume. For volumes with 65,520 cylinders or less, both sets of free space information are returned but they are the same.

- There is a new DATATYPE keyword, as depicted in Figure 4-27.

```
New DATATYPE keyword
   {ALL        }Return all of the following information
       {VOLUME    } Return free space for volume
       {VTOC         } Return free space for VTOC
       {INDEX        } Return free space for index
       {FRAGINDEX}Return the fragmentation index
```

Figure 4-27 LSPACE new DATATYPE keyword

This DATTATYPE keyword is only allowed when the DATA or EXPDATA keyword is specified. Only the information specified is returned to the caller. DATATYPE is valid for both non-EAV and EAV.

This keyword eliminates unnecessary I/O required to retrieve free space information that is not be required by the caller. DATATYPE=ALL is the default, where:

- VOLUME means provide free space information for the VOLUME.
- VTOC means provide free space information related to the VTOC.
- INDEX means provide free space information related to the VTOC INDEX.
- FRAGINDEX means provide the fragmentation index.
- ALL means provide all available LSPACE statistics. This is the default.

- There is also a new PLISTVER keyword to manage the use of the longer LSPACE parameter list.
  - PLISTVER=plistver | IMPLIED_VERSION | MAX

    This keyword defines the version of the LSPACE parameter list that to be generated for the MF=D 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 is 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 1 or larger to be specified, as shown in Figure 4-28 on page 111.

```
VERSION  KEY
  1          MSG
             EXPMSG
             DATA
             SMF
             F4DSCB
  2          XEXPMSG
             EXPDATA
             DATATYPE
```

Figure 4-28 PLISTVER= keyword of LSPACE

Note the following points:

- When MAX is specified, the generated parameter list is the largest size currently supported. This size might grow from release to release, thus possibly affecting the amount of storage needed by programs. If a program can tolerate this, always specify MAX when creating the list form parameter list, because that ensures 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 that 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.

4.10.8 Migrating to EAV - DEVTYPE macro usage

The DEVTYPE macro with the existing INFO=DASD parameter returns the number of cylinders in a 16-byte area provided by the caller. New fields returned include the following:

- An indication that cylinder-managed space exists on the volume
- An indication that extended attribute DSCBs, format-8 and format-9 DSCBs, are allowed on the volume
- The minimum allocation size in cylinders for cylinder-managed space
- The first cylinder address where cylinder-managed space begins
- The block size of the index

The DEVTYPE macro allows you to obtain device characteristic information about I/O devices, as follows:

- The DEVTYPE macro issued without the INFOLIST parameter returns a 2-byte value for the number of cylinders, and is not valid for an EAV.
- The DEVTYPE macro issued with the INFOLIST parameter (INFO=DASD) returns a different format of the device characteristics information. This includes a 4-byte value for the number of cylinders and is mapped to field DVAICYL in shipped mapping macro, IHADVA. The 2-byte field is mapped by DVACYL.

  In addition, it now returns the following information:

  – Multicylinder unit 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

The added DEVTYPE field names are listed and described in Table 4-1.

Table 4-1 DEVTYPE new fields

DVAIXVLD	BIT	DVACYLMG, DVAEADSCB, DVAVIRSZ valid.
DVACYLMG	BIT	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	BIT	Extended attribute DSCBs, format-8 and format-9 DSCBs, are allowed on this volume. Valid when DVAIXVLD is set.

DVAIXVLD	BIT	DVACYLMG, DVAEADSCB, DVAVIRSZ valid.
DVAMCU	8-bit integer	Minimum allocation size in cylinders for cylinder-managed space. Each extent in this space must be a multiple of this value. 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 V1R10 or if the status is not yet known. In these two cases, DVAIXVLD is not set.
DVALCYL	16-bit integer	First cylinder address divided by 4095 where space is managed in multicylinder units. Cylinder-managed space at this address. Valid when DVACYLMG is set. This field is zero on releases before z/OS V1R10 or if the status is not yet known. In these two cases, DVAIXVLD is not set.
DVAVIRSZ	16-bit integer	Block size of the index data set. Valid when DAVIXVLD is on. When valid and zero, the volume has no working VTOC index. This field is zero on releases before z/OS V1R10 or if the status is not yet known. In these cases, DVAIXVLD is not set.

4.10.9 Migrating from EVA - to EAV with z/OS V1R11 (EATTR attribute)

This section explains migration and coexistence considerations for the new EATTR data set attribute. Use care when specifying EATTR=OPT for data sets that are not EAS-eligible in z/OS V1R10, that is, extended-format sequential data sets. These are not allowed to be opened from z/OS V1R10 if they were allocated in z/OS V1R11 with extended attributes.

Also use care when specifying EATTR=OPT for non-VSAM data sets that are not an EAS-eligible data set type in V1R11. You must be certain that any application that accesses any non-VSAM data set with EATTR=OPT (even if not an EAS-eligible data set type in V1R11) can handle extended attributes and 28-bit cylinder addresses because in the future we might allow them to be EAS-eligible.

APARs might be needed to allow DB2 or IMS to support extended-format sequential data sets in EAS.

EAV-capable systems

EAV-capable systems can share EAVs with downlevel systems (z/OS V1R10) until explicit action is taken to begin allowing non-VSAM data sets to reside in the EAS of an EAV.

- This is because EATTR defaults to NO for non-VSAM data sets. EATTR can be specified for non-EAS-eligible data sets.

Important: A non-VSAM data set allocated with format-8 and format-9 DSCBs in z/OS V1R11 cannot be opened on a pre-z/OS V1R11 system.

- There is no affect until data set type is enabled in the system for EAS.
- However, certain applications can handle extended attribute DSCBs and 28-bit cylinder numbers, and they might become EAS-eligible data sets in the future.

Considerations for migrating to z/OS V1R11

Application programs issuing the CVAF and OBTAIN macros with the EADSCB=OK keyword will be able to see format-8 and format-9 DSCBs for types of data set that cannot be opened on that release. Now a format-8 DSCB can be found, whereas none might have been before, even if that release does not support that type of data set for EAS. This change is being made, through PTF, also on z/OS V1R10. Programmers might have expected a failure of CVAF or OBTAIN if the data set has a format-8 DSCB and cannot be opened in the current release. This problem will occur for format-8 DSCBs created on V1R11 and read on V1R10. In general, it is unlikely to be much of an issue.

A single extent can begin in track-managed space and end in cylinder-managed space. Extents of this type will be recognized on z/OS V1R10, but not created there.

This consideration is for programs that expect each extent to begin and end in either track-managed or cylinder-manager space as in V1R10, but not begin in one and end in the other. Maybe the program expected a large space request on an empty volume to be split as on V1R10. If the ALX (all extents) is coded or MXIG (maximum single extent) value is in the SPACE parameter on V1R11, the same amount of space can be obtained as in V1R10 but it might have fewer extents than before.

If space is fragmented, this might allow space allocation in cases where it was not successful before, for example, when the user coded CONTIG, or where six extents were required and only five are allowed. APAR OA26623 on V1R10 causes DADSM to tolerate an extent that straddles track-managed and cylinder-managed space. DADSM with the PTF in V1R10 does not create such an extent.

4.10.10 Indication of EAS-eligible data sets

The following bits in field **DFAFEAT8** can be tested in the DFA, mapped by IHADFA, to learn which data set types can be opened with format-8 DSCBs; see Figure 4-29 on page 114.

```
DFAVSAMFOREAS  EQU X'80'      VSAM enabled for EAS
DFASEQFOREAS   EQU X'40'      Basic, large format seq (QSAM, BSAM
*                             BDAM access) enabled for EAS
DFAPDSEFOREAS  EQU X'20'      PDSE enabled for EAS
DFAPDSFOREAS   EQU X'10'      PDS enabled for EAS
DFADIRFOREAS   EQU X'08'      Direct (BDAM access) enabled for
*                             EAS
DFAEFSEQFOREAS EQU X'04'      Ext format seq enabled for EAS
DFAUNDEFFOREAS EQU X'02'      Undefined DSORGs enbld for EAS
```

Figure 4-29 Indication of EAS-eligible data sets

Important: In V1R10, only the VSAM bit will be on (with a PTF on V1R10).

In V1R11, both the VSAM and extended format sequential bits will be on. (Additional data set types might be eligible in the future.)

4.11 TSO/E support for extended address volumes

Certain SO/E services and commands fail without EAV support. REXX execs and CLISTs can use information found on EAV data sets. In z/OS V1R11, DFSMS added support for extended format sequential data sets to reside in the EAS or extended addressing space, on extended address volumes. This allows data sets to reside on volumes with more than 65,520 addressable cylinders by using a new format of DSCB. Although VSAM data sets were supported for EAV in z/OS V1R10, TSO/E commands and services do not support VSAM data sets.

In z/OS V1R11, data sets can now be allocated using the EATTR keyword on the **ALLOCATE** command. Support for the new **ALLOCATE** command keywords was added by the allocation component, but it works like any other keyword on ALLOCATE. The options are described here:

EATTR(NO) No extended attribute. The data set cannot have extended attributes (format-8 and format-9 DSCBs) or reside in the EAS. This is the default behavior for non-VSAM data sets in z/OS V1R11 and higher, if neither NO nor OPT is specified.

EATTR(OPT) Extended attributes are optional. The data set can have extended attributes and can optionally reside in the EAS. This is the default behavior for VSAM data sets in z/OS V1R11 and higher, if neither NO nor OPT is specified.

TSO/E services and commands

TSO/E services and commands support EAV data sets in z/OS V1R11, as follows:

- Size calculations based on relative track addresses work.
- If OBTAIN is used, the EADSCB=OK keyword option is specified.
- TRANSMIT and ALLOCATE LIKE preserve the EATTR setting.
- There are two new LISTDSI variables, namely SYSEATTR and SYSEADSCB.

Updated services

The following list of services or commands were updated. Assume that those not listed here tolerate EATTR data sets, as well.

- TSO/E I/O services
- SEND parmlib support routine
- ALLOCATE command processor
- EDIT command processor (APAR OA28295)
- SUBMIT I/O and control routine
- PRINTDS allocation/unallocation routine
- LISTDSI output routine
- LISTDSI data set information routine

- LISTDSI data management linkage assist routine
- RACONVRT utility I/O service routine
- TRANSMIT allocate input and read DSCB routine
- TRANSMIT unload a PDS with IEBCOPY routine

4.11.1 Usage and invocation

TSO/E commands or services work normally with EAV data sets, as described here:

- ALLOCATE DA(SAMPLE1) EATTR(OPT) NEW
  - DOC APAR OA28338 documents the new keyword.
- TRANSMIT/RECEIVE uses a new text unit that is transparent to users.
- REXX and CLIST LISTDSI function sets two new variables:
  - SYSEATTR

    This indicates the current status of the EATTR bits in the DSCB that describe the EAS eligibility status of a data set. The EAS can only contain data sets that are EAS-eligible. A default blank indicates that the EATTR bits are '00'b.

    The defaults for EAS eligibility apply:

    - VSAM data sets are EAS-eligible, and can have extended attributes (format-8 and format-9 DSCBs).
    - Non-VSAM data sets are not EAS-eligible, and cannot have extended attributes (format-8 and format-9 DSCBs).

      Blank - This is the default setting. No EATTR specified.

      NO - There are no extended attributes and the data set cannot reside in EAS.

      OPT - The data set can have extended attributes and can reside in EAS.
  - SYSEADSCB

    This indicates whether the data set has extended attributes:

    - YES - The data set has extended attributes (format-8 and format-9 DSCBs) and can reside in the EAS.
    - NO - The data set does not have extended attributes (format-8 and format-9 DSCBs) and cannot reside in the EAS.

Sample REXX code for EATTR

Figure 4-30 shows a sample REXX exec to display the EATTR parameter.

```
/* REXX*/
say 'Enter the parameters for the LISTDSI call'
pull dsname
X = LISTDSI(dsname)
SAY "SYSDSNAME              " SYSDSNAME
SAY "SYSEATTR               " SYSEATTR
SAY "SYSEADSCB              " SYSEADSCB
```

Figure 4-30 Sample REXX displaying EATTR

Figure 4-31 shows the output produced by the REXX code.

```
SYSDSNAME               IBMUSER.SAMPLE1
SYSEATTR                OPT
SYSEADSCB               YES
```

Figure 4-31 Output of the REXX exec

Sample CLIST for EATTR

Figure 4-32 shows a sample CLIST to display the EATTR parameter.

```
PROC 0
WRITE  ENTER THE PARAMETERS FOR THE LISTDSI CALL
READ
LISTDSI &SYSDVAL
WRITE  SYSDSNAME               &SYSDSNAME
WRITE  SYSEATTR                &SYSEATTR
WRITE  SYSEADSCB               &SYSEADSCB
```

Figure 4-32 Sample CLIST displaying EATTR

4.11.2 Migration and coexistence considerations

The new TRANSMIT/RECEIVE text unit does not cause exits to fail. Records with INMEATTR key ‘8028’x can just be ignored. New LISTDSI variables are not defined on earlier releases. If used on an earlier release, the new variables will not be set. For example, the sample REXX exec shown in Figure 4-30 returns the output shown in Figure 4-33.

```
SYSDSNAME               IBMUSER.SAMPLE1
SYSEATTR                SYSEATTR
SYSEADSCB               SYSEADSCB
```

Figure 4-33 Output of the REXX exec on pre-V1R11

4.12 FTP EAV large volume access

z/OS V1R11 provides support in FTP for reading and writing extended format sequential data sets that have been allocated in the extended addressing space (EAS) of an extended address volume (EAV) or are eligible for such an allocation (format-8 DSCB).

FTP will read and write to such data sets in the EAS. In z/OS V1R11, FTP does not provide explicit configuration options to allocate data sets in the EAS or be eligible to reside in the EAS (unless implicitly done by way of SMS data class assignments).

FTP has added support for providing information about EAVs on the **qdisk** command.

Today, if you “screen-scrape” qdisk output, you must modify the screen-scraping logic because the qdisk output had to be changed between z/OS V1R10 and z/OS V1R11 to accommodate EAVs.

FTP adds support for reading and writing to and from existing EAS data sets (but not for creating them (toleration mode).

- FTP to understand format-8 DSCBs

- FTP to use TRKADDR for track calculations
- FTP qdisk option for SITE/LOCSITE output format will change to as shown in Figure 4-34

```
ftp> quote site qdisk
200-        Percent      Free      Free          Largest    Free
200- Volume Free         Cyls      Trks         Cyls-Trks   Exts  Use Attr
200- CPDLB3   45         1507       108         1440    2     22  Storage
200- CPDLB0   44        80486       156          461    0     25  Storage
200- CPDLB1   99        66619         5        65362    5      3  Storage
200 SITE command was accepted
ftp>
```

Figure 4-34 Output of FTP command with EAV support

Service aids enhancements for EAVs

For service aids enhancements for EAVs, see 9.6, “Service aids support of extended address volumes” on page 229.

4.13 Coexistence with pre-z/OS V1R10 systems

This section discusses APARs and PTFs that are available for pre-z/OS V1R10 systems that do not allow an EAV to be online. Note the following implications:

- Coexistence support is not supported.
- EAV use must be planned by applications.
- EAVs that are defined as “shared” must be set offline in HCD.

The following items have considerations related to EAVs and older levels of z/OS that do not support them.

4.13.1 DEVSERV QDASD command

You can use the **DEVSERV QDASD** command to display diagnostic information about the status of DASDs 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 if you have not already done so, so that the **DEVSERV QDASD** command will display the correct number of cylinders on z/OS V1R9.

The **DEVSERV** command has the following changes:

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

Figure 4-35 shows the **DEVSERV QDASD** command for an EAV. It also shows the expanded positions for the CYL numbers, 70119, for EAVs, which was changed in z/OS V1R10.

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

Figure 4-35 DEVSERV QDASD command for an EAV

The following APARs and PTFs must be installed on downlevel releases in support of the new functions available for EAVs.

Note: In z/OS V1R11, or after the PTF for APAR OA25793 is installed on z/OS V1R10 and z/OS V1R9, DEVSERV PATHS and the **QDASD** command display 5-digit device numbers such as 0DC65, as shown in Figure 4-35 on page 119.

Such device numbers consist of the subchannel set number and the normal device number. The blank that was in the first character position no longer exists and is replaced by 0 or 1 (the subchannel set number).

DEVSERV SMS command

You can use the **DEVSERV SMS** or **DEVSERV S** command to display the volume and storage group status of nn devices that SMS manages, starting with the specified device number.

```
DEVSERV SMS,DC65
IGD001I 15:16:18 DEVSERV SMS 720
  UNIT DTYPE  M VOLSER VOLSTAT      STORGRP  SGSTAT
  DC65,3390A ,O,MLDC65,PRIV/RSDNT,  VOLUME NOT MANAGED BY SMS
 ****************************** LEGEND ******************************
 A = ALLOCATED                      F = OFFLINE
 M = MOUNT PENDING                  N = NOT ALLOCATED
 O = ONLINE                         P = PENDING OFFLINE
 Note: Tape device type is not supported by DEVSERV SMS
```

Figure 4-36 DEVSERV command to display volume and storage group status

3390 Model A support

3390 Model A support was included in DS8000 release 3 SPE APAR. This affected one operator command response with the **DEVSERV** command, which displays the following:

- 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 (11/19/2007):
  - UA38011 5695DF1111K0 (z/OS V1R7)
  - UA38012 5695DF111180 (z/OS V1R8)
  - UA38013 5695DF111190 (z/OS V1R9)

4.13.2 LISTDATA PINNED command

The track addresses in the output of the IDCAMS **LISTDATA PINNED** command are in a different format, regardless of the volume size. Prior to z/OS V1R10, the report listed each pinned track and its associated data set name.

Beginning with z/OS V1R10, the report identifies a range of pinned tracks associated with a data set. Each track address is printed in eight hex digits in the native format of the device, using the form CCCCcccH. In addition, instead of one track being listed per line, consecutive tracks for each data set are gathered into one line and the range of track addresses is shown.

Programs that use DASD channel programs and messages that might deal with a track in the cylinder-managed space. You can compare two of these track addresses for equality but you cannot reliably use a simple comparison for greater than or less than. Any arithmetic must take this special format 28-bit cylinder number into consideration.

Use the new TRKADDR macro for all track address comparisons and calculations. This macro is available 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. 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 in z/OS V1R10 and beyond, but programs linked with it can run on downlevel releases.

For programs that use messages, the track addresses in the output of IDCAMS LISTCAT continue to be printed in eight hex digits in the native format of the device, which is the format of CCCCcccH. This means that they can be compared for equality but cannot be compared for greater than or less than unless your program converts them to another format such as with the new TRKADDR macro or IECTRKAD routine. The track addresses in the output of IDCAMS **LISTDATA PINNED** command are presented in a modified report that lists each data set name associated with a range of pinned tracks (previously the report listed each data set name associated with each pinned track address). The pinned track addresses are in the format CCCCcccH.

Note: The modified report format applies to all volumes, not simply EAVs.

LISTDATA PINNED with UNITNUMBER

This version of the command is not allowed, as explained here:

- **LISTDATA PINNED** with **UNITNUMBER** requires APAR OA21487.
- Do not issue **LISTDATA PINNED** with **UNITNUMBER** on pre-z/OS V1R10 systems to an EAV when the storage subsystems have pinned data. In this case, issue the **LISTDATA PINNED** command from a z/OS V1R10 system.
- UA40229 for z/OS V1R9.
- UA40228 for z/OS V1R8.

LSPACE macro

The LSPACE macro changes with APAR OA22449. This APAR adds support to DFSMS z/OS V1R7, V1R8, and V1R9 to allow these releases to map V1R10 LSPACE parameters XEXPMSG and EXPDATA to EXPMSG and DATA respectively, by:

- Treating expanded requests as normal and changing pre-V1R10 LSPACE systems to accept the extended parameter list and not fail it with return code 12, as was done in earlier releases.

- EXPDATA and XEXPMSG become DATA and EXPMSG. This changes pre-V1R10 LSPACE systems to treat an EXPDATA= or XEXPMSG= requests as DATA= or EXPMSG= respectively. The following PTFs are available.
  - z/OS V1R8 UA40220
  - z/OS V1R9 UA40221

ICKDSF support

The **IDCAMS LISTDATA PINNED** and **DEVSERV QDASD** commands need the following APAR to recognize EAVs under z/OS R1V8.0 and R1V9.0:

- ICKDSF R17 support for extended address volumes - PK56092

EREP V3R5 support

The following APAR provides formatting support in detailed edit and System Exception reports for EAV devices, Model A.

- EREP V3R5 support for extended address volumes - IO03548

4.13.3 DFSMShsm support

Pre-V1R10 versions of z/OS DFSMShsm can coexist in a HSMplex with z/OS DFSMShsm V1R10. In this situation, the EAV is inaccessible for pre-V1R10 systems because they will be offline. Data sets that have been migrated or backed up from an EAV to ML1, ML2, and backup volumes (non-EAV) under z/OS DFSMShsm V1R10 will be accessible for pre-V1R10 DFSMShsm systems. This toleration APAR is required to support recall, recover, and recycle for these data sets in pre-V1R10 systems.

If the toleration APAR is not installed, the recall, recover, and recycle of these data sets will fail during pre-V1R10 DFSMShsm processing. AUDIT and RECYCLE are changed to process a data set CDDs with a format-8 DSCB.

DFSMShsm APAR OA22804 provides coexistence support of pre-V1R10 DFSMShsm versions to:

- 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.
- Restore EAS-eligible data sets from a backup or migrated copy on an online volume.
- Enable pre-V1R10 systems to process the new VCC keywords that can be specified in the **management class backup copy technique** field.
- Support recall and recover on V1R9 or earlier of an EAS data set with format-8 DSCB that was migrated or backed up on V1R10:
  - Converts to format-1 DSCBs.
  - Allowed to issue the new ARC0784I message and update the FSR record if format-9 DSCB vendor attributes are lost during the RECALL/RECOVER/ARECOVER function, as shown:

```
ARC0784I EXTENDED ATTRIBUTES FOR DATA SET dsname WERE NOT RETAINED DURING
THE RECALL | RECOVER | ARECOVER
```

Explanation: The data set was recalled, recovered, or ARECOVERed from a migration copy, backup copy, or an aggregate backup successfully. However, JOBNAME, STEPNAME, creation time attributes or vendor attributes from the Format 9 DSCB of the recalled or recovered data set were not retained, because the volume on which it was placed did not support format-8 and format-9 DSCBs. The recall, recovery, or aggregate recovery continues.

Coexistence support

Coexistence is added for pre-V1R10 systems to understand the new MCV records for EAVs. These fields are ported from V1R10 systems to pre-V1R10 systems.

Restriction: If a recall for extent reduction is directed to an EAV, the selection of this request from the CRQ will be restricted in pre-V1R10 systems.

Using EAVs in mixed environments where the volume cannot be brought online to all systems can also result in problems recalling or recovering data sets that belong to a storage class having the guaranteed space attribute.

Important: If you need to recover a data set backup made from an SMS EAV to a lower level system, then prior to the recover perform the following actions:

- If the EAV is still available on the higher level system, then delete or rename the data set there.
- If the EAV is no longer available and a catalog entry exists, then use **`IDCAMS DELETE NOSCRATCH`** to remove the data set from the catalog.

Note that the backup copy *must* reside on a non-EAV and the volume must be online to the lower level system.

Coexistence considerations

Without APAR OA22804, you will notice the following impact if HSM is being used to manage EAVs on the V1R10 system:

- On the downlevel system, if an attempt is being made to recall or recover an EAS data set with a format-8 that was migrated or backed up on z/OS V1R10, the function will fail.
- This will directly impact recall, which is done implicitly. It will also impact recycle and audit.
- VCC keywords specified in the management class will not be processed correctly.

  If VPREF or VREQ is specified in the management class, the data set will be backed up using the standard I/O method without the coexistence APAR.

Important: Apply PTFs UA40303 for (V1R6), UA40304 for (V1R7), UA40305 for (V1R8), and UA40306 for (V1R9).

4.13.4 DFSMSdss restores from EAVs

DFSMSdss releases are modified to restore data from a DFSMSdss dump data set containing data from an EAV, as follows:

- Restore of data sets dumped from an EAV with physical or logical data set is supported.

- Track restore of data dumped from the track-managed space of an EAV using physical dump processing is supported.
- When a data set that was dumped from an EAV that has format-8 and format-9 DSCBs is restored with physical or logical data set processing, and vendor attributes existed in the format-9 DSCB, a MSGADR556W is issued with reason code 1, as shown here.

```
ADR556W (ttt)-,mmmm(yy), EXTENDED ATTRIBUTES FOR DATA SET dsname WERE NOT
       RETAINED DURING {COPY | RESTORE},rsn
```

Explanation: The data set was copied or restored but various extended attributes were lost for the following reasons (rsn):

1 Vendor attributes from the format-9 DSCB of the dumped data set were not retained for the target data set because the volume on which it was placed did not support format-8 and format-9 DSCBs.

2 Vendor attributes from the format-9 DSCB of the dumped or copied data set were not retained for the target data set due to problems updating the target's format-9 DSCB.

Programmer Response: 1. If the extended attributes are desired, rerun the COPY or RESTORE and specify target volumes or an SMS group that supports format-8 and format-9 DSCBs.

2. An error occurred while reading or writing an F9 DSCB. Retry the restore or copy for the data set.

- Full volume restore to a non-EAV fails because the EAV is larger than a non-EAV and MSGADR309E is issued, as shown here.

```
ADR309E (ttt)-mmmmm(yy), SOURCE AND TARGET DEVICE CAPACITIES DO NOT MATCH.
CYLINDER CAPACITY OF SOURCE VOLUME nnnnnnn, TARGET VOLUME nnnnnnn
```

Explanation: The capacity of the source volume is greater than that of the target volume. The nnnn is the highest cylinder number in hexadecimal.

System action: The task ends. Processing continues with the next control statement. The return code is 8.

- A physical track restore attempting to process tracks dumped from the cylinder-managed space of an EAV will fail. One of the following messages is issued:
  - MSGADR024E is issued when the track address specified is within the range of 65520 and 65535

```
ADR024E (ttt)-mmmmm(yy), TRACKS/OUTTRACKS VALUE IS INVALID FOR DEVICE.
       VALID CYLINDER RANGE IS 0 TO X'ccccccc'. VALID TRACK RANGE IS 0 TO X'h'
```

Explanation: The following values are not valid for the specified device types:

(1) - The range specified in the TRACKS keyword of a DUMP, COPY, RESTORE, or PRINT, or

(2) - The range specified for the OUTTRACKS keyword for COPY or RESTORE. The TRACKS values must fall within the cylinder and track range indicated in the message. The cccc and hhhh are high cylinder and head numbers, respectively, in decimal notation.

System Action: The task ends. Processing continues with the next control statement. The return code is 8.

Application Programmer Response: Correct the error, and rerun the job.

- MSGADR136E is issued when the track address specified is great than 65535, as shown here.

```
ADR136E (ttt)-mmmmm(yy), CONSTANT 'xxxx' IS NOT WITHIN VALUE RANGE
```

Explanation: The value of the constant (xxxx) is outside the range of values allowed for the associated parameter.

System Action: The task is not performed. Processing continues with the next control statement. The return code is 8.

Application Programmer Response: Check the command syntax for allowed values, correct the error, then reissue the command.

DFSMSdss considerations

When you use DFSMSdss to back up data sets with the extended attributes variable DS1EATTR set in the VTOC, you must make sure that these data sets can be restored into an environment that supports them. If the vendor attributes in the format-9 DSCB are essential to the validity of the data set, make sure that the environment on which they might be restored has EAVs to support these format-9 fields.

Note the following DFSMSdss restore considerations with APAR OA22900:

- The DFSMSdss restore functions and limitations that are supported through the coexistence APAR for pre-z/OS V1R10 systems will not perform a full volume restore of a full volume dump from an EAV.
- Tracks restore where track 0 is included will fail. 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 is changed to convert format-8 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 is issued. The message is shown here.

```
ADR556W (ttt)-,mmmm(yy), EXTENDED ATTRIBUTES FOR DATA SET dsname WERE NOT
RETAINED DURING {COPY | RESTORE},rsn
```

Explanation: The data set was copied or restored but various extended attributes were lost for the following reasons (rsn): where rsn is 1 or 2

1. Vendor attributes from the format-9 DSCB of the dumped data set were not retained for the target data set because the volume on which it was placed did not support format-8 and format-9 DSCBs.
2. Vendor attributes from the format-9 DSCB of the dumped or copied data set were not retained for the target data set due to problems updating the target's format-9 DSCB.

- Note the following additional reference information about restore toleration:
  - 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.
- The following PTFs describe the restore functions and limitations that are supported through a coexistence APAR on pre-z/OS V1R10 system:
  - Coexistence PTF numbers: UA40263 for V1R6, UA40264 for V1R7, UA40265 for V1R8, and UA40266 for V1R9.

4.13.5 DB2 support for EAVs

Note the following DB2 V8 and DB2 V9 considerations for EAVs:

- Admin enablement and log manager

  As long as BSDS and active logs are kept on non-EAVs and do not use admin enablement functions, run DB2 V8 and DB2 V9 without any new DB2 EAV support code.

  Otherwise, the coexistence support to DB2 V8 and DB2 V9 will need to be applied, which adds support to allow DB2 BSDS and active logs to reside anywhere on an EAV with:

  - APAR PK58292
- Therefore, DB2 admin enablement for EAV requires DB2 users to apply the appropriate coexistence PTFs before placing any BSDS or active logs on EAVs, or before using admin enablement functions on EAVs.
- DB2 log manager changes for EAVs are not needed as long as BSDS and active logs are kept on non-EAVs and do not use admin enablement functions; run DB2 V8 and V9 without any new DB2 EAV support code.

  However, DB2 users must apply the appropriate coexistence APARs before placing any BSDS or active logs on EAVs, or before using admin enablement functions on EAVs, as follows:

  - APAR PK61105 for DB2 V8 and V9

4.14 Coexistence with z/OS V1R10

Data set types with extended attributes are not supported in z/OS V1R10. However, note the following points.

- Coexistence support is provided in z/OS V1R10 to allow data sets with extended attributes to be processed by DADSM (such as scratch, rename, partial release) as long as no attempt is made to open the data set itself.
- Coexistence support is provided in z/OS V1R10 to allow straddled extents that can be created for VSAM data sets in z/OS V1R11 to be processed.

  A VSAM data set allocated in z/OS V1R11 with straddled extents is able to be processed from z/OS V1R10 with this coexistence support so that it can be referenced, extended, partially released, and scratched. In addition, DADSM convert routines were changed to tolerate and validate straddled extents.
- Extents that start in track-managed space and end in cylinder-managed space are tolerated.
- Coexistence support is provided for the EATTR data set attribute provided in DADSM, VSAM, and VSAM RLS functions to honor EATTR=NO and its other values for VSAM files allocated in z/OS V1R11, but processed in z/OS V1R10.

  For VSAM and VSAM RLS files, this allows an extendto a new volume to see the passed EATTR value. DADSM create functions, in turn, record the passed EATTR value in the format1 and format-8 DSCBs and search for space appropriately based on the EATTR attribute value.
- Application programs issuing CVAF and OBTAIN macros with the EADSCB=OK keyword are able to see format-8 and format-9 DSCBs for types of data sets that cannot be opened on z/OS V1R10. (Status code 84 will no longer be issued.)
  - CVAF and OBTAIN macros have been changed to return DSCB information for data sets that have extended attributes, but are not supported, from z/OS V1R10.
  - Removal of CVAF status stat084 errors in the OBTAIN, CVAFSEQ, CVAFDIR, and CVAFFILT macros allows data set list builders like ISPF and IEHLIST LISTVTOC, and all others that use these services, to display and see DSCBs with extended attributes as long as the EADSCB=OK keyword has been specified on the request. In other words, what can be seen on z/OS V1R11 is what can be seen on z/OS V1R10.
  - CVAF status code stat084 used to be set for a data set with extended attribute DSCBs where the data set type described by this DSCB was not supported in the release for EAS.
  - This restriction is being removed to facilitate support for non-VSAM data sets.
  - This indirectly affects the OBTAIN macro. Issuers of OBTAIN or a CVAF macro with EADSCB=OK might see a format-8 DSCB for a type of data set that cannot be opened in the current release.
  - This allows data set list builders to “see” non-EAS eligible data sets.
- Coexistence support is provided for DFSMSdss in z/OS V1R10 to detect extended-format sequential data sets in the EAS and fail them because they are not supported in the release; to restore extended format sequential data sets properly; to handle an extent that resides in both the non-EAS and EAS; and to properly handle updating and preserving format-9 DSCB vendor and internal fields.

- Coexistence support is provided for DFSMShsm in z/OS V1R10 to enable DFSMShsm to tolerate the DFSMShsm support provided in z/OS V1R11.

  In z/OS V1R10, DFSMShsm can RECALL, RECOVER and ARECOVER both VSAM and non-VSAM data sets from a z/OS V1R11 migration or backup copy with format-8 and format-9 DSCBs to non-EAVs. Format-8 and format-9 DSCBs will be converted to format-1 DSCBs and format-9 DSCB attributes will be lost if such a data set is recalled, recovered, or arecovered to non-EAV. In this case, the ARC0784I message will be issued if DFSMSdss issues a ADR556W message.

  For z/OS V1R9, support is added for preserving the EATTR value.

- Coexistence support is provided in Access Method Services (AMS) on z/OS V1R10 to recognize the EATTR value on an import of a portable data set created from z/OS V1R11 and also support passing the EATTR value on an z/OS V1R10 export.

  Having the EATTR value propagated to the portable data set allows a subsequent import of this portable data set to an z/OS V1R10 or higher release to support the EATTR value.

  Releases prior to z/OS V1R10 will not support EATTR on an import or export, thus the EATTR value will be not specified for an imported data set and in the exported portable data set.

z/OS BCP allocation support is changed in z/OS V1R10 to recognize and ignore the EATTR attribute. Internal use only change.

Coexistence APARs

The coexistence considerations for the EATTR data set attribute in z/OS V1R10 are provided with APAR numbers, as follows:

- OA26623, OA27069, OA27578, OA27577, OA27286, OA27270, OA27441, OA27545, OA26996, OA27146, and OA27465.

5

Predictive Failure Analysis

This chapter describes Predictive Failure Analysis (PFA), which is a component of z/OS that was made available as an SPE in March 2009 for z/OS V1R10. PFA provides support using remote checks from IBM Health Checker for z/OS to collect data about your installation.

The first two checks became available with z/OS V1R10. The second two checks became available with z/OS V1R11. The chapter discusses the highlights of PFA and explains these remote checks in more detail. The following topics are covered:

- Predictive Failure Analysis overview
- Support in z/OS V1R10
- LOGREC arrival rate check
- Message arrival rate check
- Virtual storage usage check
- Common storage usage PFA check
- PFA infrastructure
- PFA installation

5.1 Predictive Failure Analysis overview

Predictive Failure Analysis (PFA) is designed to predict potential problems with your systems. PFA extends availability by going beyond failure detection to predict problems before they occur. z/OS has implemented PFA to help eliminate soft failures such as exhaustion of common storage usage where a low priority authorized task obtains common storage, but obtains significantly more common storage than usual. This can cause a critical authorized system component to fail when attempting to obtain a normal amount of common storage. Soft failures usually occur in four generic areas:

- Exhaustion of shared resources
- Recurring or recursive failures often caused by damage to critical control structures
- Serialization problems such as classic deadlocks and priority inversions
- Unexpected state transition

Health checks

As mentioned, PFA uses remote checks from IBM Health Checker for z/OS to collect data about your installation. Using this data, PFA constructs a model of the expected (future) behavior of the z/OS images and then compares the actual behavior with the expected behavior. If the behavior is abnormal, PFA issues a health check exception.

z/OS UNIX and Java

PFA uses a z/OS UNIX System Services (z/OS UNIX) file system to manage the historical and problem data that it collects. Java 1.4 or later is required.

5.1.1 PFA-detected system failures

Detected system failures, which are caused by abnormal behavior, often generate “sympathy sickness.” With sympathy sickness, a minor problem escalates over time to the point where the service stops working. These failures are difficult to detect, which makes failure isolation difficult. Sympathy sickness has been observed when either hard failures or abnormal behavior generates a system failure that cannot be isolated to a failing component or subcomponent.

There are three general categories of software-detected system failures:

Masked failure This is a system failure that is detected by the software and corrected by the software.

Hard failure This failure occurs when the software fails completely, quickly, and cleanly, for example when an operating system kills a process.

Soft failure This failure is caused by abnormal behavior. A system failure caused by abnormal behavior is defined as unexpected, unusual, or abnormal behavior which causes the software solution to not provide the service requested. This abnormal behavior of the software, combined with events that usually do not generate failures, produce secondary effects that might eventually result in a system failure.

Abnormal system failures

A failure caused by abnormal behavior is difficult to recognize within the component, and the system can be damaged.

Systems in this state can be referred to as “sick but not dead.” There are several categories of problems that can generate “sick but not dead” systems:

Damaged systems These are systems suffering recurring or recursive errors caused by software defects.

Serialization problems These problems can be caused by incorrect priority assignments, by classic deadlocks, or by the resource owner terminating.

Resource exhaustion These problems can be caused by physical resources such as main storage or disk storage, or by software resources such as address space control blocks.

Indeterminate states The problems caused by these states are not possible to determine.

Predictive Failure Analysis uses historical data combined with machine learning and mathematical modelling to detect abnormal behavior and its potential causes. The objective is to detect events that lead to a “sick but not dead” system and allow corrective action to be taken. In general, errors fall into categories like recovery, logical errors, over-consumption, and sympathy sickness.

5.2 64-bit common support with z/OS V1R10

Many VSCR problems led to the introduction of 64-bit common support in z/OS V1R10. This trend, which started with z/OS V1R10, has opened a huge piece of common storage. However, the use of that storage must be monitored in a predictable manner, which is why Predictive Failure Analysis was developed.

This new piece of common storage 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.

In terms of RAS, because any address space can access storage in the 64-bit common area without explicitly having to request access, this is the lowest of the six envisioned alternatives listed in Table 5-1 on page 132. In fact, we are back where MVS was 25 years ago when XA was available and the advanced address space facility of ESA was not yet announced.At that time, this provided the opportunity to benefit from a larger addressing scheme and to have common areas (such as ECSA, ESQA). This is similar to the design of 64-bit common support.

Alternatives for sharing data in z/OS

Table 5-1 on page 132 summarizes the sharing locations to which you can move data, depending on the storage attributes of the required storage as well as the implied RAS level to be achieved.

Note that the order of the alternatives implies a decreasing RAS level. For example, for alternative 4. DREF storage, having 64-bit common storage areas provides a far lower RAS level for the overall z/OS structure than alternatives 1 and 2.

Table 5-1 Summary of sharing data alternatives

Storage attribute	Private (MVS)	CADS (AASF)	64-bit shared (z/OS V1R5)	64-bit common (z/OS V1R10)
1. Accessed by one space	Natural	Ideal for data isolation	Not the best solution	Not the best solution
2. Accessed by a set of spaces	Poor efficiency	Ideal for both RAS and efficiency	Natural if scalability is not a problem	Possible, but there is the potential of overlays
3. Accessed by every space	Inappropriate	Ideal for tens of GB with RAS and efficiency	Possible but cumbersome when large scale	Easy for hundreds of GB but potential RAS exposures
4. DREF storage	Yes for 31-bit No for 64-bit	Yes	No	Yes
5. Fixed storage	Yes	Yes for internal callers	No	Yes
6. Storage ownership	Task or address space	Task	System - storage must be explicitly freed	System - storage must be explicitly freed

5.3 PFA with z/OS V1R11

Predictive Failure Analysis that is shipped with z/OS V1R11 is the first pass of the z/OS capability to detect a damaged z/OS image or address space due to the exhaustion of common resources or recurring errors.

Two checks are provided to detect recurring failures; they became available with z/OS V1R10:

- Failures - LOGREC arrival rate (the dump arrival rate)
- Common storage below the bar

Two new checks, designed to detect exhaustion of storage or virtual storage leaks that generate abnormal excessive usage, are added with z/OS V1R11:

- Storage used by persistent address space
- Messages - message arrival count/CPU

These checks are explained in more detail in the following sections.

5.3.1 LOGREC arrival rate check

The first check made available in z/OS V1R10, the LOGREC arrival rate check, detects excessive failures by key. It measures the number of software failures by using the number of LOGRECs that arrive during the collection interval. The type of software errors counted are those with an SDWA record of the type “software-detected software error.”

This check categorizes the software failures by key into three categories: key 0; keys 1 to 8; and keys 9 to 15. Each category can trigger the exception for this check because PFA creates estimations of the expected number of software failures for each of these key categories. If

the number of exceptions is unusually high, the exception is issued. A list of jobs that had high arrivals are included in the report. If the arrivals are isolated to a single job or a small number of jobs, it means that the address space (or spaces) is damaged. If the arrivals cannot be isolated, is means the z/OS image is probably damaged.

The LOGREC arrival rate check is not able to detect all types of failures that can lead to a system soft failure, and it cannot detect a single critical failure. It also does not track other types of failures. It only tracks software failures.

The check needs the SDWA record with the LOGREC to track the failure. Therefore, if there is no usable SDWA record for the failure, it will not be tracked by the check. This check also looks for an excessive number of LOGRECs. Therefore, if there is a pattern of LOGRECs in which the number of LOGRECs arriving does not exceed the critical rate, then no exception can be issued.

///////

5.3.2 Common storage usage PFA check

The common storage usage PFS check is an “exhaustion of common resources” check by which PFA determines when common storage usage will be exhausted. PFA uses machine learning and historical data from this system to predict the future (within a bubble of predictability) level of common storage usage and determine whether the current trend is going to exceed the available common storage. The check combines observations at CSA and SQA for below-the-line common storage as well as ESQA and ECSA for above-the-line common storage.

This check uses a heuristic model to allocate the storage into these two buckets (below the line and above the line), and then mathematically models the predictable storage usage to detect problems. If storage is not going to be exhausted, it will issue an information message. If PFA believes storage will be exhausted before the next modeling, it will an issue an exception. PFA produces a report that shows the top contributors to readjust. In most cases, the address space causing the storage exhaustion will be on this list.

To reduce false positive numbers after an IPL, the first hour of data collected concerning storage usage is not used in the model. As with all PFA checks, collection and modeling occur asynchronously on a user-specified schedule.

Note that the common storage usage check is not able to detect all types of common storage exhaustion. It cannot currently detect exhaustion due to fragmentation or rapid growth in common storage usage in a very short period of time. An address space that uses an unusual amount of common storage without impacting the z/OS image will not cause an exception.

5.3.3 Message arrival rate check

The second check made available in z/OS V1R11, the message arrival rate check, detects excessive failures and is similar to the LOGREC arrival rate check. This check detects an abnormal arrival rate of console messages (that is, WTO and WTOR messages). The rate is calculated by dividing the count of the arrivals in the collection interval by the amount of CPU used (in seconds) in the collection interval.

If the arrival rate found at the last collection is excessively high when compared to the estimation, an exception message is issued. It provides a list of jobs that caused the high rate.

Note the following points:

- If isolated to a job or small number of jobs, an address space is damaged.
- If not isolated to a specific job or small number of jobs, the z/OS image is damaged.

Note: The message arrival rate check is not designed to detect abnormal message patterns or single critical messages.

5.3.4 Virtual storage usage check

The first check available starting with z/OS V1R11, the virtual storage usage check, also detects the exhaustion of shared resources. This usage check is introduced to detect storage leaks in a persistent address space. A “persistent” address space is defined to be an address space that starts within an hour after IPL.

The virtual storage usage check tracks frames and slots used by persistent address spaces. It models a foreseeable virtual storage usage for the persistent address spaces that have had the largest change. It estimates the number of pages that the address space will consume by the end of the model interval. It issues an exception if one of these persistent address spaces exceeds the estimate for that address space after the standard deviation is applied. The report issued with the exception lists the address spaces that have been estimated.

The virtual storage usage check is not able to detect small virtual storage leaks, and it is not meant to be a virtual storage monitor. It cannot detect virtual storage fragmentation or rapid growth on a machine-time scale.

5.4 Hypothetical CSA usage failures

Figure 5-1 on page 135 shows a hypothetical example of a case where PFA issues an exception for exhaustion of common storage. In this example, modeling occurred every 6 hours. The data collected during the first hour after IPL while the system was stabilizing is ignored. In hours 1 to 6, the system showed growth and the slope of the estimated usage showed growth as well. During the second model interval, the system showed a degree of growth, but then inclined during approximately the last hour. PFA detected that the current usage was not only greater than the estimation, but also was in danger of exhausting common storage and the exception was issued.

As you can see, no numerical values are presented here and there is a more complex algorithm involved. However, this chart illustrates that increased growth with the potential to exhaust common storage will cause the exception to be issued. To reduce false positive numbers, a configurable parameter called THRESHOLD has been provided for the common storage usage check.

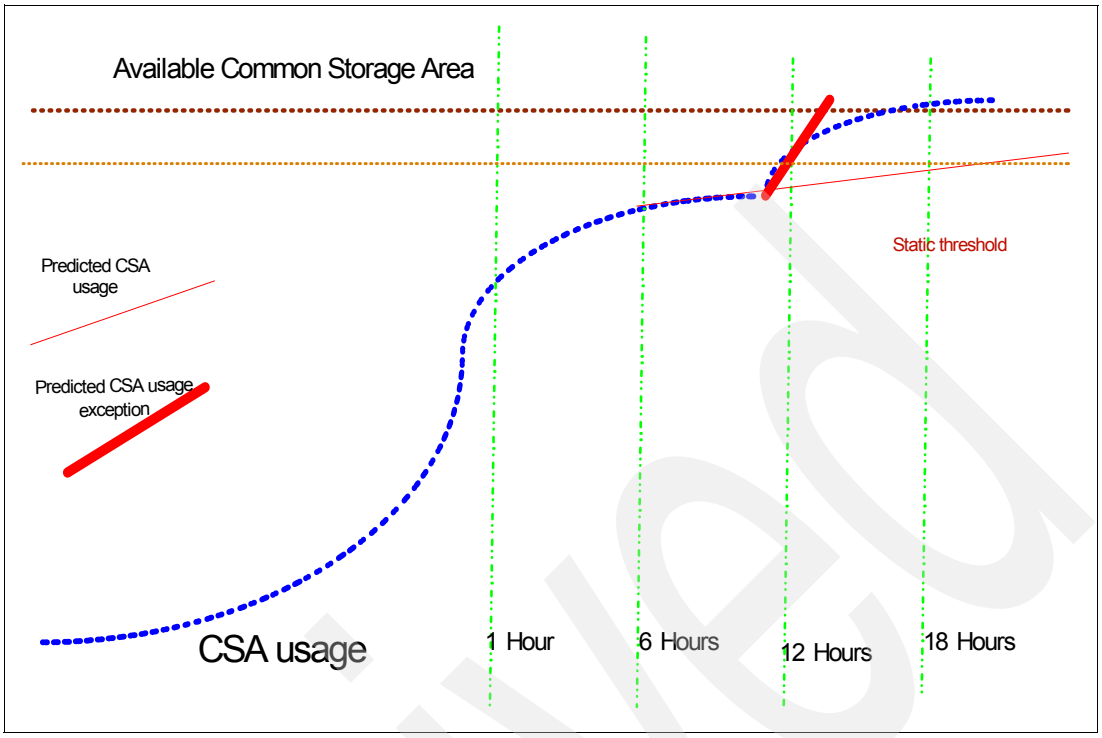


Figure 5-1 Hypothetical CSA usage failure

5.4.1 CSA exception report

A partial common storage usage check prediction report is shown in Figure 5-2 on page 136. Notice that the below-the-line and the above-the-line storage are represented separately. They are also compared separately and either can cause an exception.

A common heading is provided on the reports of all PFA checks that provides the last data collection time, the collection interval, the last model time, and the model interval. All values are in kilobytes.

The “future” prediction is the value that was modeled at the last model interval that is the estimated value at the end of the model interval (or 2 hours ahead, if the model interval is less than 2 hours).

The top estimated users’ list is provided only if an exception is issued or debug is turned on. If the check determines that there is no problem such that the information is issued, this list is not provided. Producing this list involves a certain amount of performance overhead and because this check’s comparisons are done every minute, the list of jobs is not provided unless it is needed to analyze a problem.

The top estimated users’ list can be used to determine the address space (or spaces) that is causing the potential storage exhaustion. If more information is needed, the prediction file in the PFA_COMMON_STORAGE_USAGE/data directory is available to do further analysis, such as to find the ASID and the PSW of the location in storage from which the CSA or SQA was requested.

Because the *SYSTEM* owned storage can be from many places, the current usage is not accumulated on the report to further reduce overhead, so `UNAVAILIABLE` is displayed instead.

```
 Common Storage Usage Prediction Report
(heading information intentionally omitted here)
Below line CSA+SQA (in kilobytes):
   Current usage          :         750
   Future prediction      :         613
   Capacity when predicted:        5212
Above line CSA+SQA (in kilobytes):
   Current usage          :      205555
   Future prediction      :      235408
   Capacity when predicted:      526112
Top predicted users:
   Job            Storage      Current Usage       Predicted Usage
   Name           Location     (in kilobytes)      (in kilobytes)

   _________      ________     ______________      ______________
   CSATST4        ABOVE                35002               40023
   CSATST3        ABOVE                32364               33530
   *SYSTEM*       ABOVE            UNAVAILABLE               190
```

Figure 5-2 Common storage usage prediction report

5.4.2 LOGREC arrival rate prediction report

The LOGREC arrival rate check can detect when an address space might be damaged or when the entire z/OS image might be damaged based on the number of software LOGRECs that arrive within a collection interval.

When a problem occurs, the LOGREC arrivals tend to be concentrated in a specific address space or job. This fact allows PFA to accumulate the LOGRECs into general buckets based on key for a low-cost scoring algorithm. When an address space is damaged, a very large number of LOGRECs are typically generated. When things are normal, the arrival rate is also stable (a small, single-digit number of LOGRECs).

Note: Differentiation is based on storage keys (key 0, other authorized keys, user key), and not on capabilities such as the extended authorization index.

Figure 5-3 on page 137 depicts a LOGREC arrival rate prediction report. PFA models the LOGREC arrivals for three buckets of keys: key 0; keys 1 to 7; and keys 8 to 15. It models estimations for those three categories for four ranges:

- Data that was collected in the last hour
- Data that was collected over the last 24 hours of data
- Data that was collected over the last seven days of data
- Data that was collected over the last 30 days of data.

If there is not enough data collected for all of those time lengths, the line for that time is not included on the report.

```
 LOGREC Arrival Rate Prediction Report
(heading information intentionally omitted)
                                Key 0      Key 1-7     Key 8-15

                              _________   _________   _________
Arrivals in last
   collection interval:             1           0           2
Predicted rates based on...
     1 hour of data:                1           0           1
   24 hours of data:                0           0           1
     7 days of data:                0           0           1
    30 days of data:                0           0           1
Jobs having LOGREC arrivals in last collection interval:
   Job Name        ASID        Arrivals

   ________        ____        ________
   LOGREC08        0029               2
   LOGREC00        0027               1
```

Figure 5-3 LOGREC arrival rate prediction report

The report shows the arrivals in the last collection interval's worth of time by key and the estimations by key for estimations based on those four models of data. Any one of those keys and predictable times can cause the exception to be issued when compared to the arrivals in the last collection interval's worth of time (after the configurable standard deviation is applied).

Comparisons are also made using the arrivals in the last collection interval against the estimations, not the arrivals in the last collection.

- For example, if the collection interval is 60 minutes, then the actual count in the last 60 minutes is used in the comparisons when the check is run.
- "Arrivals in last collection interval" denotes the arrivals in the last COLLECTINT minutes from the time the check was run.
- "Jobs having LOGREC arrivals in last collection interval" lists the jobs contributing to the arrival count. This list is displayed if the arrival count > 0; that is, if there were no arrivals, then the list is not displayed. Most often, the job (or jobs) with the most arrivals are the reason the exception was issued.

Note: The comparisons are made using the arrivals in the last collection interval's worth of time, not the arrivals in the last collection itself. This is done to include the most recent arrivals in the comparison. An example is included in Figure 5-4 on page 138 to further explain this algorithm.

It is important to understand what timeframes the arrivals occurred in to understand which arrivals are included in the comparison.

Figure 5-4 on page 138 shows 9 collections. Modeling occurs in the middle of the seventh collection interval. The estimations are created for 1 hour, 24 hours, 1 day, and 30 days by key, and they and include the data previously collected for those times, if available. The check runs at the times represented by the arrows. The "arrivals in the last collection interval" for each of those comparisons are represented by the matching colored brackets.

Therefore, when the first arrow on the left occurs, the check is being run by Health Checker. It accumulates the arrivals that occurred in the green brackets and compares those to the estimations modeled in the middle of collection 7.

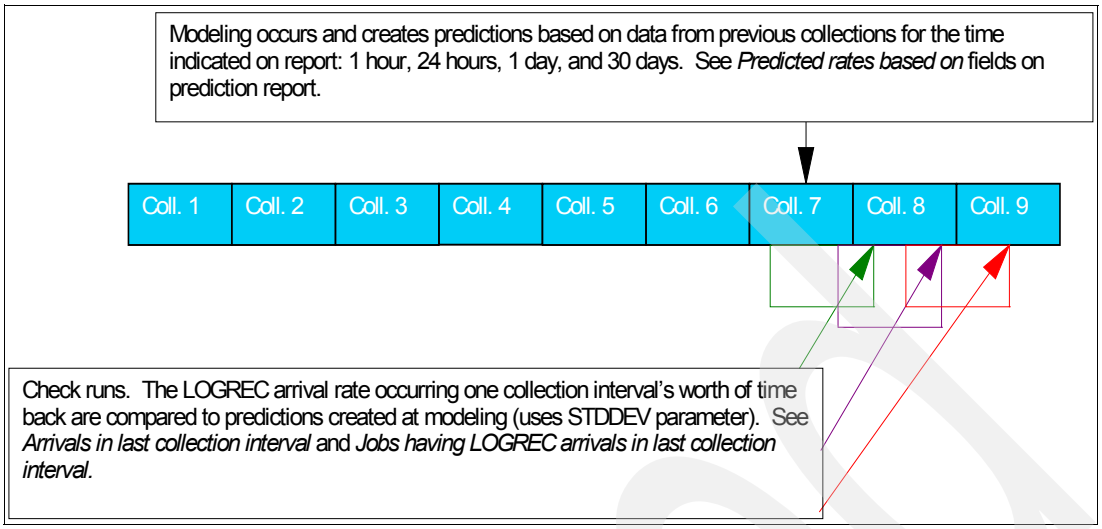


Figure 5-4 LOGREC arrival rate details

5.4.3 Virtual storage usage prediction report

The virtual storage usage check determines when a started task or started job is leaking virtual storage, to prevent an outage caused by exhausting auxiliary storage slots.

A unit of work can acquire a range of virtual storage addresses for its use. When the memory represented by the virtual storage address range is referenced or updated, the operating system backs the memory with real frames. The operating system can move the data from the real frames to auxiliary storage. This check detects abnormal usage (a larger than expected amount) of virtual storage by monitoring the usage of real frames and auxiliary storage slots of persistent jobs. Dataspaces allocated by jobs are also considered part of the virtual storage usage.

Persistent address spaces

A persistent job is defined to be one that starts within an hour after IPL. Figure 5-5 on page 139 shows an example where a persistent address space allocates more frames over the period of time. The address space seems to free up part of the storage obtained, but at a considerably lower rate than the storage obtained in that cycle. This chart shows the increased storage growth that has the potential to exhaust virtual storage, which will cause the exception to be issued.

To determine when a started task or started job is leaking virtual storage to prevent an outage caused by exhausting auxiliary storage slots and frames, the usage for each persistent job is calculated as the sum of the following:

- The number of 4 K frames used (includes dataspaces)
- The number of AUX slots used

Virtual storage usage check STDEV parameter

The virtual storage usage check looks at each individual persistent address space for storage leakage (unlike the CSA check, where common storage for the entire system is monitored); see Figure 5-5 on page 139. To reduce false positive numbers, a configurable parameter called STDDEV has been provided for the virtual storage usage check.

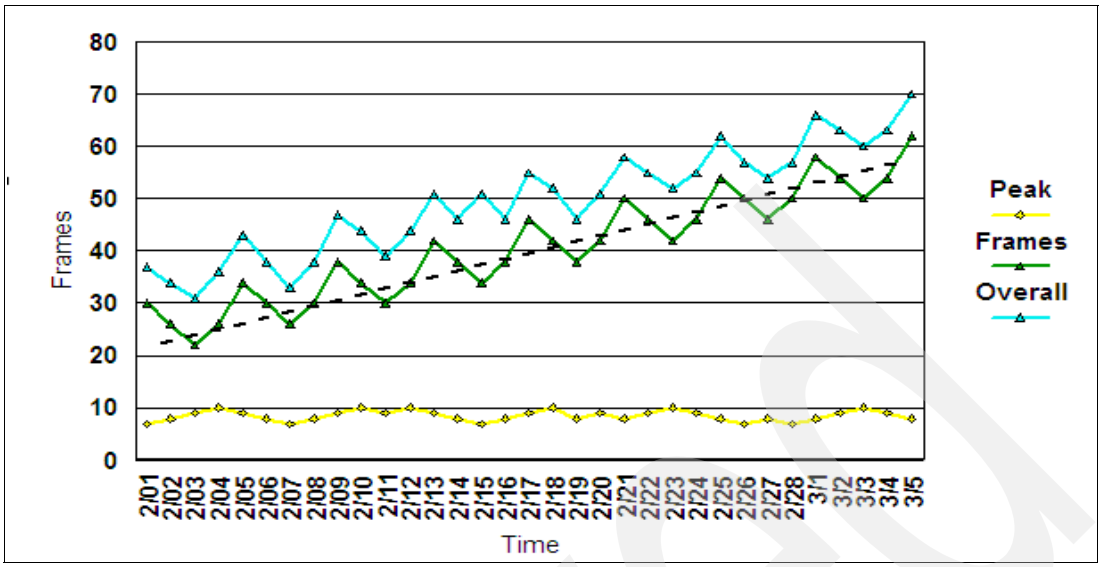


Figure 5-5 Virtual storage leakage behavior

As shown in Figure 5-6, the top predicted users are those jobs that were estimated to use the most virtual storage by the end of the model interval. Only the top 14 persistent jobs are selected to be printed or displayed in the report.

This list is sorted by predicted usage. If the check determines that there is no problem such that the informational message is issued, this list is not provided.

```
Virtual Storage Usage Prediction Report

(heading information intentionally omitted here)


Top predicted users:

   Job Name       Current Usage     Predicted Usage
   ---------------------------------------------

   ZFS                    12223              12329
   XCFAS                   1593               1601
   VTAMOSR3                1885               1881
   TRACE                    367                367
   SMS                      682                687
```

Figure 5-6 Virtual storage usage prediction report

The report also shows the current and the predicted usage for the top jobs. The current and predicted usage shown are the number of 4 K frames and AUX slots in usage.

After the modeling is done, a check is performed in which the current usage is compared to the predicted usage for each individual persistent address space. Using the STDDEV parameter, if it is determined that an address space is leaking substantial virtual storage, an exception is issued. This algorithm differs from the CSA check where an exception is raised for the entire storage, not for individual persistent address spaces.

5.4.4 Message arrival rate details and report

As mentioned, the message arrival rate check is designed to detect a damaged system based on the arrival rate being higher than expected. Messages included in the count of arrivals are:

- Single line and multi-line WTO and WTOR messages.
- Multi-line WTO messages are counted as one message.
- Branch entry WTO messages are not counted.

Messages are counted prior to being excluded or modified by other functions such as message flooding automation. The message arrival rate is a simple ratio calculated by dividing the number of arrivals in the collection interval by the CPU used in that same collection interval. Figure 5-7 illustrates the message arrival rate.

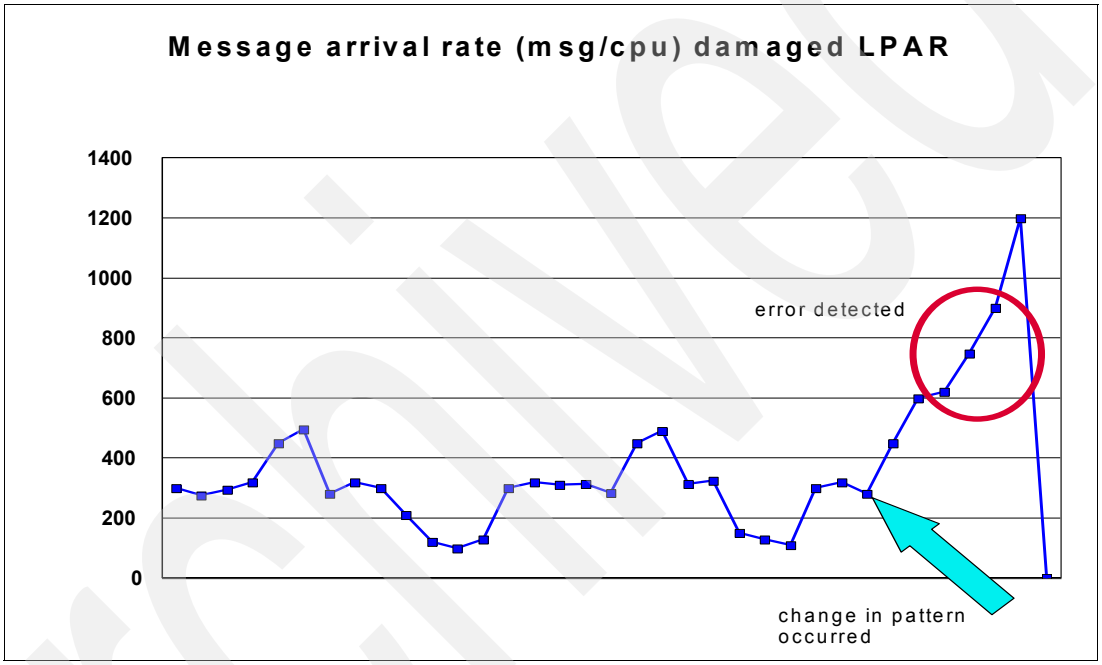


Figure 5-7 Message arrival rate

The message arrival rate check differs from the other checks in the way the check is run to make comparisons. Rather than having a set time INTERVAL as a Health Checker parameter, the check is designed to run after every collection. Performing the check automatically upon the successful completion of a collection enables it to compare the most recent arrivals with the estimations modeled at the last model interval. This enhances both the performance of the check itself as well as the responsiveness of the check to the current activity of the system.

The message arrival rate check is similar to the LOGREC arrival rate check in that they both make estimations based on 1 hour of data, 24 hours of data, and 7 days of data. Note that message arrival rate does not model back using 30 days of data, however. If that amount of data is not available for the system as a whole, that line is not printed on the report. For the jobs listed in the report details, if data is not available for a particular job for any given time frame, then UNKNOWN is printed on the report.

The report is very similar to the LOGREC arrival rate report.

5.5 Using PFA with commands and SDSF

PFA activity and results can be viewed either from the console or from SDSF. Summary information for the checks show the check name, whether it is active in Health Checker, the last collection time, and the last model time. Either all checks can be shown or individual checks can be shown by specifying the name of a check or a wildcard that matches more than one check. Wildcards can be specified as the last character of the check name.

PFA modify command

This section provides examples of the PFA **modify** command. It can be used to display summary or detailed information for the PFA checks and to display status information for the PFA infrastructure. Figure 5-8 shows a PFA check summary.

The syntax of the PFA **modify** command is very similar to the Health Checker **modify** command and is documented in the PFA documentation.

```
F PFA,DISPLAY,CHECKS
AIR013I 13:47:15 PFA CHECK SUMMARY
                                        LAST SUCCESSFUL LAST SUCCESSFUL
CHECK NAME                      ACTIVE  COLLECT TIME    MODEL TIME
PFA_COMMON_STORAGE_USAGE         YES
PFA_LOGREC_ARRIVAL_RATE          YES
PFA_FRAMES_AND_SLOTS_USAGE       YES
PFA_MESSAGE_ARRIVAL_RATE         YES
```

Figure 5-8 PFA check summary

PFA command to display check details

Detailed information for a check shows counts and times for collection and modeling. It also shows the parameters specific to the check. In fact, to display the cumulative set of parameters for this check, you must use the PFA **modify** command. PFA allows the user to modify parameters individually and accumulate the changes, rather than requiring all parameters to be specified when modifying. Be aware that displaying the check's parameters using Health Checker commands will not show the cumulative list of parameters if the parameters were changed using more than one **modify** command.

The **F PFA,DISPLAY,CHECKS,DETAIL** command in shown in Figure 5-9 on page 142, Figure 5-10 on page 142, Figure 5-11 on page 143, and Figure 5-12 on page 143. The figures show a single command response, with each figure displaying one of the four checks available for PFA.

Figure 5-9 on page 142 displays the PFA checks detail output.

```
------------------------ LIST MCS COMMAND OUTPUT ---------- Row 1 to 24 of 74
C => _                                                          S => HALF
Mode: BOTH   SC74     LAFITTE
Display =>  Time     Y  System  Y  Job     Y  Hold => N  (Y or N)
                                             13:36 09/05/05 Enter '?' for HELP
--------------------------------------------------------------------------------
 AIR018I 13:36:20 PFA CHECK DETAIL
 CHECK NAME: PFA_COMMON_STORAGE_USAGE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 9
     SUCCESSFUL COLLECTION COUNT     : 0
     LAST COLLECTION TIME            : 05/05/2009 13:33:19
     LAST SUCCESSFUL COLLECTION TIME:
     NEXT COLLECTION TIME            : 05/05/2009 13:48:19
     TOTAL MODEL COUNT               : 0
     SUCCESSFUL MODEL COUNT          : 0
     LAST MODEL TIME                 :
     LAST SUCCESSFUL MODEL TIME      :
     NEXT MODEL TIME                 : 05/05/2009 17:17:58
     CHECK SPECIFIC PARAMETERS:
       COLLECTINT                    : 15
       MODELINT                      : 360
       COLLECTINACTIVE               : 1=ON
       DEBUG                         : 0=OFF
       THRESHOLD                     : 5
 CHECK NAME: PFA_LOGREC_ARRIVAL_RATE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 2
     SUCCESSFUL COLLECTION COUNT     : 2
     LAST COLLECTION TIME            : 05/05/2009 13:18:15
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 5-9 PFA checks detail - common storage usage

Apart from the PFA common storage usage, the LOGREC arrival rate can also be displayed, as shown in Figure 5-10.

```
------------------------ LIST MCS COMMAND OUTPUT --------- Row 20 to 43 of 74
C => _                                                           S => 1
Mode: BOTH   SC74     LAFITTE
Display =>  Time     Y  System  Y  Job     Y  Hold => N  (Y or N)
                                             13:36 09/05/05 Enter '?' for HELP
--------------------------------------------------------------------------------
 CHECK NAME: PFA_LOGREC_ARRIVAL_RATE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 2
     SUCCESSFUL COLLECTION COUNT     : 2
     LAST COLLECTION TIME            : 05/05/2009 13:18:15
     LAST SUCCESSFUL COLLECTION TIME: 05/05/2009 13:18:15
     NEXT COLLECTION TIME            : 05/05/2009 14:18:15
     TOTAL MODEL COUNT               : 0
     SUCCESSFUL MODEL COUNT          : 0
     LAST MODEL TIME                 :
     LAST SUCCESSFUL MODEL TIME      :
     NEXT MODEL TIME                 : 05/06/2009 11:17:58
     CHECK SPECIFIC PARAMETERS:
       COLLECTINT                    : 60
       MODELINT                      : 360
       COLLECTINACTIVE               : 1=ON
       DEBUG                         : 0=OFF
       STDDEV                        : 2
 CHECK NAME: PFA_VIRTUAL_STORAGE_USAGE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 9
     SUCCESSFUL COLLECTION COUNT     : 9
     LAST COLLECTION TIME            : 05/05/2009 13:33:19
     LAST SUCCESSFUL COLLECTION TIME: 05/05/2009 13:33:19
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 5-10 PFA checks detail - LOGREC arrival rate

Figure 5-11 on page 143 displays virtual storage usage.

```
------------------------- LIST MCS COMMAND OUTPUT --------- Row 38 to 61 of 74
C =>                                                              S => 2
Mode: BOTH   SC74     LAFITTE
Display =>  Time     Y  System  Y  Job     Y  Hold => N  (Y or N)
                                             13:36 09/05/05 Enter '?' for HELP
------------------------------------------------------------------------------
 CHECK NAME: PFA_VIRTUAL_STORAGE_USAGE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 9
     SUCCESSFUL COLLECTION COUNT     : 9
     LAST COLLECTION TIME            : 05/05/2009 13:33:19
     LAST SUCCESSFUL COLLECTION TIME: 05/05/2009 13:33:19
     NEXT COLLECTION TIME            : 05/05/2009 13:48:19
     TOTAL MODEL COUNT               : 0
     SUCCESSFUL MODEL COUNT          : 0
     LAST MODEL TIME                 :
     LAST SUCCESSFUL MODEL TIME      :
     NEXT MODEL TIME                 : 05/05/2009 17:17:58
     CHECK SPECIFIC PARAMETERS:
       COLLECTINT                    : 15
       MODELINT                      : 360
       COLLECTINACTIVE               : 1=ON
       DEBUG                         : 0=OFF
       STDDEV                        : 3
 CHECK NAME: PFA_MESSAGE_ARRIVAL_RATE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 9
     SUCCESSFUL COLLECTION COUNT     : 9
     LAST COLLECTION TIME            : 05/05/2009 13:33:19
     LAST SUCCESSFUL COLLECTION TIME: 05/05/2009 13:33:19
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 5-11 PFA checks detail - virtual storage usage

Figure 5-12 displays the message arrival rate.

```
------------------------- LIST MCS COMMAND OUTPUT --------- Row 56 to 74 of 74
C =>                                                              S => 2
Mode: BOTH   SC74     LAFITTE
Display =>  Time     Y  System  Y  Job     Y  Hold => N  (Y or N)
                                             13:36 09/05/05 Enter '?' for HELP
------------------------------------------------------------------------------
 CHECK NAME: PFA_MESSAGE_ARRIVAL_RATE
     ACTIVE                          : YES
     TOTAL COLLECTION COUNT          : 9
     SUCCESSFUL COLLECTION COUNT     : 9
     LAST COLLECTION TIME            : 05/05/2009 13:33:19
     LAST SUCCESSFUL COLLECTION TIME: 05/05/2009 13:33:19
     NEXT COLLECTION TIME            : 05/05/2009 13:48:19
     TOTAL MODEL COUNT               : 0
     SUCCESSFUL MODEL COUNT          : 0
     LAST MODEL TIME                 :
     LAST SUCCESSFUL MODEL TIME      :
     NEXT MODEL TIME                 : 05/05/2009 17:17:58
     CHECK SPECIFIC PARAMETERS:
       COLLECTINT                    : 15
       MODELINT                      : 360
       COLLECTINACTIVE               : 1=ON
       DEBUG                         : 0=OFF
       STDDEV                        : 3
F PFA,DISPLAY,CHECKS,DETAIL
****************************** Bottom of data ********************************



 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 5-12 PFA checks detail - message arrival rate

Display PFA infrastructure status

The PFA infrastructure status can also be displayed using the PFA **`modify`** command. The number of checks registered is the number of checks that exist in the PFA infrastructure. The

number of checks active are the number of PFA checks that are ACTIVE(ENABLED) in Health Checker; see Figure 5-13.

```
F PFA,DISPLAY
AIR017I 10.31.32 PFA STATUS
NUMBER OF CHECKS REGISTERED    : 4
NUMBER OF CHECKS ACTIVE        : 4
COUNT OF COLLECT QUEUE ELEMENTS: 0
COUNT OF MODEL QUEUE ELEMENTS  : 0
COUNT OF JVM TERMINATIONS      : 0
```

Figure 5-13 PFA checks status

5.5.1 SDSF support for PFA

After PFA has been started, the SDSF Health Checker panel (CK) displays the results of PFA checks; see Figure 5-14.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HEALTH CHECKER DISPLAY  SC74                    LINE 49-72 (128)
COMMAND INPUT ===>                                     SCROLL ===> HALF
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=SC74
NP    NAME                          CheckOwner      State              Status
      PFA_COMMON_STORAGE_USAGE      IBMPFA          ACTIVE(ENABLED)    SUCCES
      PFA_LOGREC_ARRIVAL_RATE       IBMPFA          ACTIVE(ENABLED)    SUCCES
      PFA_MESSAGE_ARRIVAL_RATE      IBMPFA          ACTIVE(ENABLED)    RUNNIN
      PFA_VIRTUAL_STORAGE_USAGE     IBMPFA          ACTIVE(ENABLED)    SUCCES
      RACF_FACILITY_ACTIVE          IBMRACF         ACTIVE(ENABLED)    SUCCES
      RACF_GRS_RNL                  IBMRACF         ACTIVE(ENABLED)    SUCCES
      RACF_IBMUSER_REVOKED          IBMRACF         ACTIVE(ENABLED)    EXCEPT
      RACF_ICHAUTAB_NONLPA          IBMRACF         ACTIVE(ENABLED)    SUCCES
      RACF_OPERCMDS_ACTIVE          IBMRACF         ACTIVE(ENABLED)    SUCCES
      RACF_SENSITIVE_RESOURCES      IBMRACF         ACTIVE(ENABLED)    EXCEPT
      RACF_TAPEVOL_ACTIVE           IBMRACF         ACTIVE(ENABLED)    EXCEPT
      RACF_TEMPDSN_ACTIVE           IBMRACF         ACTIVE(ENABLED)    EXCEPT
      RACF_TSOAUTH_ACTIVE           IBMRACF         ACTIVE(ENABLED)    SUCCES
      RACF_UNIXPRIV_ACTIVE          IBMRACF         ACTIVE(ENABLED)    SUCCES
      RSM_AFQ                       IBMRSM          ACTIVE(ENABLED)    SUCCES
      RSM_HVSHARE                   IBMRSM          ACTIVE(ENABLED)    SUCCES
      RSM_MAXCADS                   IBMRSM          ACTIVE(ENABLED)    SUCCES
      RSM_MEMLIMIT                  IBMRSM          ACTIVE(ENABLED)    EXCEPT
      RSM_REAL                      IBMRSM          ACTIVE(ENABLED)    SUCCES
      RSM_RSU                       IBMRSM          ACTIVE(ENABLED)    SUCCES
      RTM_IEAVTRML                  IBMRTM          ACTIVE(ENABLED)    EXCEPT
```

Figure 5-14 SDSF CK panel

General comments are provided for the various PFA checks such as shown in Figure 5-15 on page 145 for common storage usage.

```
    Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PFA_VIRTUAL_STORAGE_USAGE     LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> HALF
******************************** TOP OF DATA *********************************
CHECK(IBMPFA,PFA_VIRTUAL_STORAGE_USAGE)
START TIME: 07/10/2009 10:16:45.884251
CHECK DATE: 20080330  CHECK SEVERITY: MEDIUM
CHECK PARM: DEBUG(0)                    STDDEV(3) COLLECTINT(15)
MODELINT(360) COLLECTINACTIVE(1)


AIRH150I Comparisons of predictions and arrivals will occur when the
check is run after modeling has run and succeeded.  Modeling is
scheduled for 06/12/2009 15:47:08.

END TIME: 07/10/2009 10:16:45.885559  STATUS: SUCCESSFUL
```

Figure 5-15 PFA Common storage usage general comments

Details are shown in the same SDSF panel; see Figure 5-16.

```
                Common Storage Usage Prediction Report

Last successful model time     :  05/27/2009 08:30:24
Next model time                :  05/27/2009 14:30:24
Model interval                 :  360
Last successful collection time:  05/27/2009 09:15:29
Next collection time           :  05/27/2009 09:30:29
Collection interval            :  15

Below line CSA+SQA (in kilobytes):
   Current usage          :        1155
   Future prediction      :        1155
   Capacity when predicted:        6196

Above line CSA+SQA (in kilobytes):
   Current usage          :       65829
   Future prediction      :       67864
   Capacity when predicted:      208396

END TIME: 05/27/2009 09:29:00.781337  STATUS: SUCCESSFUL
******************************** BOTTOM OF DATA ********************************
```

Figure 5-16 SDSF Common storage usage prediction report

5.6 PFA infrastructure

The PFA infrastructure manages the PFA address space, connects to IBM Health Checker for z/OS (referred to as “Health Checker” from now on), displays the status of PFA, and launches the JVM to model the data to create an estimation.

There are no hardware dependencies for PFA, and exploiters of this function are all system operators. PFA simply has to be started and then monitored for the exceptions it produces; automate the monitoring. There are no APIs to exploit. The software dependencies for PFA are shown in Figure 5-17 on page 146. They are IBM Health Checker for z/OS, z/OS UNIX file system, and Java 1.4 or later.

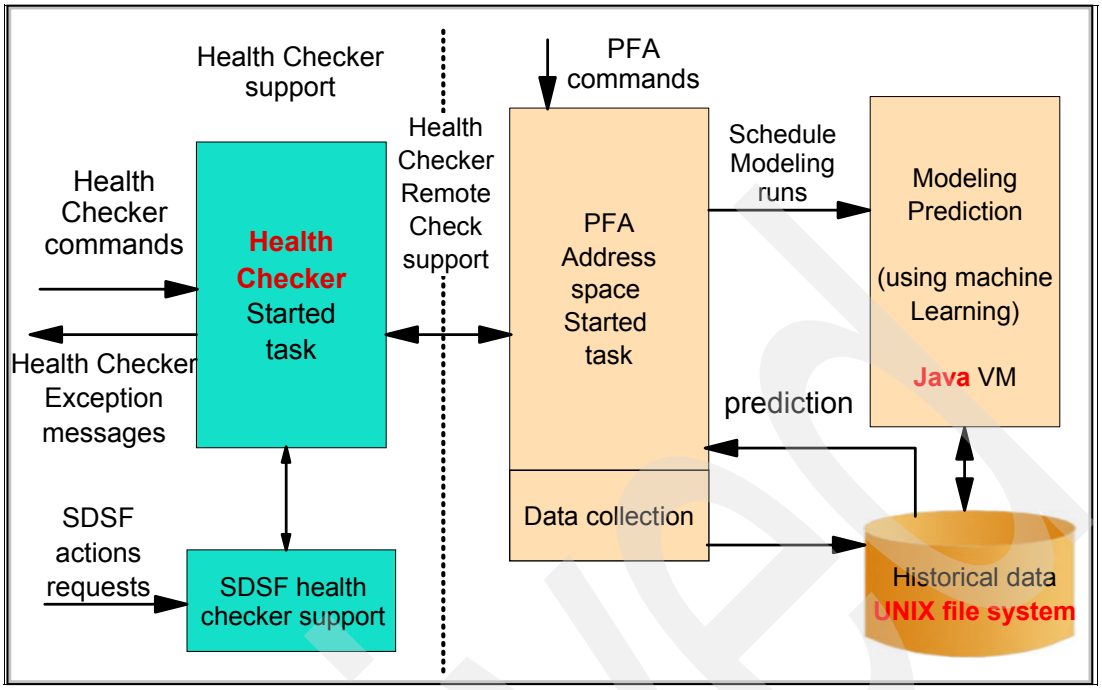


Figure 5-17 PFA infrastructure

5.6.1 PFA parameters for health checks

PFA is analyzed using remote health checks. Therefore, the health check commands, the interface through SDSF, and the reporting mechanism available through Health Checker are fully usable for the PFA checks.

PFA also contains check-specific code that collects the data for an individual check, models the data to generate a estimation, and compares the actual values to the estimations to issue an exception or an informational message.

PFA checks have three basic internal functions, as explained here:

- PFA checks collect data.

  Data collection for a check is specific to that check; the check code collects data from the system that is pertinent to the check. For example, if the check needs to calculate a type of storage usage, it interrogates the system control blocks to accumulate the storage used. If it is counting message arrivals of a certain kind, it uses the appropriate system interface to collect that data.

  Data collection occurs asynchronously on an interval that is configurable by the user. For example, the default value for the data collection function for checks might be to collect data every 15 minutes. This parameter is called COLLECTINT.

- PFA checks model the data to generate an estimation based on the data collected.

  This modeling function takes the collected data and estimates the value that it expects to see at the end of the model interval. The model interval is also configurable by the user. For example, the default value for the modeling interval for checks might be to model data every 6 hours. This parameter is called MODELINT. Modeling also runs asynchronously when it is determined that it is time to model.

- PFA checks perform the comparisons needed to issue an exception or an informational message.

  The checks compare what is occurring on the system to what was estimated and issues the appropriate message and report. This function is typically initiated by Health Checker when the time in the INTERVAL parameter for the check is reached. It can also be done for most checks by a user running the check using Health Checker commands.

PFA and z/OS UNIX

PFA also manages its data store, which is in the UNIX file system. The collected data is stored for use by the modeling code to produce an estimation. The estimations are also stored in the file system for use by the code that performs the comparisons and produces the reports.

All PFA checks have two additional check-specific parameters: COLLECTINACTIVE and DEBUG, as explained here:

- The COLLECTINACTIVE parameter is set to yes by default and collects and models data for the check even if the check is not active (enabled) in Health Checker.
- The DEBUG parameter is used to collect additional debug information to help analyze a PFA problem.

PFA checks can also have check-specific parameters. At this time, each check has a parameter to assist PFA in reducing false positive numbers. These parameters are also configurable by the user to allow for greater flexibility on a per-system basis.

5.6.2 Differences between PFA checks and other remote health checks

As mentioned, PFA checks have several parameters. Health Checker requires all parameters to be specified on a modify even if only one parameter is being changed. And, if using the Health Checker **`modify`** command on the command line, only 126 characters are allowed and not all PFA check parameters can be specified. In addition, it was not considered very useful to need to input all parameters just to change the value of one parameter.

Therefore, PFA made a change so that not all parameters need to be specified when modifying parameters using Health Checker. PFA internally tracks the values of each parameter so that if not all parameters are specified, the previous value for the parameters not specified are retained.

However, Health Checker is not aware of the internal storage of the PFA parameters. Therefore, it only has the capability of displaying the last modify operation performed. If you use multiple modify operations or do not specify all of the parameters, Health Checker can only display the last parameter modified. Therefore, to see all parameters in use by any PFA check, the modify PFA display command must be used.

When the debug parameter in Health Checker is set, PFA is not notified until the next run of the check. However, debug data needs to be generated for the collect and model phases of the checks as well. Therefore, every PFA check has a debug parameter that is a check-specific parameter applying to all phases of PFA checks. The debug parameter in Health Checker is ignored by PFA.

After an exception is issued, the exception is issued every time the check is run even though new data has not yet been collected and no new estimations have been modeled. This problem was especially annoying for system operators who carry a pager. Therefore, PFA has

been adapted so that when an exception is issued, the check is deactivated in Health Checker so that the check is not run again.

- If COLLECTINACTIVE is on, then after new data is collected and new estimations are modeled, the check is activated again which immediately runs the check.
- If the exception needs to be issued again, it is issued with new data on the report and the check is deactivated again.
- If everything is now OK, the informational message is issued.
- If COLLECTINACTIVE is off, PFA will not collect or model data and the check will remain deactivated until manually activated by the user or COLLECTINACTIVE is turned on.

Figure 5-18 illustrates the overall PFA processes.

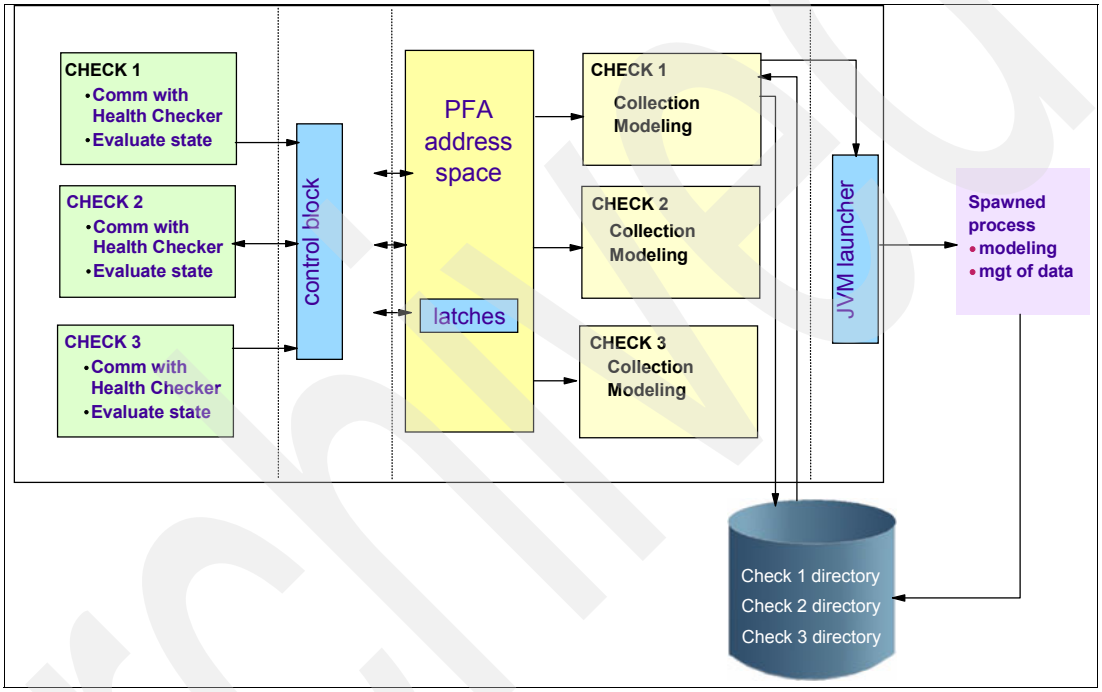


Figure 5-18 Overall PFA process

5.7 PFA installation

PFA is shipped with z/OS, starting in V1R11. It contains everything available in the SPE as well as two additional checks. For both releases, follow the install instructions carefully. Installation and configuration takes approximately 30 minutes. You are required to create a user and to run a provided installation script.

PTFs required for z/OS V1R10

The required PTFs are:

- APAR OA27165 (PTFs UA46241 and UA46243)
- A prerequisite MAPMVS APAR OA25773 (PTF UA43943)

PFA documentation is provided in *z/OS Problem Management*, G325-2564. For z/OS V1R10, documentation has been available online since March 2009.

Create a user ID

Create a user ID (for instance, pfauser) to define the location in the z/OS UNIX file system that stores the PFA data and connects the PFA user ID to an existing or new RACF group. The home directory of the user ID that owns the PFA started task must match where the install script is run.

If you are using PFA in a sysplex that shares file systems for z/OS UNIX, use a unique directory for each LPAR so that the event data that PFA writes to the file system is stored separately for each system.

Note: Do not use the same user ID that is assigned to the IBM Health Checker for z/OS.

Define the PFA started task

Define the PFA started task by creating a RACF profile for the pfauser with the following characteristics:

- OMVS segment with a UID parameter (for example, omvs(uid(7)))
- Home directory (for example, home(/pfa))
- PROGRAM path name of /bin/sh (for example, program(/bin/sh))

Copy the sample PFA procedure

Copy the sample PFA procedure, AIRPROC, from SYS1.SAMPLIB to the PFA member of SYS1.PROCLIB data set. If SMP/E does not write the executable code in the z/OS UNIX file system to PARM='path=(/usr/lpp/bcp)', then change the PARM value in AIRPROC to the path in which you store the executable code.

AIRSHREP

Under UNIX System Services (for example, OMVS as shown in Figure 5-19), you can see the files needed to install PFA in directory /usr/lpp/bcp.

```
ROGERS @ SC74:/u/rogers>ls -al /usr/lpp/bcp
total 2316
drwxr-xr-x   5 SYSPROG  OMVSGRP      608 Jul 25 17:38 .
drwxr-xr-x  58 SYSPROG  OMVSGRP     2080 Jul  8 14:28 ..
-rwxr-xr-x   2 SYSPROG  OMVSGRP    85930 Jul 25 17:38 AIRJCART.jar
-rwxr-xr-x   2 SYSPROG  OMVSGRP    41692 Jul 25 17:38 AIRJCHK.jar
-rwxr-xr-x   2 SYSPROG  OMVSGRP   233472 Jul 25 17:38 AIRLCSAC
-rwxr-xr-x   2 SYSPROG  OMVSGRP   249856 Jul 25 17:38 AIRLLARC
-rwxr-xr-x   2 SYSPROG  OMVSGRP   245760 Jul 25 17:38 AIRLMARC
-rwxr-xr-x   2 SYSPROG  OMVSGRP   233472 Jul 25 17:38 AIRLVSUC
-rwxr-xr-x   2 SYSPROG  OMVSGRP     5274 Jul 25 17:38 AIRSHKP.sh
-rwxr-xr-x   2 SYSPROG  OMVSGRP     9093 Jul 25 17:38 AIRSHREP.sh
drwxr-xr-x   2 SYSPROG  OMVSGRP     1344 Jul 25 17:38 IBM
drwxr-xr-x   7 SYSPROG  OMVSGRP      416 May 19 05:37 mca
drwxr-xr-x   2 SYSPROG  OMVSGRP      256 May 19 05:37 samples
ROGERS @ SC74:/u/rogers>
 ===>
```

Figure 5-19 Directory structure

From the user ID you have defined for PFA (for instance, pfauser), run the script AIRSHREP from directory /usr/lpp/bcp, as shown in Figure 5-20 on page 150.

```
ROGERS @ SC74:/u/rogers>/usr/lpp/bcp/AIRSHREP.sh
 All existing checks files and directories removed.
 Successfully created the Common Storage Usage Check Directory Structure
 Successfully created the Logrec Arrival Rate Check Directory Structure
 Successfully created the Virtual Storage Usage Check Directory Structure
 Successfully created the Message Arrival Rate Check Directory Structure
 Successfully created and populated ini file for the Common Storage Usage Check
 Successfully created and populated ini file for the Logrec Arrival Rate Check
 Successfully created and populated ini file for the Virtual Storage Usage Check
 Successfully created and populated ini file for the Message Arrival Rate Check
ROGERS @ SC74:/u/rogers>ls -al
total 196
drwxr-x---  11 SYSPROG  SYS1         800 Aug 12 16:47 .
dr-xr-xr-x   7 SYSPROG  TTY            0 Aug 12 16:35 ..
-rw-------   1 SYSPROG  SYS1        1458 Aug 12 16:50 .sh_history
drwxr-xr-x   2 SYSPROG  SYS1         256 Sep 13  2006 00000100
drwxr-xr-x   5 SYSPROG  OMVSGRP     1120 Jul 26  2006 ITSO-link
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 PFA_COMMON_STORAGE_USAGE
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 PFA_LOGREC_ARRIVAL_RATE
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 PFA_MESSAGE_ARRIVAL_RATE
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 PFA_VIRTUAL_STORAGE_USAGE
drwxr-xr-x   2 SYSPROG  SYS1         256 Sep 13  2006 db2
drwxr-xr-x   2 SYSPROG  SYS1         256 Sep 13  2006 echo
drwxr-xr-x   2 SYSPROG  SYS1         256 Sep 13  2006 ims
ROGERS @ SC74:/u/rogers>ls -al PFA*
PFA_COMMON_STORAGE_USAGE:
total 8
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 .
drwxr-x---  11 SYSPROG  SYS1         800 Aug 12 16:47 ..
drwxrwxrwx   2 SYSPROG  SYS1         256 Aug 12 16:47 data
-rwxrwx---   1 SYSPROG  SYS1         748 Aug 12 16:47 ini

PFA_LOGREC_ARRIVAL_RATE:
total 8
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 .
drwxr-x---  11 SYSPROG  SYS1         800 Aug 12 16:47 ..
drwxrwxrwx   2 SYSPROG  SYS1         256 Aug 12 16:47 data
-rwxrwx---   1 SYSPROG  SYS1         748 Aug 12 16:47 ini

PFA_MESSAGE_ARRIVAL_RATE:
total 8
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 .
drwxr-x---  11 SYSPROG  SYS1         800 Aug 12 16:47 ..
drwxrwxrwx   2 SYSPROG  SYS1         256 Aug 12 16:47 data
-rwxrwx---   1 SYSPROG  SYS1         748 Aug 12 16:47 ini

PFA_VIRTUAL_STORAGE_USAGE:
total 8
drwxrwxrwx   3 SYSPROG  SYS1         320 Aug 12 16:47 .
drwxr-x---  11 SYSPROG  SYS1         800 Aug 12 16:47 ..
drwxrwxrwx   2 SYSPROG  SYS1         256 Aug 12 16:47 data
-rwxrwx---   1 SYSPROG  SYS1         748 Aug 12 16:47 ini
```

Figure 5-20 PFA installation

Installing PFA in a z/OS UNIX shared file system environment

If your context is a zFS shared file system environment, a z/OS UNIX file system can be created that is shared among members of the sysplex with directories that are local to the LPAR.

Follow these steps:

1. Define the file systems as one for each LPAR.
2. After defining the file systems for each LPAR, define a symbolic link (that is, a symlink) to the sysplex root. From the `root` directory, enter the command shown in Figure 5-21, using the UID you assigned to pfauser.

```
cd
ln -s \$SYSNAME/pfa pfa
```

Figure 5-21 Define symlink

   This results in a home directory of `/systemname/pfa`.
3. Create the PFA directory in each of the system directories by entering the following commands for each of your system names. For example, the command for system Z1 is **`mkdir /Z1/pfa:`** and so on for each system.
4. Create the new file systems (one for each system) and mount them at the appropriate system mount point. For example, for OMVSSPT.Z1.PFA.ZFS, the mount point is `z1/pfa/etc` for each system.
5. Place an entry in the SYS1.PARMLIB(BPXPRMxx) member to mount the new file systems during IPL. (If you do not want to wait until the next IPL, you can manually mount these file systems.) Use the UNMOUNT attribute on the BPXPRMxx parmlib member to unmount the file system when OMVS or the LPAR is taken down. The file system mount point is `/sysname/pfa`; see Figure 5-22.

```
SYS1.PARMLIB(BPXPRM00)
MOUNT FILESYSTEM('OMVSSPT.&SYSNAME..PFA.ZFS') TYPE(ZFS)
MODE(RDWR) MOUNTPOINT('/&SYSNAME./pfa') UNMOUNT
```

Figure 5-22 BPXPRMxx add-ons

5.8 Serviceability information

Originally, PFA was designed to be a black box; you were supposed to start it and forget it. However, it was later decided through client interaction that more information is needed to ensure that PFA is running correctly. Therefore, a comprehensive “modify PFA” command has been built to display status information for the PFA infrastructure as well as detailed status for each individual check.

When a check exception is issued, the reports previously described greatly assists in analyzing the problem. In addition, the PFA documentation outlines best practices for each check to provide advice on how to analyze the problem.

Each check has a `/data` directory in the pfauser’s home directory. For example, if the pfa user is “pfauser,” then the data directory for the CSA usage check is `/u/pfauser/PFA_COMMON_STORAGE_USAGE/data`.

The files needed by PFA to collect data, model data, and perform the check are found in the `/data` directory. These files are documented in the PFA documentation and can be used to help you analyze the exception data.

If a PFA problem is suspected, keep in mind that the `/data` directory also contains log files that might have additional debug information. If the PFA debug parameter is turned on, additional information will be found in the log files.

If a PFA problem is suspected, IBM service will likely request the last exception report and the `/data` directory for the check. Preferably, recreate the problem with debug turned on.

For a “debug” parameter that is check-specific, note the following points:

- Additional diagnostic information is generated in the log files when debug is turned on.
- This parameter is not the debug parameter available through Health Checker because the Health Checker parameter did not apply to all three major functions. Instead, it is a parameter listed in the check-specific parameters for each check.

If a problem is suspected in PFA itself, PFA can simply be stopped. In addition, if a problem is suspected in one of the PFA checks, that check can be deleted from Health Checker. Deleting only the check that is failing will allow the other checks to continue processing. Deletion can be performed by a command as shown in Figure 5-23.

```
f hzsproc,delete,check(ibmpfa,PFA_COMMON_STORAGE_USAGE)
```

Figure 5-23 Deleting a check from Health Checker

6

HCD and HCM enhancements

Hardware Configuration Definition (HCD) allows you to define I/O configurations to both the software and hardware from a single, interactive interface. To define configurations to the software and hardware, you use HCD to create an input/output definition file (IODF).

HCM presents an interactive configuration diagram that allows you to maintain not only the logical connectivity data in the IODF, but also the physical information about a configuration. The logical information in the IODF represents the operating system and the channel subsystem definitions. The physical information, such as cabinets, patchports, crossbar switches, cables, locations and so on, adds the infrastructure to the logical data.

This chapter describes the new enhancements in the HCD and HCM in the following areas:

- HCD and HCM support for IPv6
- HCM installation (now performed with InstallShield 2009)
- OSA-ICC support for D/T3215
- Purpose and use of HCM toleration support for multi-user access enabled IODFs
- Edit HCD profile options
- HCD reports improvements

6.1 HCD and HCM IPv6 connections support

HCD is a mandatory product that z/OS systems must use to manage their configuration. HCM is an optional product that you can use to provide a graphical interface running in a Windows platform, and then use to update the HCD system datasets or create graphical maps of your configuration.

Communications between Windows Vista and its server running in z/OS in previous z/OS releases used IPv4. However, z/OS V1R11 provides support to IPv6. z/OS V1R11 clients that have configured their TCP/IP and its Windows workstation to run with IPv6 can log on the HCM to run with IPv6.

6.1.1 HCM logon using IPv6

z/OS V1R11 HCM supports the Internet Protocol, Version 6 (IPv6), which you can use when establishing a communication session with HCD. It must be installed on your local workstation.

During the HCM login you can define the IP address or the host name resolved by an IPv6 name server in the dialog panel, as shown in Figure 6-1.

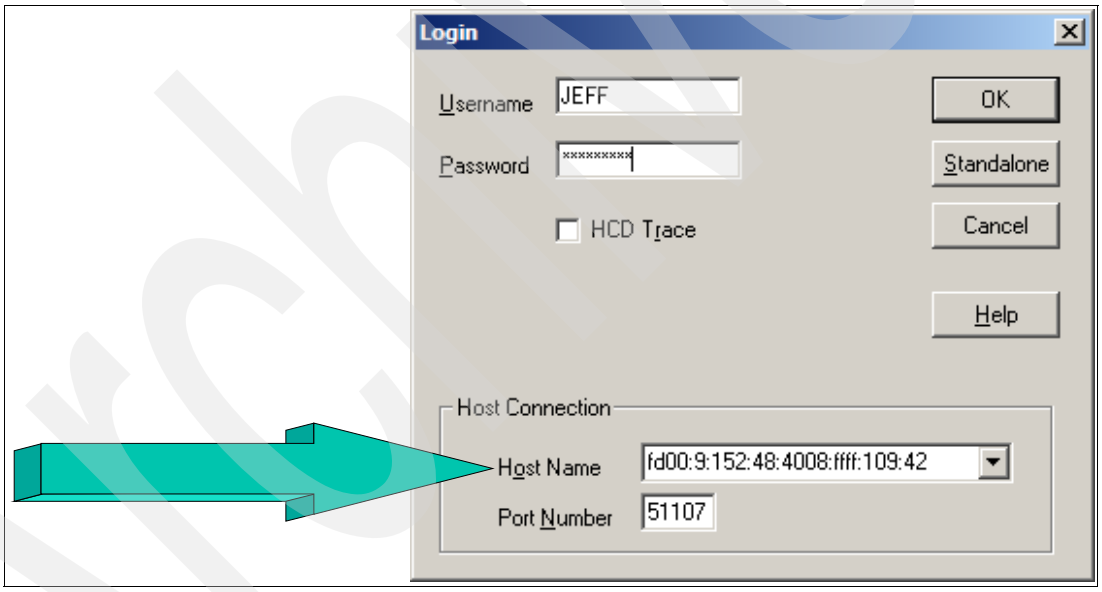


Figure 6-1 HCM login dialog panel

6.2 HCM installation support for Windows Vista

The installation process for HCM has changed. The code is delivered as an MSI package and is installed using the Windows Installer. The location of the EEQHCM.INI file has changed from the C:\WINDOWS user-specific application data directory. HCM creates separate versions if multiple users are working on the same workstation. Users have the benefit of their own preferred session settings. Due to changes in the user authorization methods for Windows versions, the change of the location is necessary to allow users who are administrators to update this file.

HCM lets you load a multi-user access enabled IODF from the HCD configuration file and lets you also open an existing HCM configuration with a multi-user access enabled IODF. You can use the viewing features of HCM on such configuration diagrams and you may make changes. However, you cannot update the associated IODF.

Before you can install HCM on your workstation, the code needs to be installed on the z/OS host using SMP/E.You can then start to download the HCM MSI installation package on your workstation as described here:

- The HCM MSI installation package is available in the data set member SYS1.SEEQINST(EEQINSTM) on the z/OS host.
  - Download the HCM MSI package in binary mode onto your workstation into a temporary folder and rename it to 'eeqinst.msi'.
- Install the HCM MSI package using the Windows Installer by double-clicking the package file or by issuing the command:

  ```
  msiexec /package eeqinst.msi
  ```

- The Windows Installer guides you through the installation. During theinstallation you are asked for the destination location of HCM. You can choose the default folder:

  ```
  C:\Program Files\IBM\Hardware Configuration Manager\
  ```

- After HCM is successfully installed on your system, you can delete the HCM MSI package. The HCM installation creates a program folder IBM Hardware Configuration Manager in your Start menu, from which you can launch HCM.

6.2.1 Differences using z/OS V1R11 HCM

With z/OS V1R11 HCM, users will notice the following changes:

- Starting with Windows XP, user management with groups and permissions was introduced to Windows. Default user groups are Users and Administrators. Users of group Users are not allowed to write in system directories (for example, `\Windows` or `\Program` files). As a desired result, only Administrators are allowed to perform software installations.

  An undesired effect is the use of the HCM ini file. File EEQHCM.INI used to be located in the `\Windows` directory. A normal user cannot write or update this file. Therefore, a normal user can work with HCM, but no user settings are saved (logon information, view preferences, column settings, filters, report preferences, and so on cannot be saved).

  With z/OS V1R11, the location of the EEQHCM.INI file changes from C:\WINDOWS to the user-specific application data directory. This does not affect your migration because the first invocation of HCM automatically copies the old INI file to the new location, or creates a new one if no INI file is found.

How PDFs are assigned to objects

A controller is assigned to a particular PDF in *one* of the following ways:

- When an HCM file is first created by loading an IODF (or when opening an existing file, which causes an IODF resync), HCM uses the “preferred PDF” for the controllers it automatically creates. The preferred PDF or a given control unit type is determined by searching the three controller PDF directories for a PDF with a matching control unit type.

  The CPDFA directory is searched first, then CPDFB and CPDFC. The preferred PDF comes from the first directory where a match is found. If more than one matching PDF is found in that directory, the one that comes first alphabetically will be used as the preferred PDF. If no PDFs are found in any directory, then HCM uses the default PDF, which is a single-segment controller with one channel interface labeled “A.”

- When a new controller is created within an existing HCM configuration, HCM will find all of the PDFs in all three directories that match the control unit type chosen by the user. If exactly one is found, then that one will be used. If two or more are found, the user will be prompted to choose the correct one. If none are found, the default DF is used.
- An existing controller can be edited and given another PDF. In this case, any PDF may be chosen, even ones that do not match the control unit type.

6.2.2 New functions for saving information

Pressing the corresponding Save button in the following dialogs lets you store the displayed information.

- **Save Port List button in the Switch dialog**

  With this button, the information displayed in the visible columns of the Ports section is stored in the specified file.

- **Save As button in List Devices for OS Configuration**

  With this button, you can store all information pertaining to the selected operating system in the specified file.

Physical description files (PDFs)

HCM provides a possibility to save the current layout of a controller as a model for other controllers. Users can change the physical description of a controller in the Edit Controller dialog.

From this dialog, HCM now offers the facility to save the layout of a tailored controller definition as a new PDF, which can be used for further controller definitions.

The Windows Installer guides you through the installation. During the installation you are asked for the destination location of HCM.

- You can choose the default folder:

  `C:\Program Files\IBM\Hardware Configuration Manager\`

- Or you can specify another folder by clicking the **Change** button.

> **Note:** If you install HCM into the default folder and want to use your own physical description files (PDFs), you must have administrator rights to access the default folder.
>
> Otherwise, you must install HCM into another folder to which you have write access.

6.3 HCD D/T3215 support

With prior releases, clients using z/TPF cannot define D/T3215 on the OSC channel path because it only supports D/T3270-X.

With z/OS V1R11, HCD allows the definition of D/T3215 devices or D/T 3270-X to an OSC channel and OSC control unit. A mix of both device types is *not* accepted. The IOCP needs to distinguish between OSC-3270 and OSC-3215 attachment. This is done using the CHPARM keyword on the CHPID statement. If a 3215 is attached to an OSC channel path, then CHPARM needs to specify value 40; otherwise, the value is not set (which means CHPARM=00).

The attachment of a 3215 device to the VM operating system is defined in VM UIM CBDUS260. The following error message can be displayed:

```
CBDG490I OSC control unit @1 can not be defined to both device types D/T3215
and D/T3270-X
```

HCD D/T3215 coexistence and migration

This support is been rolled back to z/OS V1R8 by APAR OA24287.

6.4 HCD support for MUA IODF files

With HCD for z/OS V1R9, multiple users can read an IODF simultaneously, but no user can read an IODF while it was accessed in update mode by another user. Also, a user can only update an IODF if no other user accessed the IODF in either read or update mode.

Starting with HCD for z/OS V1R10, when creating an IODF you can specify a multi-user access option in the Create Work I/O Definition File dialog shown in Figure 6-2. The default is single-user access.

```
                    Create Work I/O Definition File
The specified I/O definition file does not exist. To create a new
   file, specify the following values.

   IODF name . . . . . 'DOCU.IODF01.ZOS110.HCDUG.WORK'

   Volume serial number . ______ +

   Space allocation . . . 1024 (Number of 4K blocks)

   Activity logging . . . Yes (Yes or No)

   Multi-user access  . . No (Yes or No)

   Description  . . . . . ________________________________________________
                          ________________________________________________
                          ________________________________________________

   F1=Help F2=Split F3=Exit F4=Prompt F9=Swap F12=Cancel
```

Figure 6-2 Create an IODF data set with the MUA option

Having exclusive access to an IODF, users can also switch between single-user mode and multi-user access by using an option in the Change I/O Definition File Attributes dialog.

Note: To enable an IODF for multi-user access you need ALTER access authority. You can check whether the multi-user access property is enabled for an IODF by using View I/O definition file information from the Maintain I/O Definition Files task.

6.4.1 HCM support to read MUA IODF data set

z/OS V1R10 HCM provides information about whether an IODF is enabled for multi-user access, but it cannot open the IODF. An attempt to open such an IODF was rejected with error message CBDA650I.

z/OS V1R11 HCM can open a MUA-enabled IODF in read mode by using the MUA-toleration support. The configuration diagram can be viewed and reports can be drawn from the configuration, whether or not it is concurrently being updated in HCD. HCM users can perform physical changes to their HCM configuration file, but no logical changes to the IODF.

6.5 HCD edit profile option

Until now it was only possible to view, change, and set HCD profile options by directly editing the HCD profile data set outside of an HCD session. For users who do not use the new dialog to work on their HCD profile, nothing changes. They can continue to manage their profile keyword settings as before, editing the HCD profile data set outside of HCD.

However, starting with z/OS V1R11, users can invoke HCD `option 0. Edit profile options` as shown in Figure 6-3.

```
CBDPM000               z/OS V1.11 HCD
Command ===>
------------------------------------------------------------------

                          Hardware Configuration

Select one of the following.

0    0.  Edit profile options
     1.  Define, modify, or view configuration data
     2.  Activate or process configuration data
     3.  Print or compare configuration data
     4.  Create or view graphical configuration report
     5.  Migrate configuration data
     6.  Maintain I/O definition files
     7.  Query supported hardware and installed UIMs
     8.  Getting started with this dialog
     9.  What's new in this release

For options 1 to 5, specify the name of the IODF to be used.

I/O definition file . . . 'HCDI.IODF00.WORK1                    +
```

Figure 6-3 New dialog edit profile option

Using the new dialog edit profile option enables uses to perform the following tasks:

- View the actually set value of each profile keyword, whether it is set explicitly or defaulted by HCD.
- Edit the displayed values, which are presented in alphabetical order.
- Add other settings of profile keywords where multiple settings are allowed.
- Remove keyword settings or reset them to their HCD defined default.
- Add a comment of maximum length 32 to each setting of a profile keyword.
- Get online help information about the allowed value range for each individual keyword.

- Get prompt information for most keywords.
- Save changed values and comments back to the profile data set automatically when leaving HCD.

HCD profile options

If you select HCD option **0. Edit profile options**, a list of all HCD profile keywords with their current value setting is displayed, as shown in Figure 6-4 on page 159. For keywords that are not explicitly set, the default value is shown if it exists. Keywords are ordered alphabetically. The list is scrollable vertically.

Column A marks, for each profile option, whether a change to its value becomes active immediately (Y) or whether the new value is only available after the current HCD session has ended (N). Scrolling to the right displays user-added descriptions for the profile keyword settings. Value and description fields are editable if the user has write authority to the accessed HCD profile data set. If write authority is not granted, updates are denied by a message and settings can be viewed only.

Positioning the cursor to the value field of a keyword and pressing F1=Help displays keyword-related help what explains the purpose, default, and allowed values and syntax of values for this keyword. For most of the keywords, value prompt information is available. Most of the keywords may only be specified one time in an HCD profile. They are marked by a pound (#) sign in the Action column.

For keywords that may be specified multiple times (such as ALLOC_SPACE), two actions are implemented through the Action column:

- Entering `A` (Add) in the action column of a keyword that may be specified more than one time adds another assignment row for this keyword. The value field is initially empty and can be used to add an additional keyword value.
- Entering `D` (Delete) in front of a keyword defined multiple times removes the marked value definition from the list.

Changes made to the HCD profile options by using the new dialog are written back to the HCD profile data set when you leave the HCD session. The HCD profile, when edited within an HCD session, is written out in a new fixed format. Also use this format when editing the HCD profile data set outside of HCD, which you may still do as before.

```
                HCD Profile Options
CBDPPEF0                                          Row 1 of 44 More:        >
Command ===> ___________________________________________ Scroll ===> PAGE

Edit or revise profile option values.

HCD Profile : HCDI.HCD.PROFILE

/ Profile keyword           A Value +
# ACTLOG_VOL                Y _____________________________________________
_ ALLOC_SPACE               Y HCDASMP,70
_ ALLOC_SPACE               Y HCDIN,85
# BATCH_IODF_NAME_CHECK     Y YES
# BYPASS_UPD_IODF_FOR_SNA   Y NO
# CHANGE_LOG                N YES
# CHECK_IODF                Y YES
# CHLOG_VOL                 Y _____________________________________________
# COLOR_BACKGROUND          Y _____________________________________________
# CHLOG_EXTENSION           Y 0
# COLOR_HIGH                Y RED
# COLOR_NORM                Y BLUE
# COLOR_TEXT                Y PINK
_ CU_ATTACHABLE_DEVICE      N 3340,DUMMY
```

Figure 6-4 HCD edit profile options

HCD header profile

The header of the HCD profile shows the date and time when the profile was written and identifies the user ID that wrote it, as shown in Figure 6-5.

All profile keywords that are not set to their default value are listed in alphabetical order in the first profile section. Comments may follow each keyword and must start on the same row. Comment continuation lines must start with an asterisk (*) in column 1. Only the first 32 characters of a comment are recognized and displayed in the profile edit dialog. Any other comments between keyword assignment lines are not recognized when reading in the profile data set for display in the dialog.

HCD trace commands are saved in the second section. They cannot be modified within the edit profile dialog and are always left untouched in the profile data set.

```
/* ********************************************************************** */
/*                                                                        */
/* HCD Profile                                                            */
/* Created :  2008-12-13  16:05:13  by user : HCDI                        */
/*                                                                        */
/* ********************************************************************** */
/* ********************************************************************** */
/*                                                                        */
/*     HCD Profile Section for Standard Profile Options                   */
/*                                                                        */
/* ********************************************************************** */
CHECK_IODF = YES
COLOR_NORM = BLUE
COLOR_TEXT = PINK
CU_ATTACHABLE_DEVICE = 3350,DUMMY /* mapping 3350                         */
DEFAULT_PARSING_MACLIB = SYS1.MACLIB.TESTLAB /* overwrite default
*                                                 SYS1.NUCLEUS           */
DELAYED_GROUPING = YES
GCR_FORMAT = DCF
LINES_PER_REPORT_PAGE = 65
MAP_CUTYPE = NSCLCA1,NOCHECK
             :
/* ********************************************************************** */
/*                                                                        */
/*     HCD Profile Section for TRACE commands                             */
/*                                                                        */
/* ********************************************************************** */
trace on all level=255
trace off close reset
```

Figure 6-5 Header of HCD profile

6.6 HCD usability and productivity enhancements

HCD implemented several client requirements to become more user-friendly and increase the productivity of I/O configuration work.

Channel Path panels

Channel Path List panels CBDPCHF1 to CBDPCHF4 were enhanced to display the identifier of the processor for which the partition matrix is displayed. The label “Channel subsystem ID” has been shortened to “CSS ID.” Thus, clients do not have to keep track of the processor ID when entering these panels, as shown in Figure 6-6.

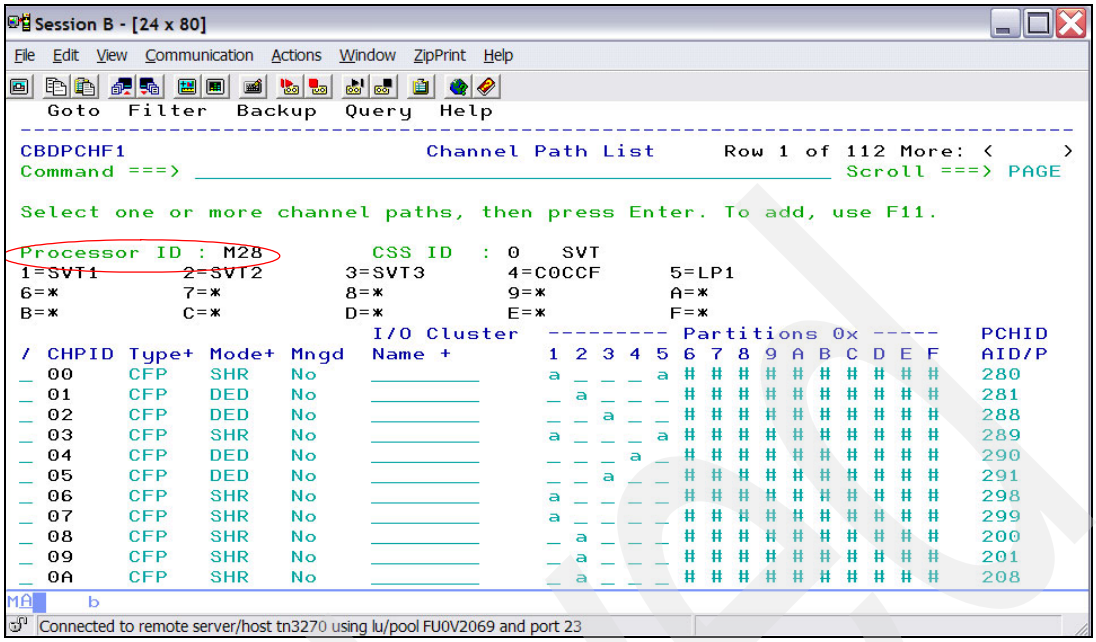


Figure 6-6 HCD Channel Path list identifies the processor ID

6.6.1 Changing the channel path definition

In the HCD dialog on the “Channel Path List” or “‘Change Channel Path Definition” panels, a user can overtype a channel path’s operation mode with another mode.

Changing the operation mode of a channel path is dependent on its type. For example, BL, BY, CVC, CBY, and CF receiver channel paths cannot be shared.

Before you can change the operation mode of a channel path, the rules for partition access and candidate lists of those channel paths that are attached to the affected logical control units must conform to the rules for the new operation mode. You must check which partitions have access to these channel paths.

When changing the channel path operation mode from SHR to REC or DED, you first must remove partitions in the appropriate access and candidate lists. The partition lists for the affected logical control units must be changed when the mode change has been done.

Previously, HCD automatically changed the operation mode to the mode matching the partition assignment without commenting this. Now, however, a new informational message (CBDA369I, shown in Figure 6-7 on page 162) is introduced which HCD issues whenever a user has explicitly typed in an operation mode that does not match the partition assignments.

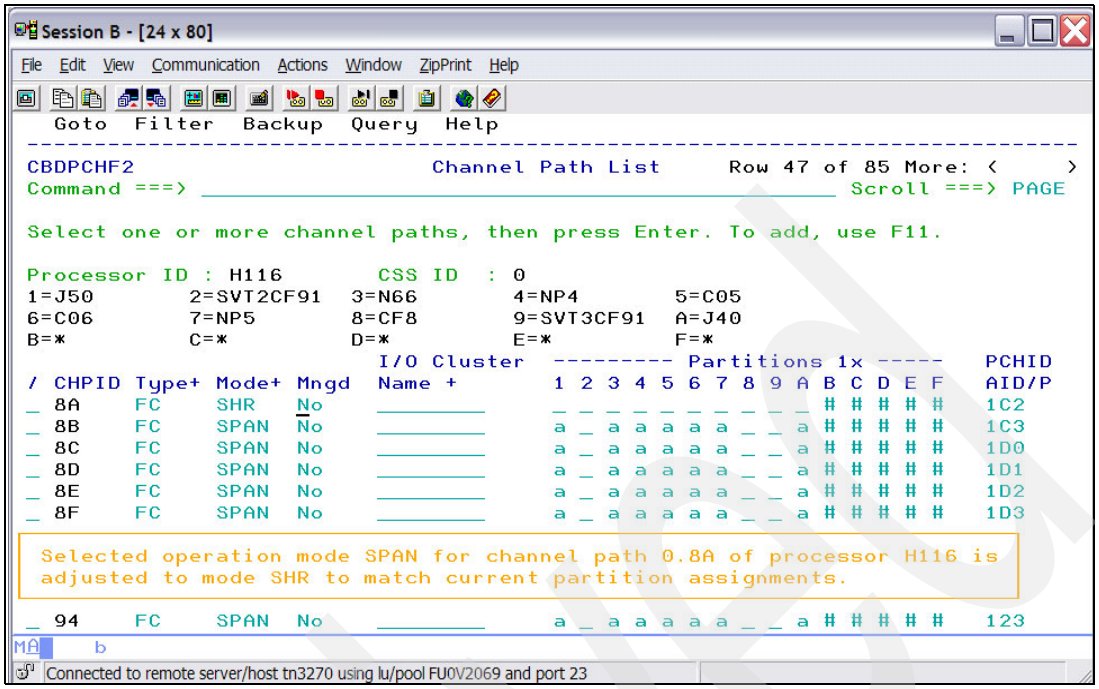


Figure 6-7 Message CBDA369I explains the HCD reaction when your change is not acceptable

Note: Messages displayed on the dialog panels do not show the message ID. To view the message help containing the message text with its number, press F1 (Help).

6.6.2 Deleting a partition confirmation panel

If a user deletes a partition which has CHPIDs exclusively assigned, HCD rejects the deletion because it requires that the channel path remains assigned to at least one partition.

The error message CBDA425I `Partition SCLM1 used by channel path 0.16 can not be deleted.` lists the first CHPID that has this condition. After having removed the CHPID from the partition, the error message can occur with a further CHPID fulfilling this condition, and so forth. Thus, deleting a partition can be an inefficient task.

Now, the Confirm Delete Partition panel that lists all CHPIDs that are connected to the partition flags all CHPIDs that are exclusively assigned to the partition to be deleted; see Figure 6-8 on page 163. Thus, all the flagged CHPIDs have to be removed at the same time and the partition can be deleted more efficiently.

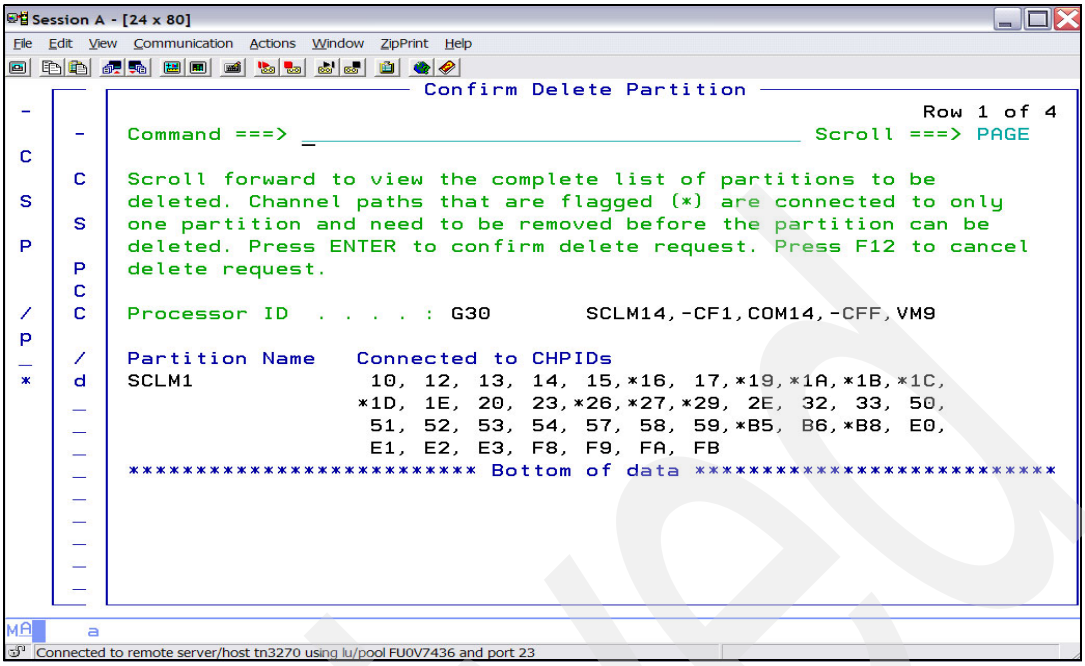


Figure 6-8 Message CBDA425i flags the channel path exclusively attached to partition to be deleted

6.6.3 CBDPUTDS panel enhancement

The IODF prompt panel CBDPUTDS is enhanced with column MUA to show information regarding whether an IODF can be accessed in single-user update mode only (No) or in multi-user access mode (Yes); see Figure 6-9.

If the data set is migrated, this value is not available in the catalog entry (as well as the size value) and therefore is not shown.

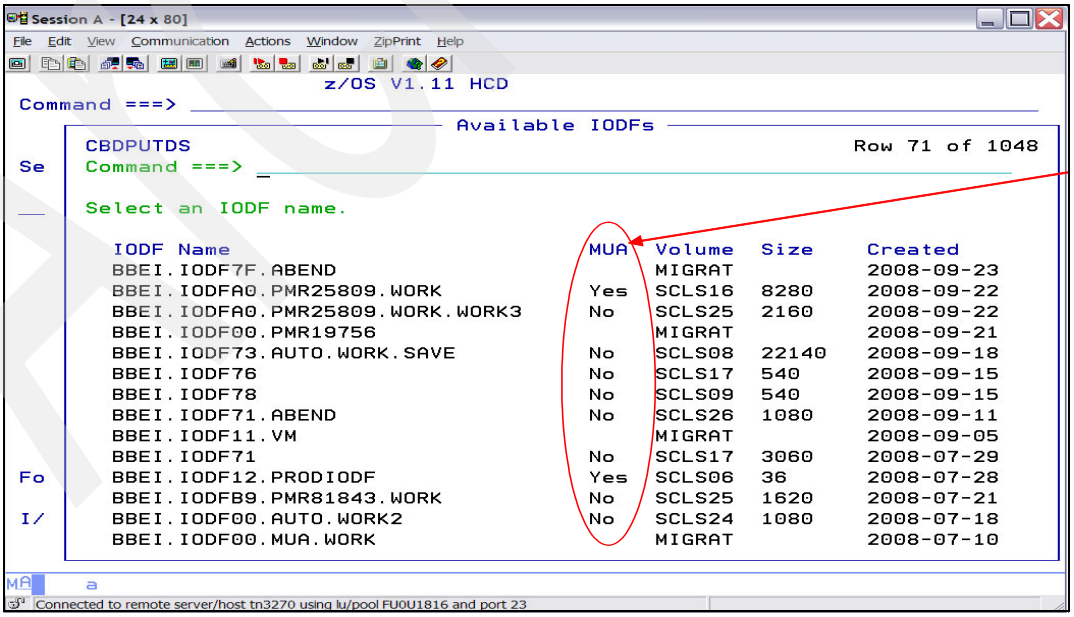


Figure 6-9 Panel CBDPUTDS identifies the MUA attribute of the IODF datasets

6.6.4 Creating a working IODF dataset with the MUA attribute

In z/OS V1R11, when creating a work IODF from a production IODF, the user has the option to set the work IODF directly in multi-user access mode rather than having to perform a separate Change IODF action if multi-user access mode is desired for the new work IODF; see Figure 6-10. The default value is No.

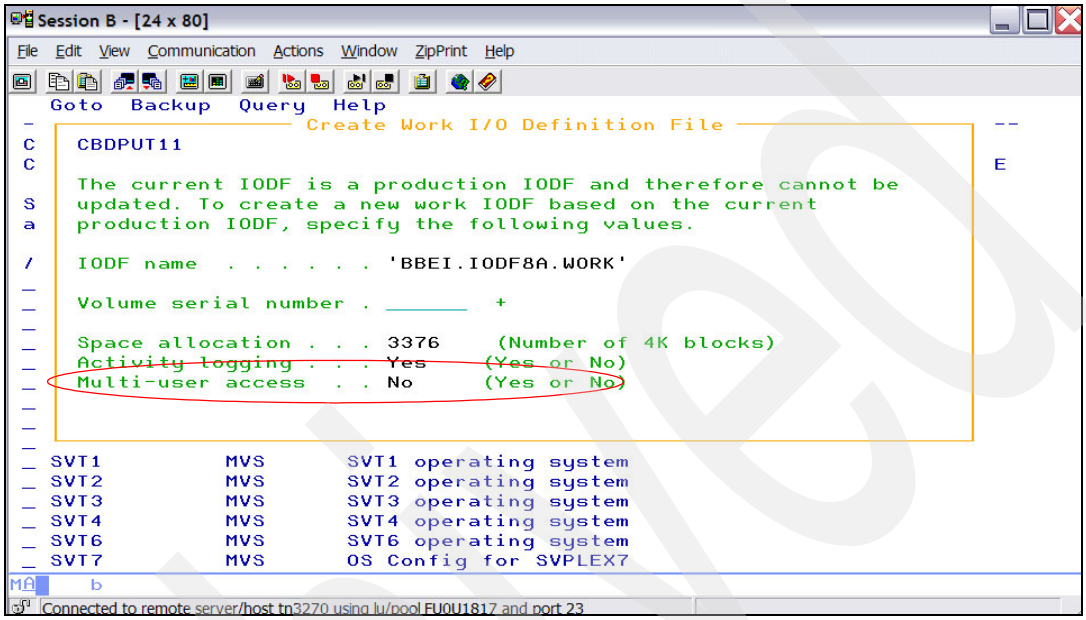


Figure 6-10 Option to create an IODF work file from production with MUA

Migration and coexistence considerations

This function was rolled back to z/OS V1R10 by APAR OA27720.

6.6.5 Control Unit summary report enhancement

When defining DASDs in the I/O configuration, clients need to keep track of the number of logical paths that may be established to the control units. If the maximum number is exceeded, HCD warns the user at Build Production IODF time. However, there was no function available to show the potential number of logical paths for the connections that were already defined. This information is essential to enable the client to see how many additional paths can be defined before reaching the maximum.

Now, the Control Unit Summary Report is enhanced to show the actual number of logical paths per control unit; see Figure 6-11 on page 165.

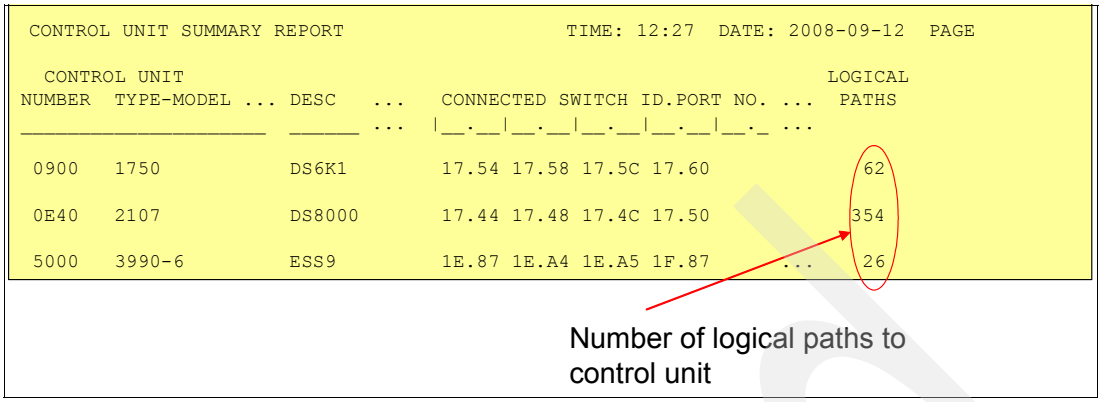


Figure 6-11 Control Unit Summary now reports the logical paths already defined for each CU

The Control Unit Detail Report is enhanced to also show the number of logical paths to a control unit per each channel subsystem, and the total number of logical path per controller port for each control unit; see Figure 6-12.

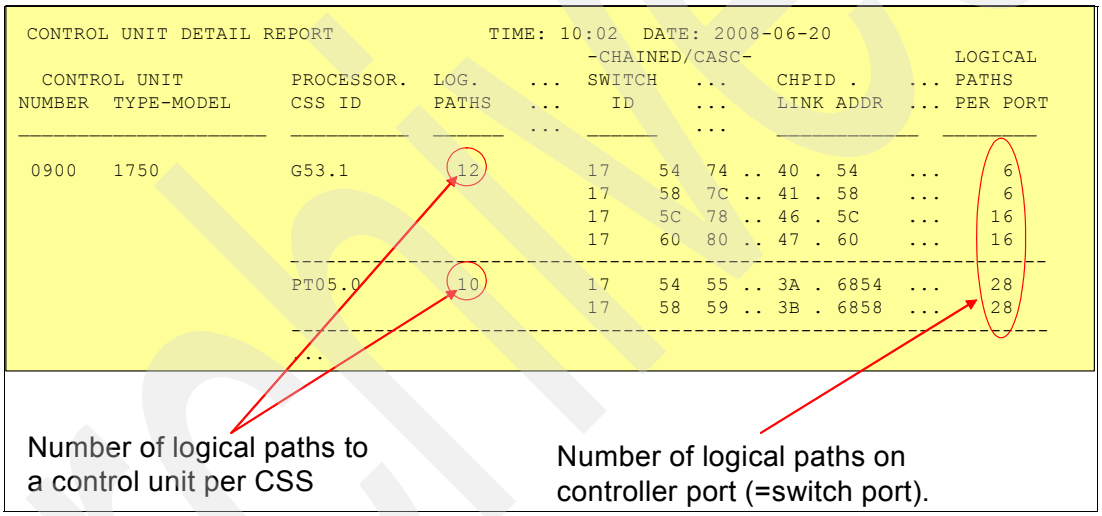


Figure 6-12 Control Unit Detail Report includes logical paths and logical paths per port information

6.6.6 EDT report enhancement

Up to z/OS V1R10, the EDT report only showed an esoteric if it had devices assigned. This did not give a complete picture of the EDT if it contained esoteric definitions without device assignments.

Now, the EDT Report is enhanced to show the esoteric name, type, VIO indication, and token for an empty esoteric; see Figure 6-13 on page 166.

Migration and coexistence considerations

This support is rolled back to z/OS V1R7 by APAR OA24437.

```
                                                E D T   REPORT
  OPERATING SYSTEM CONFIGURATION ID:  B710
  EDT IDENTIFIER: A1   DESCRIPTION:
                                                    AFFINITY  ALLOCATION   ASSOCIATED
NAME         NAME TYPE  VIO  TOKEN   PREF          INDEX    DEVICE TYPE  GENERICS
_________    _________  ___  _____  ______       ________  ___________  __________
POOLDASD     ESOTERIC                 280          FFFF      3010200F     3390
-------------------------------------------------------------------------------
RMMVTL       ESOTERIC                1000          0080      78048081     3490
-------------------------------------------------------------------------------
SHRPSS       ESOTERIC
-------------------------------------------------------------------------------
SYSDA        ESOTERIC    Y            280          FFFF      3010200F     3390


-------------------------------------------------------------------------------
TAPE         ESOTERIC                 950          0400      78048083     3590-1
                                     1000          0080      78048081     3490
```

Figure 6-13 EDT report includes empty esoteric

6.6.7 CSS/OS device compare report enhancement

Previously, the CSS/OS Device Compare Report was not written in a compact format. Device numbers with adjacent ranges of devices were not grouped together.

To improve the overview in the report, a more compact format has now been adopted that shows devices with adjacent ranges joined together. Additionally, the device range has been increased to show a maximum of four digits; see Figure 6-14.

```
  CSS/OS Device Compare   TIME: 07:11  DATE: 2008-04-21  PAGE S-1

        IODF Name : HCI.IODF00.CSSOS

Processor Id: PROC01  CSS Id: 0  Partition Name: PART01  ./.
                 Operating System Configuration Id: OS01

|-------------------------|--------------------|--------------------|
| Device, Range           | CSS Device Type    | OS Device Type     |
|-------------------------|--------------------|--------------------|
| A000,80                 | 3390B              | same               |
| A050,176                | 3390A              | same               |
| A200,80                 | 3390B              | same               |
| E210,16           -     |                    | FCTC               |
| E220,208                | FCTC               | same               |
|-------------------------|--------------------|--------------------|
```

Figure 6-14 CSS/OS device report lists the consecutive devices by range

Migration and coexistence considerations

This support was rolled back to z/OS V1R7 by APAR OA25199.

6.6.8 IODF processor compare report enhancement

Previously, processor tokens were compared in the Processor Compare Report only if the two IODFs used for the comparison were production IODFs. For a work IODF, the string “n/a” was shown.

Not this is enhanced to always compare the processor token of two IODFs whether they are work IODFs, production IODFs, or a combination of both; see Figure 6-15.

```
                                Processor Compare Report

              New IODF name:  BBEI.IODF8A.WORK                      Old IODF name:  BBEI …


|------------------------------|----------------------|----------------------|---------------
| PROC                         | New IODF             | Old IODF             | Description
|------------------------------|----------------------|----------------------|---------------
| G127                         | Actual Data          | Old Data             |
|                              |----------------------|----------------------|---------------
|                              | LPAR                 | same                 | Processor Conf
|                              | 2084-A08             | 2084-B16             | Processor Type
|                              |                      | same                 | Processor Seri
|                              | IBM390PS             | same                 | Processor Netw
|                              | G127                 | same                 | Processor CPC
|                              | Old name for Danu  + | same                 | Processor Desc
|                              | T102                 |                      |
|                              | 8080000A376F209709 + | 0080000A376F209709 + | Processor Token
|                              | -02-1716:45:34       | -02-1716:45:34       |
|                              |                      | same                 | Processor Local
|------------------------------|----------------------|----------------------|---------------
```

Figure 6-15 Processor Compare Report shows token for any IODF dataset type comparison

Migration and coexistence considerations

This function was rolled back to z/OS V1R10 by APAR OA25199.

6.6.9 IOCP deck generation enhancement

When HCD generates an IOCP deck, it uses IOCP defaults for the PARTITION/NOTPART keywords on the CHPID and IODEVICE statements to keep the size of the IOCP deck as small as possible.

Note the following conditions:

- The access list of a channel path for a channel subsystem is equal to the partition list of the CSS
- Or, a device has no explicit device candidate list defined for a channel subsystem (and uses the default device candidate list)

If these conditions exist, then HCD generates no CSS clause on the PARTITION/ NOTPART keyword of the corresponding CHPID or IODEVICE statement for that channel subsystem. This makes it tedious for clients using the generated IOCP deck to check their I/O configuration for proper partition assignments.

Now a new profile option, SHOW_IOCP_DEFAULTS (in HCD option 0), has been introduced that you can use to explicitly show and document the partition defaults of CHPID and IODEVICE statements in generated IOCP decks.

When this option is set to YES:

- The default access list is generated for spanned channel paths for a channel subsystem,
- The default device candidate list is generated for devices without an explicit device candidate list for a channel subsystem and written as a comment following the CHPID or IODEVICE statement with an additional PARTITION/NOTPART keyword.

  The comment starts with tag *$DFLT$. The default for this option is NO, which means that the additional comments are not generated.

The profile option is only applicable for generating IOCP input data sets (option 2.3 of HCD), and not for generating I/O configuration statements (option 2.10 of HCD).

```
         CHPID PATH=(CSS(0,2),0A),SHARED,SWITCH=51,PCHID=3F0,TYPE=FC
*$DFLT$         PARTITION=((CSS(0),(LP1501,LP1502),(=)))
*$DFLT$         PARTITION=((CSS(2),(LP1521),(=)))
         CHPID  PATH=(CSS(0,2),0B),SHARED,                              *
                PARTITION=((CSS(0),(LP1501,LP1502),(LP1503)),           *
                SWITCH=52,PCHID=3F8,TYPE=FC
*$DFLT$         PARTITION=((CSS(2),(LP1521),(=))
         CNTLUNIT CUNUMBR=9B11,PATH=((CSS(0),D1),(CSS(1),D1)),CUADD=0, *
               UNIT=IQD
         IODEVICE ADDRESS=(9B10,010),CUNUMBR=(9B11),UNIT=IQD
*$DFLT$         PARTITION=((CSS(0),TPCW,WSTCPROD,WSTHPROD,WSTPPROD)
*$DFLT$         PARTITION=((CSS(1),LINP,LINZ))
         IODEVICE ADDRESS=(9B1A,230),CUNUMBR=(9B11),                    *
               PARTITION=((CSS(1),LINP,LINZ)),UNIT=IQD
*$DFLT$         PARTITION=((CSS(0),TPCW,WSTCPROD,WSTHPROD,WSTPPROD))
```

Figure 6-16 IOCP deck generated with option SHOW_IOCP_DEFAULTS set to YES

Migration and coexistence considerations

This function was rolled back to z/OS V1R7 by APAR OA26729.

6.6.10 Message CBDA816I enhancement

The HCD dynamic reconfiguration facility can become disabled if the tokens do not match. Message CBDA816I has been changed so that the date and time of the IODF tokens are shown in externalized format. This makes it easier for users to see the differences in the tokens and thus in the IODF versions; see Figure 6-17.

```
CBDA816I No configuration changes are allowed. Active I/O
 definition does not match IODF BBEI.IODF0B. IODF token:
 00800001C38F2097 00:27:49 08-08-28, active token:
 00800001C38F2097 02:23:14 08-08-28.
```

Figure 6-17 CBDA816I message with externalized token format

7

DFSMShsm enhancements

DFSMShsm is a functional component of the DFSMS family, which provides facilities for managing your storage devices. DFSMShsm is a DASD storage management and productivity tool for managing low-activity and inactive data. It relieves you from having to handle manual storage management tasks and improves DASD use by automatically managing both space and data availability in a storage hierarchy.

DFSMShsm cooperates with the other products in the DFSMSdfp family to provide efficient and effective storage management. DFSMSdfp provides a storage management subsystem (SMS) that allows storage administrators to control the use of storage. The storage management subsystem provides storage groups, storage classes, management classes, and data classes that control the allocation parameters and management attributes of data sets.

DFSMShsm performs space management and availability management of each data set as directed by the management class attributes of that data set. In addition, the storage group controls the allocation of the data set when DFSMShsm returns the data set to level 0 (L0) storage.

This chapter describes DFSMShsm enhancements:

- Data set backup retention period
- ML1overflow
- Fast replication enhancements

7.1 Data set retention period

Availability management is the function of the DFSMShsm program used to ensure that you can retrieve usable copies of data sets that become lost or damaged. DFSMShsm can create backup versions of data sets either automatically or by command.

Although existing functions allow you to define a retention period in the SMS panel, if the number of backup versions exceeds the maximum number of backup versions defined in the SMS panel or with the **SETSYS** command, then the backup version will be rolled off. This means that the oldest is deleted.

Now, however, DFSMShsm enhancements allow you to specify a retention period on the data set **backup** command. This allows you to keep an individual backup copy for either a shorter- or longer-than-normal period of time.

7.1.1 z/OS V1R11 enhancements for retention period

DFSMShsm maintains backup copies as active backup copies and retained backup copies. Active copies are the backup copies that have not yet rolled off. Retained copies are the backup copies that have rolled off from the active copies, but have not yet reached their retention periods.DFSMShsm can maintain a maximum of 100 active copies.

DFSMShsm can maintain more than enough retained copies for each data set to meet all expected requirements. Roll-off processing occurs when the number of existing active copies exceeds the maximum number of active backup versions, as defined by the management class attribute Number of backup versions (data exists) for SMS-managed data sets, and the value of the SETSYS VERSIONS command, for non SMS-managed data sets.

When a new backup version causes the maximum number of active copies to be exceeded, DFSMShsm checks all of the active copies to see if any active copy has retained days specified, and if the retained days value is met. The active backup copies are then rolled off in the following order:

1. Roll-off processing removes all backup versions that have expired RETAINDAYS values.
2. If at least one active backup copy has retain days specified, but none of the active backup copies have met their retain days:
   a. The oldest active backup copy is rolled-off if it does not have a RETAINDAYS value specified.
   b. If the oldest active backup copy has RETAINDAYS value specified, then the oldest copy is replaced by the new backup copy, and the replaced active copy is maintained as a retained backup copy.
3. If none of the active backup copies have RETAINDAYS values specified, the excess versions are deleted, starting with the oldest version. The maximum number of active backup copies is 100.

The management class retention period defines the maximum number of days to maintain a backup copy. RETAINDAYS is a new keyword that defines the minimum number of days to maintain a backup copy. This value takes precedence over any existing retention period setting. In z/OS V1R11, the **(H)BACKDS** command is enhanced to allow you to specify a RETAINDAYS keyword to create a backup copy with a specified retention period.

DFSMShsm keeps valid backup versions as long as you want, even for an indefinite time. The following values are the valid maximum allowable numbers of backup versions for different BCDS record lengths:

- Record length of 2040 to 6543 --29 versions maximum
- Record length of 6544 or more - 100 versions maximum

7.1.2 RETAINDAYS keyword

RETAINDAYS is an optional parameter specifying a number of days to retain a specific backup copy of a data set. If you specify RETAINDAYS, the number of retain days is a required parameter that specifies a minimum number of days (0-50000) that DFSMShsm retains the backup copy.

You specify the RETAINDAYS keyword in the following manner:

99999 If you specify 99999, the data set backup version never expires.

50000+ Any value greater than 50000 (and other than 99999) causes a failure with an ARC1605I error message.

0 A retain days value of zero (0) indicates:

- The backup version can expire within the same day it was created if EXPIREBV processing takes place or when the next backup version is created.
- The backup version is kept as an active copy before roll-off occurs.
- The backup version is not managed as a retained copy.

Restriction: RETAINDAYS applies only to cataloged data sets.

You can keep an individual backup copy for either a shorter or longer than normal period of time by specifying the RETAINDAYS keyword on the **BACKDS** command. RETAINDAYS controls the minimum number of days that a backup copy of a data set is maintained. DFSMShsm uses the RETAINDAYS value to determine when a backup copy is to be rolled-off during EXPIREBV processing.

Note: If you specify RETAINDAYS with an uncataloged data set, **BACKDS** processing fails with an ARC1378I error message.

EXPIREBV command

The **EXPIREBV** command is used to delete unwanted backup and expired DFSMShsm aggregate backup and recovery support (ABARS) versions of SMS-managed and non-SMS-managed data sets from DFSMShsm-owned storage.

The optional parameters of the **EXPIREBV** command determine the deletion of the backup versions of non-SMS-managed data sets. The management class attributes determine the deletion of backup versions of SMS-managed data sets. The management class fields “Retain Extra Versions” and “Retain Only Version” determine which ABARS versions or incremental backup versions are deleted.

The RETAINDAYS keyword specified on the data set backup request determines how long a data set backup copy is kept for both SMS-managed and non-SMS-managed data sets.

When you enter the **EXPIREBV** command, DFSMShsm checks the retention days for each active backup copy for each data set, starting with the oldest backup version and ending with the third newest version. If the version has a specified retention days value, DFSMShsm calculates the age of the version, compares the age to the value of the retention days, and expires the version if it has met its RETAINDAYS.

The second-newest version is treated as though it had been created on the same day as the newest backup version, and is not expired unless the number of retention days specified by RETAINDAYS have passed since the creation of the newest backup version. EXPIREBV does not process the newest backup version until it meets both the management class retention values and the RETAINDAYS value.

During **EXPIREBV** processing, DFSMShsm checks the retention days for each retained backup copy. The retained copy is identified as an expired version if it has met its retention period. The **EXPIREBV DISPLAY** command displays the backup versions that have met their RETAINDAYS value. The **EXPIREBV EXECUTE** command deletes the backup versions that have met their RETAINDAYS value.

7.1.3 Retained copy

As mentioned, when a retention period creates the need to manage a backup copy after it rolls off from the maximum number of active copies, DFSMShsm manages the copy as a retained copy. Retained copies are the set of backup copies that have rolled off from the active copies and have not yet reached their retention periods. This allows you to maintain an indefinite number of retained copies for each data set.

Note the following points:

- The management class retention period defines the *maximum* number of days to maintain a backup copy.
- RETAINDAYS now defines the *minimum* number of days to maintain a backup copy.

Specifying retention periods

DFSMShsm now allows you to specify the RETAINDAYS value in the data set backup through the following paths:

```
BACKDS, (H)BACKDS, ARCHBACK, ARCINBAK, HBACKDS through ISMF panel.
```

If RETAINDAYS is specified on the command, the number of retain days is a required parameter that must be an integer number in the range of 0 to 50000, or 99999 which indicates that the backup copy should never expire.

You can use one of the following methods to specify a retention period for a copy of a backup data set using the new RETAINDAYS keyword:

- By using the **(H)BACKDS** command

  Figure 7-1 shows an example new **H(BACKDS)** command.

```
(H)BACKDS dsname RETAINDAYS(100)
```

Figure 7-1 (H)BACKDS command with RETAINDAYS parameter

- By using the ARCHBACK macro

- By using the ARCINBAK program

  The ARCINBAK program is updated to allow the request of a backup of one or more data sets in a batch environment with the new RETAINDAYS keyword. A five-character value must be specified as a parameter of the RETAINDAYS keyword. The RETAINDAYS parameter must be an integer number in the range of 0 to 50000, or 99999 which indicates that the backup copy should never expire. Figure 7-2 on page 173 shows sample JCL to invoke ARCINBAK.

```
//JOBNAME JOB . . .,USER=USERID,PASSWORD=USERPSWD
//STEP1    EXEC PGM=USERPGM
//SYSPRINT DD SYSOUT=A
//DSET1    DD DSN=USERID.NO3.GDG(-1),DISP=OLD
//DSET2    DD DSN=USERID.NO3.PSFB,DISP=OLD
//DSET3    DD DSN=USERID.NO4.PSFB,DISP=OLD
//DSET4    DD DSN=USERID.NO3.KSDS,DISP=OLD
//*
//STEP2    EXEC PGM=ARCINBAK,PARM=('RETAINDAYS(365),
//      TARGET(TAPE),CC=(PREFERRED,PHYSICALEND)')
//ARCPRINT DD SYSOUT=A
//ARCSNAP  DD SYSOUT=A
```

Figure 7-2 ARCINBAK sample JCL

Note: For more information about using the **(H)BACKDS** command, see *z/OS DFSMShsm Storage Administration*, SC35-0421. For more information about using the ARCHBACK macro, see *z/OS DFSMShsm Managing Your Own Data*, SC35-0420.

7.1.4 Retained backup copies

DFSMShsm with z/OS V1R11 maintains backup copies as active backup copies and retained backup copies. DFSMShsm can maintain a maximum of 100 active copies. DFSMShsm can maintain more than enough retained copies for each data set to meet all expected requirements.

Roll-off processing occurs when the number of existing active copies exceeds the maximum number of active backup versions, as defined by the management class attribute number of backup versions (data exists) for SMS-managed data sets, and the value of the SETSYS VERSIONS command, for non SMS-managed data sets. When a new backup version causes the maximum number of active copies to be exceeded, DFSMShsm checks all of the active copies to see if any active copy has retained days specified, and if the retained days value is met. The active backup copies are then rolled off in the following order:

1. Expiration date coded
2. Base SCRATCH or NOSCRATCH
3. Management class EXPIRE/MIGRATE

In Figure 7-3 on page 174, the maximum number of backup versions (specified in the SMS management class or on the **SETSYS** command) is set to three, so there are three active backup versions of data set My.Data. However, the maximum number of backup versions can be set to any value from 1 to 100. An unlimited number of retained backup copies can be maintained for each data set.

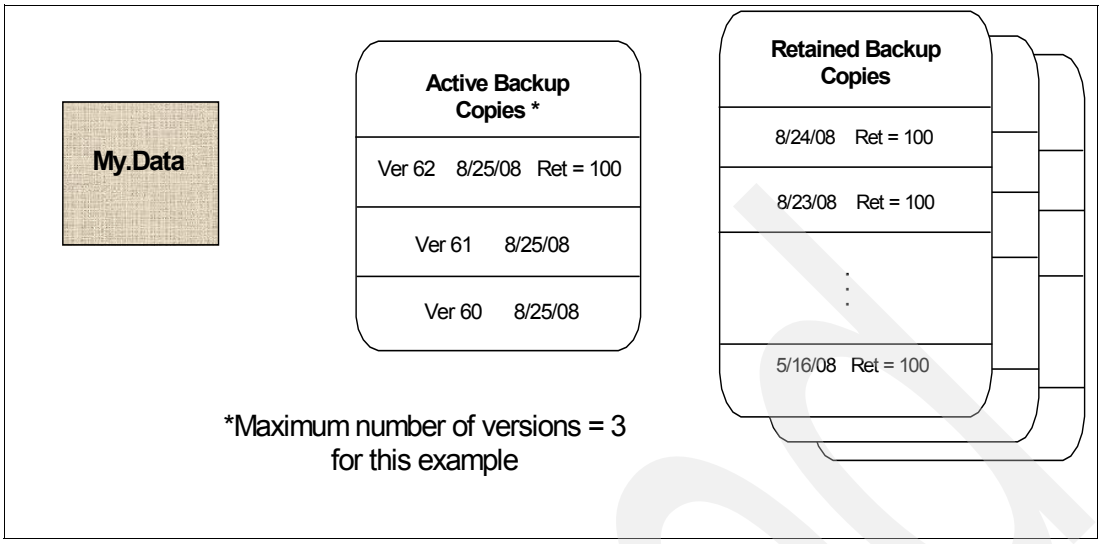


Figure 7-3 DFSMShsm active backup and retained backup copies of My.Data

7.1.5 Retained copy expiration

Retained copies can only be removed with the **EXPIREBV** command (or manually removed with the **BDELETE** command). The **EXPIREBV** function check for an exceeded retention period will take precedence over all existing criteria when expiring active and retained backup copies. Only **EXPIREBV** or **BDELETE** commands can remove those retained backup versions.

Note: The RETAINDAYS value takes precedence over any existing retention period setting.

7.1.6 Retained copy deletes

The new keyword ALL was added to the **(H)BDELETE** command. The ALL keyword is mutually exclusive with the existing VERSIONS and DATE keywords. The **(H)BDELETE** command will fail with a parse error if none of the mutually exclusive keywords ALL, VERSIONS, or DATE is specified. Keep the following specifications in mind:

When ALL is specified, DFSMShsm deletes all backup versions of the specified data set except for the retired versions if they exist, including both active and retained backup copies.

- DATE and TIME can be used to delete both active and retained backup versions that match the criteria.
- The VERSIONS keyword applies to active backup versions only.
- If multiple data sets are specified on the **(H)BDELETE** command, the ALL keyword will apply to all specified data sets.
- If a partially qualified data set is specified, the ALL keyword will apply to all of the data sets that meet the filter criteria.

Figure 7-4 on page 175 shows (**H)BDELETE** command examples.

```
(H)BDELETE(dsname) DATE(yyyy/mm/dd) TIME(hhmmss) or
(H)BDELETE(dsname) ALL
```

Figure 7-4 DATE, TIME, and ALL keywords added to the (H)BDELETE command

ARCHBDEL macro

The new parameters of the ARCHBDEL macro are:

ALL When ALL=YES is specified, DFSMShsm deletes all backup versions for the specified data set except for the retired versions if they exist, including both active and retained backup versions.

DATE This parameter specifies the address of a field that contains the date when the backup version to be deleted was created. For dateaddr, substitute the address of a 7-byte data area that contains the backup date. The field referenced by dateaddr must contain 7 digits in the form of yyyyddd.

TIME This parameter specifies the address of a field that contains the time when the backup version to be deleted was created. For timeaddr, substitute the address of a 6-byte data area that contains the backup time. The field referenced by timeaddr must contain 6 digits in the form of hhmmss.

VERS This parameter specifies a list of versions to delete. Specify the address of a structure that contains a 2-byte number of entries in the list, followed by entries (of 2 bytes each) containing the version number (001-999) that you want to delete. You can use the ARCXTRCT macro to extract data for version numbers that currently exist. You cannot use the VERS keyword to delete retained backup copies.

Note: The DATE and TIME keywords must be specified together, otherwise the macro invocation will fail. The keywords ALL, VERS, and DATE are mutually exclusive, and one of them *must* be specified on the command or the ARCHBDEL invocation will fail. The VERS parameter is valid against active backup versions only.

To delete a retained copy, use the DATE and TIME keywords. Use the **LIST** command or ARCXTRCT macro to extract data from DFSMShsm to obtain the backup version creation date and time for available backup versions.

7.1.7 Retained copy recovery

To recover an active backup version with retain days specified at the time of BACKDS processing, you can use the existing **RECOVER** command. To recover a retained backup version, you must use the DATE and TIME keywords. Retained copies can only be recovered by specifying the date, and optionally the time, that the version was created. Version and generation cannot be used for recovering retained copies.

The DATE keyword must be used to recover retained backup copies. Use the new optional TIME keyword to recover a specific copy when multiple retained copies with the same date exist. The DATE and TIME keywords can also be used to recover active versions. If the TIME keyword is specified, the DATE keyword must also be specified or the **(H)RECOVER** command will fail.

For hhmmss, substitute the time when the backup version to be recovered was created. The range for hours is 00-23, and minutes and seconds is 00-59. If TIME is specified, DFSMShsm will recover the backup copy with the specified date and time. Consider the following:

- If a copy with the specified date and time does not exist, the command will fail.

- If only DATE is specified, current processing will continue to take place the most recent version created on, or prior to, the specified date will be recovered.

Attention: TIME(hhmmss) is an optional sub-parameter of DATE that specifies the exact time in hours, minutes, and seconds (hhmmss) when the backup version is created. The valid range for hours is 00-23, and for minutes and seconds is 00-59. If you specify TIME, you must also specify DATE, otherwise the RECOVER command fails. You must specify a fully qualified data set name. You cannot specify TIME on a volume recovery request. If a version with specified date and time cannot be found, the command fails.

Figure 7-5 shows an example of the **HRECOVER** command.

```
(H)RECOVER(dsname) DATE(yyyy/mm/dd) TIME(hhmmss)
```

Figure 7-5 (H)RECOVER command example

Important: To recover a retained backup version specify the date, and if necessary, the time that the retained backup copy was created. You cannot recover a retained copy with the GENERATION and VERSION keywords.

ARCHRCOV macro

TIME is a new optional keyword of the ARCHRCOV macro. If TIME is specified without DATE, the macro invocation will fail. TIME specifies the address of a field containing the time of creation of a backup version within a specified DATE that you want to recover. For timeaddr, substitute the address of a 6-character data area containing the time in the form hhmmss. The parameters have the following specifications:

- The range for hours is 00-23, and minutes and seconds is 00-59.
- If TIME is specified, DFSMShsm will recover the backup copy with the specified date and time.
- If a copy with the specified date and time does not exist, the request will fail.
- If only DATE is specified, DFSMShsm will follow its current practice of recovering the most recent backup version created on or prior to the specified date.

Note: When **(H)RECOVER** is issued without VERSIONS,GEN, or DATE, TIME, the most recent active copy will be recovered.

Recovery considerations

External output includes DCOLLECT, ARCXTRCT, FIXCDS, LISTING OUTPUT, FSR record, and MESSAGE updates.

ARCXTRCT has two new optional keywords, DATE and TIME, which are added to the syntax. When specifying these parameters they must be specified together, and they are only valid with the DATA=BUVERS form of the macro. They are ignored if specified with any other form of the macro. For dateaddr, substitute the address of a 7-byte data area containing the date in the form yyyyddd. For timeaddr, substitute the address of a 6-byte data area containing the time in the form hhmmss. The range for hours is 00-23, and minutes and seconds is 00-59.

When DATE and TIME are specified, DFSMShsm will return information up to 100 versions created before the specified DATE and TIME.

If there are additional versions that can be listed, a new flag (BUVEFMAS) in the last backup version entry will be set ON to indicate more versions exist and can be listed. If the you want to see the additional versions beginning from the point the last invocation ended, the DATE and TIME of the last version listed need to be specified as the DATE and TIME parameters and the request must be resubmitted.

The **FIXCDS** command can be used to DISPLAY, VERIFY, CREATE, and PATCH the new MCBR record (type=Z), by specifying the original user data set name as the key. The new INDEX option added to the **FIXCDS** command applies only to Z-record processing and must be used to indicate the nth entry of the MCBR record set. There can be multiple MCBR records associated with a single data set and each MCBR record contains up to 100 entries. The CREATE keyword is valid only with a hex-key. The range of the INDEX is from 1 to 65535. The default value of INDEX is one (1),which is the most recent MCBR record.

The **(H)LIST DSNAME** and **(H)LIST LEVEL** commands have added two news parameters, RETAINDAYS and ACTIVE SELECT. When ACTIVE is specified, only the backup information of the active copies of the data set and the total number of backup copies will be displayed. If RETAINDAYS is specified, all retained and active backup copies with RETAINDAYS specified at the time of the data set backup will be provided. Version and generation information will not be displayed for the retained backup copies. If one of the new SELECT options is not specified, then both the active and retained backup copy information will be shown in the output and RETDAYS=*****, will be displayed in the output.

Figure 7-6 shows an example of the **(H)LIST** command.

```
(H)LIST LEVEL(qualifier) SELECT(RETAINDAYS|ACTIVE) BCDS/BOTH
(H)LIST DSNAME(dsname) SELECT(RETAINDAYS|ACTIVE) BCDS/BOTH
```

Figure 7-6 (H)LIST command example

Messages update

(H)BACKDS: ARC1334I, ARC1378I, ARC1381I, and ARC1605I

(H)RECOVER: ARC1068I.

(H)BDELETE: ARC0182I, ARC0185I, and ARC0189I.

Facility class profiles

Users must have READ access authority to the profile to use a command or a command and parameter. A security administrator can create the following fully-qualified, specific profiles to authorize or deny the use of DFSMShsm user commands.

New facility class profiles are added to be used for security for the RETAINDAYS keyword:

- STGADMIN.ARC.ENDUSER.HBACKDS
- STGADMIN.ARC.ENDUSER.HBACKDS.NEWNAME
- STGADMIN.ARC.ENDUSER.HBACKDS.RETAINDAYS
- STGADMIN.ARC.ENDUSER.HBACKDS.TARGET

You must have READ authority to issue the **HBACKDS RETAINDAYS** command. The default is that the **BACKDS RETAINDAYS** command is disallowed.

Coexistence support for RETAINDAYS keyword

Pre-z/OS V1R11 DFSMShsm functions that encounter a backup copy with a RETAINDAYS value, or a retained backup copy made on a z/OS V1R11 system, will have limited processing ability.

The functions affected include (H)BACKDS, (H)RECOVER, EXPIREBV, (H)BDELETE, DELVOL, FREEVOL, AUTOBACKUP, RECYCLE, and AUDIT.

Note: APAR OA26327 provides coexistence for systems at releases prior to z/OS V1R11.

7.2 ML1 enhancements

DFSMShsm ML1 enhancements provide the capability to back up and migrate data sets larger than 64 K to disk as large format sequential data sets. DASD volumes available to users for data allocation are called primary volumes. ML1 volumes are DFSMShsm-owned DASD volumes that contain data sets migrated from primary volumes. The data can be compressed. ML1 volumes are always DASD.

Large format data sets are physical sequential data sets that have the ability to grow beyond the previous size limit of 65535 tracks per volume. In previous releases, data sets larger than 64 K tracks in size can migrate and back up only to tape.

OVERFLOW and NOOVERFLOW volumes

Data sets larger than 58 K tracks will be allocated as LFS data sets regardless of whether they are on an OVERFLOW or NOOVERFLOW volume. DFSMShsm will enable ML1 OVERFLOW volumes to be selected for migration processing in addition to their current use for data set backup processing.

There are two types of ML1 volumes, OVERFLOW and NOOVERFLOW. They are mutually exclusive optional sub-parameters of the MIGRATION parameter that you use to specify how a level 1 volume is considered during selection for placement of a data set migrate or backup version.

OVERFLOW OVERFLOW specifies that the volume is considered only if:

- The data set being migrated or backed up is larger than a given size.
- An out-of-space failure has occurred when using the least-usage and most-free-space selections that do not include the overflow volumes.

NOOVERFLOW NOOVERFLOW specifies that the volume is considered with other level 1 volumes for migration data and backup versions of any size.

If you are adding a migration volume to DFSMShsm, the default is NOOVERFLOW. If you are changing the attributes of a volume and do not specify either subparameter, the OVERFLOW attribute is not changed.

Note: The OVERFLOW and NOOVERFLOW sub-parameters do not apply to the MIGRATIONLEVEL2 parameter. If you specify the OVERFLOW or NOOVERFLOW sub-parameter when it does not apply, DFSMShsm ignores it.

Migration or backup example

Figure 7-7 shows how three different-sized data sets are processed when migrating or backing up to DASD using OVERFLOW and NOOVERFLOW ML1 volumes as follows:

- The 500 track data set (solid line) when migrating is allocated on ML1 NOOVERFLOW volumes as a basic format sequential data set. This is how current HSM processing works prior to z/OS V1R11.
- The 40 K track data set (first dotted line) migrates to an OVERFLOW volume because it is larger than the specified DATASETSIZE parameter. The migration copy will be a basic format sequential data set, because its expected size is smaller than 58 K tracks.
- The 100 K tracks data set (bottom dotted line is allocated as a large format sequential data set on a volume in the ML1 OVERFLOW volume pool.

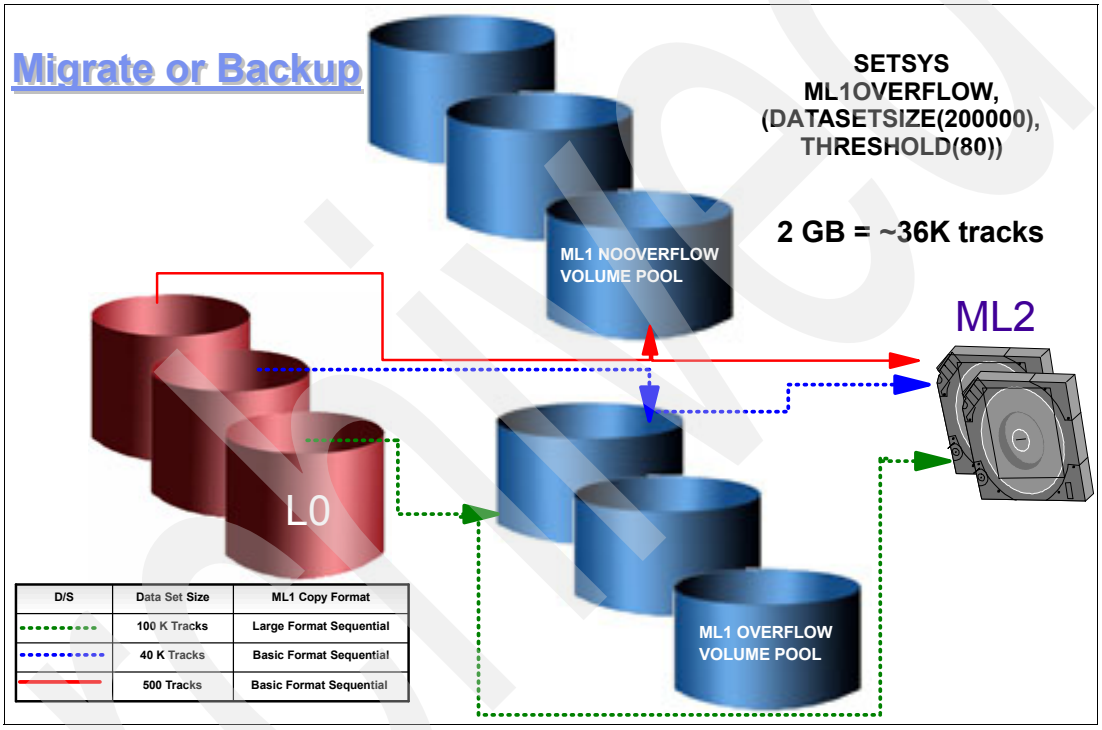


Figure 7-7 Data set allocation format and volume pool determination example

ML1 and ML2 volumes

DFSMShsm allocates migration copies and backup copies for large data sets (>58 K tracks) as large format physical sequential (LFS) data sets.

Migration occurs to either of two levels: migration level 1 or migration level 2.

- Migration level 1 (ML1) volumes are always DASD.
- Migration level 2 (ML2) volumes can be either DASD or tape. The computing center controls which volumes are to be used as migration volumes.

For data sets smaller than 58 K tracks, DFSMShsm allocates a basic sequential format data set for the backup copy. For data sets larger than 58 K tracks, DFSMShsm allocates a large format sequential data set for the backup copy.

Basic or large format sequential data sets will prefer OVERFLOW or NOOVERFLOW volumes based on the SETSYS ML1OVERFLOW(DATASETSIZE(dssize)) value. If there is not enough free space on the NOOVERFLOW or OVERFLOW volume for a particular backup

copy, DFSMShsm tries to create the backup on a OVERFLOW or NOOVERFLOW volume respectively. If the data set is too large to fit on a single ML1 volume (OVERFLOW or NOOVERFLOW), then the migration or backup fails.

Also, if the frequency of the backup is high, roll-off processing will cause even the backup copy to be replaced more quickly than normal processing.

Note: The maximum migration or backup copy size will now be limited to the size of the ML1 volume's free space.

7.2.1 Specify ML1 volumes to hold large data sets

Whether you are implementing space management or availability management, you need migration level 1 volumes. Migration processing requires migration level 1 volumes as targets for data set migration. Backup processing requires migration level 1 volumes to store incremental backup and dump VTOC copy data sets.

They may also be used as intermediate storage for data sets that are backed up by data set command backup.

ARCCMDxx parmlib member

The ARCCMDxx parmlib member contains the commands that define one or more DFSMShsm hosts parameters. When an instance of DFSMShsm is started, it reads its corresponding ARCCMDxx parmlib member to obtain environmental information about the way you have defined DFSMShsm.

Ensure that you include the **ADDVOL** command specifications for migration level 1 volumes in the ARCCMDxx parmlib member so that DFSMShsm recognizes the volumes at each startup. If the **ADDVOL** commands for migration level 1 volumes are not in the ARCCMDxx parmlib member, DFSMShsm does not recognize that they are available unless you issue an **ADDVOL** command at the terminal for each migration level 1 volume.

As shown in Figure 7-8, an ML1 volume becomes a repository for large data set backup versions and migration when there is insufficient space on other ML1 volumes.

```
ADDVOL MIG201 UNIT(3390) MIGRATION(ML1 OVERFLOW)
```

Figure 7-8 ADDVOL command to ML1 volume overflow

There is also a new command, shown here, that determines the size of data sets for which an ML1 OVERFLOW volume is preferred for migration or backup:

```
SETSYS ML1OVERFLOW(DATASETSIZE(dssize) THRESHOLD(threshold))
```

DSSIZE This parameter specifies the minimum size, in K bytes, of the data set for which an ML1 OVERFLOW volume is preferred for migration or backup. This includes invoking inline backup, the **HBACKDS** or **BACKDS** commands, or the ARCHBACK macro. The default is 2000000 K bytes.

THRESHOLD This parameter specifies the limit for the percentage of occupied space of the OVERFLOW ML1 volume pool. When the threshold is reached or exceeded for the entire pool, migration of data sets from level 1 volumes to level 2 volumes will start during secondary space management (SSM).

Note: The maximum migration or backup copy size will now be limited to the size of the ML1 volume's free space.

7.3 SMS critical data set separation by volume allocation

Data set separation allows you to designate groups of data sets in which all SMS-managed data sets within a group are kept separate. For prior releases, when allocating new data sets or extending existing data sets to new volumes, SMS volume selection frequently calls SRM to select the best volumes. Unfortunately, SRM can select the same set of volumes that currently have the lowest I/O delay. Poor performance or a single point of failure can occur when a set of function-related critical data sets are allocated onto the same volumes.

Now, with data set separation by volume starting in z/OS V1R11, you can specify that critical data sets be allocated onto other volumes, to prevent DASD hot spots and reduce I/O contention for them. The existing data set separation function is extended from the PCU level to the volume level, to separate critical SMS-managed data sets onto other extent pools and volumes from other data sets in the separation group.

A striping enhancement removes a number of restrictions to make SMS striping volume selection as close as possible to conventional volume selection.

7.3.1 Data set separation by volume

Starting in z/OS V1R11, you can specify that critical data sets be allocated onto different volumes, to prevent DASD hot spots and reduce I/O contention for them. The existing data set separation function is extended from the PCU level to the volume level, to separate critical SMS-managed data sets onto different extent pools and volumes from other data sets in the separation group.

The syntax of data set separation function was enhanced. The old syntax only supports PCU separation. The new syntax supports separation at both PCU and volume levels. Wild card characters * and % are supported at the third-level qualifier of the data set name, such as A.B.*. Old syntax for PCU separation will continue to function. Both old and new syntax can coexist in the same separation profile.

Data set separation profile

To use data set separation, you must create a data set separation profile and specify the name of the profile to the base configuration. During allocation, SMS attempts to separate the data sets listed in the profile.

A data set separation profile contains at least one data set separation group. Each data set separation group specifies whether separation is at the PCU or volume level and whether it is required or preferred. It also includes a list of data set names to be separated from each other during allocation.

Restriction: You cannot use data set separation when allocating non-SMS-managed data sets or during use of full-volume copy utilities such as PPRC.

Recommendations for using data set separation

Use data set separation only for a small set of mission-critical data for the following reasons:

- When separation is at the PCU level, an existing data set might be allocated to one or more PCUs from which the current allocation is to be separated. In this case, SMS either rejects all volumes on the PCUs from the SMS volume preference list or places them at a less-preferred position. Volume rejection might drastically reduce the number of eligible volumes or result in allocation failures if SMS rejects all volumes.
- The use of wildcard characters in the data set names that need to be separated from each other might drastically reduce the number of eligible volumes or result in allocation failure if SMS rejects all volumes.
- Data set separation can affect system performance. The following factors affect code path length when data set separation is requested:
  - The number of data set names listed in the data set separation profile. (SMS must scan the profile to determine whether data set separation processing is needed for the current allocation.)
  - The number of data set names listed within a data set group when data set separation is performed for that group.
  - The use of wildcard characters in the separation data set names.
  - The number of volumes that are eligible for selection.

Old syntax supports only PCU separation

The previous syntax is as follows:

```
{SEPARATIONGROUP | SEP}
{FAILLEVEL | FAIL}({PCU|NONE})
{DSNLIST | DSNS | DSN}(data set name[,data set name,...])
```

New syntax supports both PCU and volume separation

To separate data sets by volume, specify the groups of data sets that need to be kept separate from each other, and whether the separation is mandatory or preferred. This specification is done in the separation profile. The specified separation groups include a list of data set names to be separated from each other during allocation.

To specify data set separation by volume, use the new VOLUME value on the SEPARATIONGROUP keyword of the data set separation profile. You can specify the separation level to be by PCU as previously, or by volume. A new TYPE keyword specifies whether the separation is required or preferred. For example, the following syntax specifies that separation is required to be by volume:

```
{SEPARATIONGROUP | SEP}({PCU | VOL})
TYPE({REQ | PREF})
{DSNLIST | DSNS | DSN}(data set name[,data set name,...])
```

The new parameters are:

- **SEPARATIONGROUP(PCU)**

  This indicates that separation is on the PCU level.

- **SEPARATIONGROUP(VOLUME)**

  This indicates that separation is on the volume level. VOLUME may be abbreviated as VOL.

- **TYPE(REQUIRED)**

  This indicates that separation is required. SMS fails the allocation if the specified data set or data sets cannot be separated from other data sets on the specified level (PCU or volume). REQUIRED may be abbreviated as REQ or R.

- **TYPE(PREFERRED)**

  This indicates that separation is preferred. SMS allows the allocation if the specified data set or data sets cannot be separated from other data sets on the specified level. SMS allocates the data sets and issues an allocation message that indicates that separation was not honored for a successful allocation. PREFERRED may be abbreviated as PREF or P.

- **DSNLIST(data-set-name[,data-set-name,...])**

  This specifies the names of the data sets that are to be separated. The data set names must follow the naming convention described in *z/OS MVS JCL Reference*, SA22-7597. You can specify the same data set name in multiple data set separation groups. Wildcard characters are supported, beginning with the third qualifier. For example, you can specify BJONES.TEST.* but not BJONES.*.

  The wildcard characters are as follows:

  ***** This indicates that either a qualifier or one or more characters within a qualifier can occupy that position. An asterisk can precede or follow a set of characters.

  ****** This indicates that zero or more qualifiers can occupy that position. A double asterisk cannot precede or follow any characters; it must be preceded or followed by either a period or a blank.

  % This indicates that exactly one alphanumeric or national character can occupy that position. You can specify up to eight % characters in each qualifier.

Important: The following earlier form of the syntax for SEPARATIONGROUP is tolerated by z/OS V1R11. It supports separation at the PCU level only.

```
SEPARATIONGROUP|SEP
FAILLEVEL|FAIL ({PCU|NONE})

DSNLIST|DSNS|DSN (data-set-name[,data-set-name,...])
```

7.3.2 SMS volume selection

SMS volume selection will attempt to allocate data sets that are listed in the volume separation groups onto other extent pools and volumes. During volume selection, SMS prefers candidate volumes as follows:

- Candidate volumes that are in different extent pools than volumes on the separation volume list are the most preferred.
- Candidate volumes that are not on the separation volume list but are in the same extent pools as the volumes on the separation volume list are the next preferred.
- When separation by volume is required, SMS must allocate the data set onto the volumes that are not used by any of the data sets in the same separation group.
- When separation by volume is preferred, the candidate volumes that are already occupied by other data sets in the separation group are ranked lower and are less preferred.

Data set separation by volume function is intended for a small set of critical SMS-managed data sets. Data set separation by volume function is requested when volume separation group is defined in the data set separation profile.

Migration and coexistence considerations

Coexistence APAR OA25344 must be installed on z/OS V1R19 and V1R10 when the new syntax is used, as follows:

- Ignores volume separation syntax.
- Supports new syntax for PCU separation.
  - Ignores PCU separation groups that use wild card characters.
- Without OA25344, data set separation function is deactivated if new syntax is detected in V1R9 and V1R10.
- Data set separation function is deactivated if new syntax is detected in V1R8 and earlier.

Publications changed

See “Using Data Set Separation” in *z/OS DFSMS Storage Administration Reference (for DFSMSdfp, DFSMSdss, DFSMShsm)*, SC26-7402, for more information about this topic.

7.4 SMS striping volume selection enhancements

Striping volume selection is very similar to conventional volume selection. Volumes that are eligible for selection are classified as primary and secondary and assigned a volume preference weight, based on preference attributes.

In prior releases SMS striping volume selection had several restrictions, as follows:

- Volumes that are above the high allocation threshold are rejected.
- Enabled and quiesced volumes are treated equally.
- Normal and overflow storage groups are treated equally.
- Few volume preference attributes are supported.

DFSMSdfp in z/OS V1R11

DFSMSdfp in z/OS V1R11 removes the restrictions and selects volumes by volume preference weight, similar to non-striping volume selection allowing striping volume selection, to support all current and future volume preference attributes specified in SMS constructs as listed:

- Allow volumes that are above the high allocation threshold to be eligible for selection as secondary volumes. This prevents rejection due to the high allocation threshold being low.
- Prefer enabled volumes over quiesced volumes.
- Prefer normal storage groups over overflow storage groups.
- Support the data set separation function.
- Support the multi-tiered storage group function that honors the storage group sequence derived by ACS.
- Support availability, accessibility, PAV and other volume preference attributes that are used in non-striping volume selection.

7.4.1 Selection criteria with z/OS V1R11

SMS calculates the average preference weight of each storage group using the preference weights of the volumes that will be selected if the storage group is selected for allocation. Then, SMS selects the storage group that contains at least as many primary volumes as the stripe count and has the highest average weight.

If there are no storage groups that meet these criteria, the storage group with the largest number of primary volumes is selected. If multiple storage groups have the largest number of primary volumes, then the one with the highest average weight is selected. If there are still multiple storage groups that meet the selection criteria, then SMS selects one at random.

After selecting a storage group, SMS selects volumes by their preference weight. Primary volumes are preferred over secondary volumes because they have a higher preference weight. Secondary volumes are selected when there are an insufficient number of primary volumes. If there are multiple volumes with the same preference weight, SMS selects one of the volumes at random, as follows:

- Rank volumes by preference weight from each individual controller.
- Select the most preferred storage group that meets (or closely meets) the target stripe count.
- Select the most preferred volume from individual controllers to meet the stripe count (try to spread stripes across controllers).
- Automatically activate the fast volume selection function to avoid overutilizing system resources.
  - Fast volume selection rejects the remaining candidate volumes that do not have sufficient free space when 100 most preferred volumes have been rejected by DADSM for insufficient space.

Note: For non-guaranteed space, the volume must be able to satisfy the primary space that is requested divided by the number of stripes. For example, if primary space is 15 MB and the number of stripes is three, the volume must be able to satisfy an allocation of 5 MB.

7.5 IDCAMS delete mask

In prior releases the **DELETE** command can delete a generic entry name, but only one qualifier can be replaced by the wildcard asterisk (*). The command **DELETE A.B.*** allows the deletion of three entry levels only. The deletion does not perform on data set name A.B.C.D

z/OS V1R11 improvements

In DFSMSdfp in z/OS V1R11, the IDCAMS **DELETE** command is enhanced to include a new function called **DELETE MASK**. The **DELETE MASK** command allows only one data set entry name to be specified. If multiple entry names are specified, the **DELETE** request will fail with error messages. If more than 100 data set names are filtered from the wildcard notation, AMS only deletes the first 100 data sets identified by the filtering process.

If you do not specify the **MASK** keyword on the **DELETE** command, or if you do not explicitly specify **NOMASK**, the previous wildcard rules remain in effect (one asterisk (*) can replace one qualifier in a data set name).

Consider the following changes with z/OS V1R11:

- It allows users to specify the data set name selection criteria desired with a mask entry name and a keyword MASK.
- A mask entry name (also called a filter key) can have two consecutive asterisks (**) or one or more percentage signs (%).
- The two consecutive asterisks(**) represent zero (0) or more characters and it is not limited to the number of levels. For example, A.B.** means all data set names with two or more levels with A and B as their first and second qualifiers, respectively.
- The percentage sign (%) replaces any character in that same relative position. For example, ABCDE matches the mask entry A%%DE, but not A%DE.
- The MASK keyword turns the new feature on; for example:

  DELETE A.B.** MASK

  DELETE A.BC.M%%K MASK
- The NOMASK keyword turns the new function off. The default is NOMASK.

Using the MASK keyword

Using the MASK keyword, the you can:

- Optimize the **DELETE** command
- Delete up to 100 data sets without specifying multiple entry names
- Remove the stipulation on the deleted data set name selection criteria

7.5.1 Implementation with z/OS V1R11

With z/OS V1R11, the support is invoked by the IDCAMS **DELETE** command and is called by a batch job or a TSO command interface. Note that **Delete Mask** *cannot* delete the following types:

- TRUENAME (TRUENAME)
- Non-VSAM volume record (NVR)
- VSAM volume record (VVR)
- PDS or PDSE member
- Library entry (LIBRARYENTRY)
- Tape volume entry (VOLUMEENTRY)

New or changed messages:

IDC2899I MASK PARAMETER NOT ALLOWED FOR...

IDC2900I MASK PARAMETER NOT ALLOWED FOR...

IDC2901I MASKING ENTRY NAME REQUIRES A KEYWORD...

IDC2092I NO ENTRIES FOUND FOR MASK xxxxxxxxx.

7.6 Catalog health check enhancements

No supported release of z/OS honors the IMBED and REPLICATE attributes for catalogs. These attributes can waste DASD space and often degrade performance, and several unplanned outages have occurred, so avoid using the IMBED and REPLICATE attributes. These attributes were initially intended as a performance improvement on older disk technology, but now they are obsolete and have been replaced by newer, cached devices.

z/OS V1R11 support

Now if a new catalog is defined with these attributes, they are simply ignored on the define and the catalog is defined without them.

In z/OS V1R11, the catalog health check now monitors the system and notifies the user of any user catalogs and any master catalogs with IMBED or REPLICATE attributes.

New console messages are displayed depending on the check's findings.

- If no catalogs are detected by the health check, then an informational message is displayed on the console, as follows

  ```
  IGGHC103I CHECK(IBMCATALOG,CATALOG_IMBED_REPLICATE) ran successfully and
  found no exceptions.
  ```

- If one or more catalogs are detected by the check with IMBED/REPLICATE attributes, then the following exception message is displayed on the console:

  ```
  IGGHC104E The health check has detected one or more catalogs defined with
  the IMBED and/or REPLICATE attributes.
  ```

A report is then generated in SDSF, which you can view to identify the affected catalogs. The report consists of four columns, as shown in Figure 7-9.

- Column 1 identifies the name of the affected catalog.
- Column 2 identifies the attribute (IMBED or REPLICATE).
- Column 3 identifies the catalog component (DATA or INDEX) with the attribute.
- Column 4 identifies whether the catalog was available for processing.

```
 Display  Filter  View  Print  Options  Help

-------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY CATALOG_IMBED_REPLICATE       LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> HALF
******************************** TOP OF DATA
********************************
CHECK(IBMCATALOG,CATALOG_IMBED_REPLICATE)
START TIME: 08/11/2009 19:32:53.924561
CHECK DATE: 20080619  CHECK SEVERITY: LOW

IGGHC106I
Following catalog(s) were inspected by the health check

-------------------------------------------------------------------------
 CATALOG NAME                                ATTRIBUTE  COMP AVAILABILITY
-------------------------------------------------------------------------
CATALOG.DB2ICF2.VTOTCAT                       IMBED     DATA  YES
CATALOG.DB2ICF2.VTOTCAT                       IMBED     INDEX YES
CATALOG.DB2ICF2.VTOTCAT                       REPLICATE INDEX YES
CATALOG.HCD.USERCAT                           IMBED     DATA  YES
CATALOG.HCD.USERCAT                           IMBED     INDEX YES
CATALOG.SHRICF1.VIODFPK                       IMBED     DATA  YES
CATALOG.SHRICF1.VIODFPK                       IMBED     INDEX YES
CATALOG.TOTICFM.VTOTCAT                       IMBED     DATA  YES
CATALOG.TOTICFM.VTOTCAT                       IMBED     INDEX YES
CATALOG.TOTICFM.VTOTCAT                       REPLICATE INDEX YES
MCAT.OS3RSA.VOS3CAT                           IMBED     DATA  YES
MCAT.OS3RSA.VOS3CAT                           IMBED     INDEX YES
MCAT.OS3RSA.VOS3CAT                           REPLICATE INDEX YES
MCAT.OS3R3V01.VOS3CAT                         IMBED     DATA  YES
```

Figure 7-9 Catalog health check report

Informational message

If the check encounters an error during catalog search interface processing, an informational message is displayed and the check is disabled until the error is resolved.

IGGHC107I CSI has encountered an error in its processing while attempting to return data from the Catalog Address Space. Check message HZS1002E for diagnostic information. The check is disabled.

If the check found no user catalog connector in the master catalog, then the following informational message is displayed:

IGGHC108I No user catalog connector entries were found in the master catalog

At the end, in SDSF a summary report is printed to display the findings of the check, as follows:

IGGHC109I IMBED/REPLICATE Health Check Summary Report.

7.7 SMS Open, Close, and EOV enhancements

If an exit gives an unsupported return code, the system writes a message that displays the name of the exit and the invalid return code. The system then abends the job.

Open/Close/EOV provides the following enhancements in z/OS V1R11:

- Displaying the fact that a data set has an expiration date set to never expire, in message IEC507D
- The system no longer reprocesses the same volume when duplicate volume serial numbers are detected during multivolume tape processing (automatic recovery for message IEC708I).
- Providing improved mapping for SMF type 14 and 15 records, with the option to provide DSECTs for the individual segments of the records.
- Providing an installation-wide abend option for multi-volume tape conditions that generate messages IEC709I, IEC710I, IEC711I, and IEC712I. New parameter bits and a new return code are added for the label anomaly exit, to allow abending of jobs for these conditions.

7.7.1 Label anomaly installation exit

On other platforms, tape labels are written by the application program or a utility program and not the operating system. They often do not have the concept of a multivolume file, but rather a series of single volume files. Several problems occur when processing these multivolumes' standard tapes on z/OS.

- The system is too lax in allowing volumes to be missing or out of the order in which they were created.
- There is no ABEND option when OPEN and EOV detect missing volumes or volumes being read out of order.
- Many applications knowingly create these conditions.

z/OS V1R11 enhancements

z/OS V1R11 provides for multiolume tape conditions when reading an unexpected volume sequence number, or when a missing final volume or missing first volume (reading forward or backward) is encountered during multivolume tape processing.

If you decide to use the new installation-wide abend option, check for the new Label Anomaly indicator flags for these conditions and then use return code C as documented in the label anomaly exit information in *z/OS DFSMS Installation Exits*, SC26-7396. With return code C and the corresponding anomaly bit on, this exit issues a message and abends the job.

Restricted: The facilities discussed in this publication are available on certain System z Processor Complexes. The information published herein should not be construed as implying any intention by IBM to provide these facilities on models other than those described. This facility is available on System z10 EC GA2 and z10 BC processors and is supported by a new z/OS component (Instrumentation), Hardware Instrumentation Services (HIS).

```
IEC709I dddd,volser,jn,sn,ddn-nu EXPECTED VOLSEQ: nnnn FOUND: nnnn

Explanation: Volume is out of order)

IEC710I dddd,volser,jn,sn,ddn-nu another volume expected

Explanation: Last volume is missing)

IEC711I dddd,volser,jn,sn,ddn-nu RDBACK – NOT LAST VOLUME OF DATA SET

Explanation: Read backward missing last volume

IEC712I dddd,volser,jn,sn,ddn-nu READ - NOT FIRST VOLUME OF DATA SET

Explanation: Read forward missing first volume)
```

Return codes

For these anomalies, any return code other than 12 is ignored. The related message continues to be issued and no ABEND will be detected. A new return code 12 is recognized only for these anomalies, and it results in either of the following:

ABEND 413-08 and IEC145I for OPEN, or

ABEND 637-2C and IEC026I for EOV

Note: When a duplicate volume serial is detected in the volume list, EOV processing issues message IEC708I,and then processes the next volume serial after the second duplicate.

Using the new tape label anomaly option:

- You can detect missing or out of sequence volume processing of multivolume standard label tape data sets.
- You can determine whether to allow processing to continue or to abnormally terminate the job.

7.7.2 DASD data set expiration control

Originally OS/360 did not have the concept of never-expire because the operating system was not expected to exist after 1999. Years were expressed as two digits.

Before a data set's expiration, OPEN for writing causes the system operator to be asked permission to write and system automation cannot distinguish between two causes that can generate the message:

```
IEC507D E dev,ser,jjj,sss,dsname REPLY 'U'-USE OR 'M'-UNLOAD
   Cause 1: Data set's expiration date has not been reached.
   Cause 2: Data set has a never-expire date of 99365 or 99366.
```

z/OS V1R11 provides an installation automation option to enforce only never-expiring dates (that is, 99365 or 99366) by automating the reply to:

```
IEC507D E dev,ser,jjj,sss,dsname [NEVEREXPIRE] REPLY 'U'-USE OR 'M'-UNLOAD
```

7.8 VSAM data trap

Sometimes VSAM detects a structure error in an index that was caused by an earlier error in the data component. Without a data trap, no documentation or dumps are available to help you diagnose the problem.

VSAM has an *index trap* that checks each index record before writing it. The index trap is on, by default.

Now, the VSAM data trap enhances serviceability. When data is corrupted, the data trap helps clients to gather useful information to give to IBM Service to analyze the failure.

The VSAM data trap diagnostic tool:

- Prevents corrupted KSDS (key-sequenced data set) records from being written to DASD
- Maintains the integrity of the data set
- Captures information and a dump for broken data records for further analysis

The VSAM data trap is off, by default.

Activate the VSAM data trap

To activate the trap, issue the following command:

```
V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),ON
```

To deactivate the trap, issue the following command:

```
V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),OFF
```

To display the status of VSAM data trap, issue the following command:

```
D SMS,MONDS(IGWVSAM.BASE.VSAM.DEBUG.FEATURES)
```

Using the VSAM data trap, you can collect dumps and failure information for IBM service to further determine the problem when knowing data was or might be corrupted. It affects writing to all KSDSs in the system.

However, there can be a high performance impact when the trap is turned on, so avoid having it turned on during normal processing. Instead, turn it on only when collecting diagnostic information for further problem determination.

7.9 VSAM system-managed buffering enhancement

Record Access Bias specifies whether to let VSAM determine how many and which type of buffers to use when accessing VSAM extended format data sets by batch processing. This is known as system-managed buffering. It is available to VSAM data sets in any record organizations that are allocated in extended format.

The performance of a KSDS data set is directly related to buffers that are allocated at open time. A small data set usually results in a few buffers being allocated, but these are enough for its processing. However, if a data set increases its size significantly during processing, then it might not have enough buffers to reach the same performance level as when it opened. The application might have to close and reopen the data set to return the same performance level.

Now, z/OS V1R11 VSAM can possibly calculate a more optimized value for the number of index buffers and data buffers as specified by the SMBVSP parameter, to avoid the need for an application to close and reopen the data set.

SMBVSP parameter

This option specifies the amount of virtual storage to obtain for buffers when a data set is opened. You can specify the virtual buffer size in kilobytes (from 1K to 2048000K) or in megabytes (from 1M to 2048M). The SMBVSP parameter can be used to restrict the size of the pool that is built for the data component.

You can also use SMBVSP to increase the storage amount used for both data buffer space and index buffer space. VSAM chooses a maximum number of index buffers, either based on 20% of the SMBVSP value that you specify, or the current data set size. You can use SMBVSP to improve performance when too few index buffers were allocated for a data set that grows from small to large, without being closed and reopened over time.

8

Hardware Instrumentation Services

The Hardware Instrumentation Services (HIS) function provides a framework for collecting and recording hardware event data for IBM System z10 machines. APARs OA25750 and OA25773 are required to enable full function of this support.

This function collects hardware event data for processors in SMF records type 113, subtype 2, as well as UNIX System Services output files. Note that you can only use HIS for IBM System z10 or later machines.

8.1 CPU Measurement Facility overview

Having discussed several software techniques to improve performance with HiperDispatch in previous chapters, here we focus on measuring performance and analyzing possible weaknesses. A significant improvement achieved with the System z10 server is hardware support for performance analysis. This support makes understanding performance problems easier and much more precise than the software-based methods used before. The new hardware instrumentation does not disturb the system being observed, although it requires support from z/OS.

These sections provide a brief overview of the CPU Measurement Facility architecture. Subsequent sections explain how to invoke this function in z/OS.

Note: Hardware instrumentation services (HIS) is a function that collects hardware event data for processors in SMF records type 113, subtype 2, as well as a UNIX System Services output files. You can only use HIS for IBM System z10 or later machines.

You must start the HIS address space on each system where you want to collect data. Then you must configure and activate HIS data collection (hardware event counters and sampling facilities) for each system by issuing the `F hisproc` command.

8.1.1 Hardware-based performance analysis

Hardware instrumentation offers many advantages over the software-based performance tools used in previous systems. The CPU Measurement Facility, available in z10 systems, provides support built into the processor for performance analysis. Exploiting this mechanism allows you to observe the performance behavior of an operating system running client applications, and it does so with little impact to the system being observed.

The CPU Measurement Facility is designed to increase the ability to run hardware instrumentation in the field and to support application performance tuning.

In the past, System z mainframe hardware instrumentation has been used for the following tasks:

- Debugging hardware performance and enabling performance tuning by the sophisticated compilers of operating systems and internal subsystems
- Allocating (or deallocating) a sufficient hardware system area for storing measurement data

However, allocation or deallocation of hardware system area requires a machine power-on. Also, hardware instrumentation runs on CPs at the basic machine-level only; thus, running it requires the entire machine. Because of these constraints, hardware instrumentation has been limited mainly to laboratory use. The CPU Measurement Facility resolves these issues by enabling the control program to allocate real storage in which the machine can store measurement data and to virtualize the hardware instrumentation so that it can run in multiple LPARs concurrently.

The format and meaning of measurement data used to facilitate application performance tuning are defined in the architecture. The remainder of the CPU Measurement Facility, used to establish controls and extract measurement data, is basically the same, independent of the intended use. The discussion from this point focuses on the use of measurement data to tune application performance.

Installation considerations

This architecture defines the CPU Measurement Facility, which is available in the z/Architecture architectural mode. When the facility is installed on a CPU, the same facility is installed on all CPUs in the configuration.

Measurement data provided by the CPU Measurement Counter Facility or the CPU Measurement Sampling Facility is intended for statistical estimation of performance characteristics. Measurement data is only substantially accurate, and may not be repeatable. For example, it is unpredictable if the instruction count counts an instruction that caused an exception or counts the target of EXECUTE. Also, certain system internal activities may be included in measurement data.

The CPU Measurement Facility consists of two parts:

- A CPU Measurement Counter Facility

  This facility provides a means to measure activities in the CPU and in various shared peripheral processors.

- A CPU Measurement Sampling Facility

  This facility provides a means to take a snapshot of the CPU at a specified sampling interval and requires the SET PROGRAM PARAMETER facility as a prerequisite. Each part consists of a number of instructions to set and extract controls or measurement data, and consists of several events of an external-interruption subclass.

Note: Disable the CPU Measurement Counter Facility and the CPU Measurement Sampling Facility when they are not being used.

8.2 CPU Measurement Counter Facility

The CPU Measurement Counter Facility on each CPU consists of a local counter-set-state control register, a number of counters, several external-interruption events, a measurement-counter-extraction-authorization control, and a number of instructions.

The following list summarizes the instructions.

```
EXTRACT COPROCESSOR-GROUP ADDRESS
EXTRACT CPU COUNTER
EXTRACT PERIPHERAL COUNTER
QUERY COUNTER INFORMATION
SET CPU COUNTER
SET CPU-COUNTER-SET CONTROLS
SET PERIPHERAL COUNTER
SET PERIPHERAL-COUNTERSET CONTROLS
```

Measurement counters

The measurement-counter-extraction-authorization control, which is bit 15 of control register 0, specifies whether selected CPU counter contents can be extracted in the problem state. The bit is initialized to zero (0).

The facility provides two kinds of counters: local counters and global counters.

- Local counters are local to the CPU. These counters are used to count logical CPU activities and are grouped into several CPU counter sets.

- Global counters are used to count activities of peripheral processors, which are shared among certain CPUs, and are grouped into different peripheral counter sets.

All counters are 64 bits. A carry-out of bit position 0 of any counter is ignored.

When any peripheral counter set is provided, the facility also includes a global counter-set-state control register.

The number and the type of installed counters are model-dependent. Two 16-bit counter version numbers, called counter first version number and counter second version number, are provided by QUERY COUNTER INFORMATION. Each version number identifies installed counters in certain counter sets.

When there is a change to the meaning of a counter or the number of installed counters in any counter set specified by a counter version number, then another value is indicated by that counter version number.

Both counter version numbers start at one and increase consecutively.

Cryptographic coprocessor support

IBM z/Architecture includes a message-security assist that supports cryptographic operations. The crypto activity counter set collects information about activities caused by executing cryptographic operations. Because the z10 machine is the first in which two CPs share a cryptographic coprocessor, this counter set provides information about characteristics of the cryptographic workload and cryptographic interference between the two sharing CPs.

A cryptographic coprocessor group consists of a cipher coprocessor for the Data Encryption Standard and the Advanced Encryption Standard and a hash coprocessor for the Secure Hash Algorithm. Both coprocessors can perform cryptographic operations simultaneously. However, when two sharing CPs attempt to use the same coprocessor, one will be blocked until a predetermined time slice has passed.

Additionally, a counter set is provided for each cryptographic coprocessor group to count cryptographic activities contributed by both of the sharing CPs. This counter set provides information about the utilization of each cryptographic coprocessor.

Sampling provides a means to take a snapshot of the CP at a program-specified sampling interval. At the end of each sampling interval, the machine stores sample data into measurement blocks allocated by the control program. Each sample data includes an instruction address, the primary address space number, and state information about the CP. This allows tooling programs to map instruction addresses into modules or tasks and facilitates the determination of hotspots. A control program can set up alerts before the measurement blocks are filled up so that it can take appropriate actions to save the sampling data and to free up space in the measurement blocks.

8.2.1 Counter sets

When both counter version numbers are one, the facility provides four CPU counter sets on each CPU. They are basic counter set, problem-state counter set, crypto-activity counter set, and extended counter set. Counters in the CPU counter sets count CPU activities on the logical CPU basis. The facility also provides one kind of peripheral counter sets, called coprocessor-group counter sets. One coprocessor group counter set is provided for each coprocessor group. Counters in a coprocessor-group counter set count activities of each coprocessor in the group at the basic machine level.

At the basic machine level, a coprocessor group consists of a number of coprocessors and can be attached to one or more CPUs. When authorized, all CPUs in the model have access to all coprocessor-group counter sets and the global counter-set-state control register.

When the counter first version number is one, it identifies installed counters in the basic counter set and problem-state counter set; when the counter second version number is one, it identifies installed counters in the crypto-activity counter set, the extended counter set, and the coprocessor-group counter sets.

The CPU Measurement Sampling Facility on each CPU consists of two sampling functions, several sampling control registers, several external-interruption events, and two instructions. The instructions are summarized here:

- QUERY SAMPLING INFORMATION
- SET SAMPLING CONTROLS

The facility provides a means to take a snapshot of the logical CPU at a specified sampling interval, which is a processing time interval as seen by the CPU. Each snapshot produces a set of sample data, which includes the instruction address of an instruction being executed and state information about the CPU. The CPU measurement sampling facility requires the SET PROGRAM PARAMETER facility as a prerequisite.

The facility includes two sampling functions: basic sampling and diagnostic sampling. They both use the same sampling control registers, the same sampling buffer structure, the same external interruption events, and the same instructions. The main difference between these two functions is the sample data. The sample data size and format for each sampling function are model-dependent, and are determined by a 16-bit data entry format code that is stored in each sample data.

The data entry format code starts with 0001 (hex) for the basic-sampling data entry; the code starts with 8001 (hex) for the diagnostic sampling data entry. Both data entry format codes continue consecutively. When the size, the meaning, or the format of a sample data is different from the previous model, then the appropriate data entry format code is incremented.

The sample data provided by the basic sampling function is not included in the sample data provided by the diagnostic sampling function. To get meaningful diagnostic sampling data, both sampling functions must be activated. The state of each sampling function can be individually set by executing SET SAMPLING CONTROLS. Both sampling functions are disabled by initial CPU reset, clear reset, or Power-on Reset.

8.3 Hardware Instrumentation Services address space

The Hardware Instrumentation Services (HIS) address space must be started on each system where you want to collect data. Then you must configure and activate HIS data collection (hardware event counters and sampling facilities) for each system by issuing the **f hisproc** command.

Important: Assign a sufficiently high dispatch priority to the instrumentation started task hisproc, so that the task can write sampling data to the SMP output files in a timely manner.

8.3.1 Setting up hardware event data collection

Hardware Instrumentation Services (HIS) is a function that collects hardware event data for processors at the System z10 Enterprise Class level or later.

During a run of hardware data collection, the system writes the data to a UNIX System Services output file as well as to SMF record type 113, subtype 2. The system writes the raw data to SMF record type 113 at 15-minute intervals during the data collection run, and writes the delta data to the UNIX System Services output file at the end of the run. The delta data is the data from between when the instrumentation run was initiated and the end of the run, showing the delta incremental values of the instrumentation run on the specified processor.

8.3.2 Security specifications

Before you start the HIS data collection, you may first need to authorize to the sampling facilities and counter set types you want to use through the support element (SE) console. For information about how to set up the authorization of the sampling facilities and counter sets, see *Support Element Operations Guide for System z10 machine* on the Resource Link™ home page at:

`http://www.ibm.com/servers/resourcelink`

Setting up hardware event data collection

Follow these steps to set up hardware event data collection:

1. Define a user ID for the HIS started task *hisproc*. Define the user ID with an OMVS segment that specifies:
   - Any UID.
   - A default `HOME` directory. For example, you might define the user ID as follows:
     - ADDUSER hisproc OMVS(UID(25) HOME('/user'))
2. Create a user `HOME` directory (in a local filesystem) by issuing the following **`mkdir`** command under OMVS:

   ```
   mkdir /hisuser
   ```

   Then assign read/write/exec authority to the `/hisuser` directory with the following **`chmod`** command:

   ```
   chmod 007 /user
   ```

3. Enable SMF record type 113 using either the **`SET SMF`** or **`SETSMF`** commands. For example, you might enable SMF record type 113 as follows:
   - Issue **`SET SMF=xx`** to select the SMFPRMxx parmlib member you want to update with information for SMF record type 113.
   - Reply to the message issued in response to the **`SET SMF=xx`** command to change any SMFPRMxx parameters. For example, you might reply with the following information:

     ```
     nn,sys(type(113)),intval(01),maxdorm(0100)
     ```

     Reply `nn,U` to continue.

     This reply will prompt the system to collect type 113 records, give you 1-minute collection intervals, and the data will only stay in the buffers for 1 minute before being written to the MANA or MANB data set. You can change other SMFPRMxx parameters on the **`SET SMF`** command response or the **`SETSMF`** command.

4. Start the HIS started task with the following command:

   ```
   START hisproc
   ```

   This command does the following:

   - It starts the Hardware Instrumentation Services (HIS) address space for the system.
   - It creates a new instrumentation started task, *hisproc*, for the system.

   See “Start, configure, and stop hardware event data collection” on page 199, for complete information about the **START hisproc** command.

5. Configure and start a run of data collection on a system with the following command:

   F hisproc,BEGIN

Note: You must explicitly start each run of hardware data collection. You cannot set up data collection to start running automatically.

8.3.3 Collecting data with HIS

You can specify in advance the duration of a run of data collection you want by using the DURATION parameter with the **F hisproc,BEGIN** command. If you configure a new processor online in a system after you have already issued the **F hisproc,BEGIN** command to start a data collection run for that system, then HIS might not collect data for that processor. To ensure that data is collected for all the processors on a system, bring the processors online before beginning a hardware data event collection run. The system does not collect data on a processor that is configured offline.

See “Start, configure, and stop hardware event data collection” on page 199, for complete information about the **F hisproc** command.

Stopping data collection

To explicitly stop the run of data collection on a system, use the following command:

```
F hisproc,END
```

As an alternative, you can use the **DURATION** parameter on the **F hisproc,BEGIN** command to specify when you want a data collection run to end. When the data collection run ends, the system writes all the collected data to the UNIX System Services data set at the path specified and to the SMF data set that was set up by the installation.

You can also use the **p hisproc** command to stop a run of data collection. Note that if you use the **STOP** command, you must restart the address space again with the **START** command before starting the next run of data collection.

See “Start, configure, and stop hardware event data collection” on page 199, for complete information about the **F hisproc** command.

Start, configure, and stop hardware event data collection

Use the **F hisproc** command to manage hardware event data collection for systems at the System z10 Enterprise Class level or higher. Use **F hisproc,BEGIN** to configure and start a run of data collection, and use **F hisproc,END** command to stop the run. You must explicitly start each run of hardware data collection; you cannot set up data collection to run automatically.

During a run of hardware data collection, the system writes the data to a UNIX System Services output file as well as to SMF record type 113, subtype 2. The system writes the raw data to SMF record type 113 at 15-minute intervals during the data collection run, and writes the delta data to the UNIX System Services output file at the end of the run. The delta data is the data from between when the instrumentation run was initiated and the end of the run, showing the delta incremental values of the instrumentation run on the specified processor.

Before you issue the **F hisproc,BEGIN** command to configure and start hardware event gathering on a system, you must perform several setup tasks; see 8.3.1, “Setting up hardware event data collection” on page 198.

See 8.4.3, “Accessing the output from a hardware event data collection run” on page 209 for information about the various files that HIS generates, depending on the **F hisproc,BEGIN** command parameters you specify.

If you configure a new processor online in a system after you have already issued the **F hisproc,BEGIN** command to start a data collection run for that system, HIS might not collect data for that processor. To ensure that data is collected for all the processors on a system, bring the processors online before beginning a hardware data event collection run. The system does not collect data on a processor that is configured offline.

8.4 HIS command file rules

Use the following syntax rules for the command file:

- The parameters specified can reside on multiple lines of the command file.
- You must separate keywords with commas.
- The system treats a blank character between any two non-blank characters on the same line (unless within quotes), or a blank line (or EOF) as the end of the command.
- You can use columns 1-72 of each line.
- A quoted string (a title or path name, for example) cannot span more than one line.
- Parameters and values in the command file must be in uppercase, unless the parameter value is a quoted string.

HIS command file

The command file referenced contains parameters for data collection runs, set up the same way they appear in the **MODIFY** command. The same rules and formatting apply to the command file that you use in the console command. The command file gives you an alternative to specifying data collection options in a **F hisproc** command, which can be useful if you have difficulty fitting all the desired parameters on the command.

The HIS command file must have a RECFM format of LRECL=80 fixed length record. The same syntax rules apply to the parameters in the command file as to an **F hisproc** command. The system flags duplicate parameters or mutually exclusive parameters entered in the **F hisproc** command and the command file as errors and you will receive an error message.

Figure 8-1 on page 201 shows an example of valid command file contents.

```
TITLE='Test run 123',
PATH='/user/john',CTRSET=(BASIC,PROB),
SAMPTYPE=(BASIC),
SAMPFREQ=850000,DATALOSS=IGNORE,
MAPASID=(7,0E,E1D),
MAPJOB=(PROG*,DB*,GRS,JE??)
```

Figure 8-1 Valid command file contents example

HIS start procedure

To use a command file for **F hisproc**, you must specify the ddname in the HIS started cataloged startup procedure, *hisproc*, such as CMDFILE1 or CMDFILE2 in Figure 8-2.

```
//HIS     PROC
//HIS     EXEC PGM=HISINIT,REGION=0K,TIME=NOLIMIT
//*
//* You can specify an MVS command file to contain some or all of
//* the settings for the instrumentation run. The command file
//* must have fixed-length LRECL=80 records.
//*
//* If this option is desired,
//*  1. Replace 'DUMMY' below with the name of the MVS command
//*     file and its DISPOSITION.
//*  2. Specify the DDNAME keyword on the 'MODIFY HIS' command.
//*     For example:
//*       "MODIFY HIS,BEGIN,DDNAME=CMDFILE1"
//*
//CMDFILE1 DD DUMMY
//CMDFILE2 DD DUMMY
//SYSPRINT DD SYSOUT=*
//*
//*******************************************************************/
//*                                                                 */
//*01* PROCEDURE NAME : HIS                                        */
//*                                                                 */
//*01* FUNCTION:                                                   */
//*                                                                 */
//*       The HIS procedure is used to start the Hardware           */
//*       instrumentation Services (HIS) function.                  */
//*       It invokes the module HISINIT.                            */
//*                                                                 */
```

Figure 8-2 Example of a HIS procedure

8.4.1 HIS command structure

Figure 8-3 on page 202 shows the command parameters.

```
F hisproc,{BEGIN | B} | {END | E}
[,{TITLE | TT} ='textdata']
[,PATH='pathname']
[,{DURATION | DUR}=duration_value]
[,{CTRSET | CTR } = {ALL | (ctr1[,ctr2[,ctr3]])}]
[,{DDNAME | DD}=ddname
[,{CTRONLY | MAPONLY}]
[,{MAPVERBOSE | MAPV}]
[,{BUFCNT | BUF}=bufcnt]
[,{SAMPFREQ | SF}=freq]
[,{SAMPTYPE | ST}={ALL | (samtype1[,samtype2])}]
[,{DATALOSS | DL}={IGNORE | STOP}]
[,{MAPASID | MAS}={ALL | (asid1,asid2,...asidn)}]
[,{MAPJOB | MJOB}=(job1,job2,...jobn)]
```

Figure 8-3 Modify command syntax

F hisproc command parameters

The **F hisproc** command parameters are:

hisproc This is the name of the Hardware Instrumentation Services (HIS) cataloged startup procedure.

BEGIN|B - END|E You must specify either BEGIN or END on the **F hisproc** command to begin or end a run of hardware event data collection for a system:

BEGIN | B}

This specifies that the system begin a run of the hardware event data collection for a system at the System z10 Enterprise Class level or higher. Note that you must first start the HIS address space with the **START hisproc** command before you issue the **F hisproc,BEGIN** command to start hardware event data collection.

END | E

This specifies that the system end a run of hardware event data collection. As part of the end processing, the system writes the hardware data to your UNIX System Services output file and writes the last SMF record type 113, subtype 2 record to the SMF data set.

Note that although both the **F hisproc,END** command and the **STOP hisproc** command end the data collection run, the two commands do not produce identical results:

Using the **F hisproc,END** command allows you to restart hardware collection with the **F hisproc,BEGIN** command.

Using the **STOP hisproc** command both ends the hardware collection and stops the HIS address space. You must reissue the **START hisproc** command before starting data collection with the **F hisproc,BEGIN** command.

TITLE | TT ='*textdata*' This is an optional parameter specifying up to 32 bytes of text data that is meaningful to the user. This data will be displayed in the UNIX System Services output file. You can, for example, use this field to create an “eye-catcher” to identify an instance of a hardware data collection run. The text data must be enclosed in single quotes.

PATH='*pathname*' This specifies the UNIX System Services path (in a local filesystem) where you want the system to write the collected hardware event data for one run. The system creates the file and writes the collected data to this file at the end of a run. This parameter is required, unless you have already set up the file path using the HOME keyword in the OMVS segment of an **ADDUSER** hisproc or **ALTUSER** hisproc command. See "Setting up hardware event data collection" on page 198 for more information.

The pathname must be enclosed in single quotes and can be up to 64 characters. For example, you can specify the following for *pathname*:

PATH='.', which means to use the current working directory.

PATH='/u/john'

PATH='user/mary'

See "Setting up hardware event data collection" on page 198 for information about the files that HIS generates at the UNIX System Services path.

DURATION | DUR duration_value - This is an optional parameter specifying the duration, in minutes, that you want the hardware event data collection run to last. At the end of this period, data collection stops automatically.

The default for DURATION is unlimited time value. Use the default when you want an unlimited data collection run that lasts until you explicitly stop it with one of the following commands:

```
F hisproc,END
```

```
STOP hisproc
```

For a shorter duration data collection run, specify a value for duration_value of 1 - 1440.

CTRSET | CTR {ALL | (ctr1[,ctr2[,ctr3]])}

This is an optional parameter specifying the set of counters you want to collect. ctrn can include the following, where n is B, P, C, E, and ALL:

B for basic - This counter set includes architected system activities, such as cycle count, instruction count, level 1 cache misses, for example, for a CPU in either the problem or supervisor state.

P or PROB for problem state - This counter set includes the architected system activities only when the CPU is in the problem state.

C for Crypto - This counter set includes the architected Crypto activities, such as function count, cycle count, wait function count, and wait cycle count for each of the PRNG, SHA, DEA, and AES functions, for example.

E or EXT for extended - This counter set includes model-dependent counters described in model-dependent system library publications.

ALL - When you specify ALL, the system collects event counters for all available counter sets. For example, if you have installed and authorized only the basic and problem-state counter sets on your system, a specification of CTRSET=ALL results in basic and

problem-state counter set events being collected. Note that ALL is not enclosed in parentheses. If you do not specify CTRSET, HIS uses the B (basic) and P (problem state) counter sets.

DDNAME | DD ddname - This is an optional parameter specifying the 1-to-8-character name identifying the job control language data definition (DD) statement that defines a command file for HIS **MODIFY** hisproc parameters. The command file referenced contains parameters for data collection runs, set up the same way they appear in the **MODIFY** command; the same rules and formatting apply to the command file that you use in the console command. The command file gives you an alternative to specifying data collection options in a **F hisproc** command. This can be useful if you have difficulty fitting all the desired parameters on the command.

When the system performs an HIS data collection run, the system takes the character string on each line and concatenates them into one command string for parsing. It then merges the command file contents with the parameters specified on the **F hisproc** command used to begin HIS data collection. Thus, you can specify some of your data collection options in the command file, and some in the **F hisproc** command. The statements in the command file are normal **F hisproc** parameters, without the **MODIFY hisproc,BEGIN** heading.

CTRONLY | MAPONLY These optional keywords allow you to limit collection, as follows:

CTRONLY - This specifies that you want to collect only event counter set data. Data collection for instruction address sampling is not activated. The system generates only a CNT UNIX System Services output file. (The system does not generate .SMP and .MAP output files if you specify CTRONLY.) When this keyword is specified, you cannot specify keywords associated with instruction sampling, such as SAMPTYPE and SAMFREQ.

Keywords allowed with CTRONLY include TITLE, PATH, DURATION, CTRSET.

MAPONLY - This specifies that you want to collect only load module mapping information. Data collection event counter sets and instruction address sampling is not activated. The system generates only a MAP UNIX System Services output file. (The system does not generate .SMP and .CNT output files if you specify MAPONLY.) When you specify MAPONLY, you must also specify a MAPASID or MAPJOB list to identify the address spaces for which you want private load module map data. You cannot specify keywords associated with sampling and counter sets, such as SAMPTYPE, SAMFREQ, or CTRSET.

Keywords allowed with MAPONLY include TITLE, PATH, MAPASID, APJOB.

MAPVERBOSE|MAPV This optional keyword specifies that you want the system to produce additional diagnostic information about any errors encountered during the load module mapping phase of data collection. MAPVERBOSE specifies that the system issues system messages to the job log if it encounters multiple errors during load module mapping. You can only specify MAPVERBOSE if you also activate the load module mapping with the MAPONLY, MAPASID, or MAPJOB parameters. If you specify MAPVERBOSE without specifying one of

these load module mapping parameters, the system ignores the MAPVERBOSE parameter.

Default - none; no load module mapping diagnostic information is collected.

BUFCNT | BUF This is an optional keyword specifying the number of sampling buffers (in 4 K pages) per processor for the system. A range of values between 4 - 1024 (pages) is supported. The total number of sampling buffers the system uses is calculated from the BUFCNT specified, as follows:

*(BUFCNT * Number of active processors in the configuration)*

If you specify too small a value for BUFCNT and the system runs out of buffer space, you can lose some sample data. If you specify a frequent sampling frequency on the SAMPFREQ parameter, HIS will consume more sampling buffer space and you will need a higher value for BUFCNT. If you do not specify BUFCNT, the system calculates the number of buffers needed using the number of processors in the configuration and a sample goal rate of 8 million samples in 10 minutes.

SAMPTYPE | ST ALL | (samtype[,samtype]) - This is an optional keyword specifying the sampling functions to be performed. The sampling functions (samtype) supported include:

Basic or B - For basic sampling functions.

Diagnosis or D - For basic and diagnosis sampling.

ALL - When you specify ALL, the system collects for all available sampling functions. For example, if you have installed and authorized only the basic and problem-state counter sets on your system, a specification of CTRSET=ALL results in basic and problem-state counter set events being collected. Note that ALL is not enclosed in parentheses.

Default - Basic.

SAMPFREQ | SF **freq** - This is an optional keyword specifying the frequency for the sampling functions. freq is the total number of samples to be taken in a minute on all active processors in the configuration. For example, a freq value of 500000 specifies a sampling frequency of 500,000 samples per minute. The effective sampling rate is usually smaller than the specified SAMPFREQ, because the specified sampling frequency can only be attained by a system that is 100 percent utilized. Processors in wait state will not produce samples.

Default - 800000, which is equivalent to 8 million samples in 10 minutes.

DATALOSS | DL **IGNORE | STOP** - This is an optional keyword specifying the action you want the system to take when buffer overflow occurs during sampling, resulting in sample data loss. You can specify the following for DATALOSS:

IGNORE - Use this to specify that you want the system to continue with sampling if a buffer overflow condition occurs during sampling. You can abbreviate IGNORE as I.

STOP - Use this to specify that you want to stop sampling if a buffer overflow condition occurs during sampling. If you select this option,

you can reduce the changes of losing data in the event of a buffer overflow by either allocating more buffers for data collection or increasing the priority of the HIS started task. You can abbreviate STOP as S.

Default - IGNORE.

MAPASID | MAS

ALL | (asid1,asid2,...asidn) - This is an optional parameter specifying a list of address space IDs (ASIDs), in hexadecimal, to identify the address spaces for which you want to collect private load module map data. HIS will collect the virtual storage addresses of modules loaded into private virtual storage for the specified ASIDs that are not terminating, swapping, swapped, or inactive. Acceptable hexadecimal values are between X'1' and X'FFFF'.

You can specify up to 32 ASIDs on the MAPASID parameter. If you need to collect data from more than 32 address spaces, you can use the MAPJOB parameter in place of or in addition to the MAPASID parameter. The system supports a total of up to 128 address spaces, including those specified in both the MAPASID and MAPJOB parameters. If you need load module map data for more than 128 address spaces, specify MAPASID=ALL.

Examples:

MAPASID=(7,8,32)

MAPASID=ALL

Default - None.

MAPJOB | MJOB

(job1,job2,...jobn) - This is an optional parameter specifying a list of job names for which HIS will collect the virtual storage addresses of modules loaded into private virtual storage. HIS collects the virtual storage addresses for jobs that are not terminating, swapping, swapped, or inactive. The system ignores duplicate, invalid, or inactive job names specified. Specify job names with 1-8 characters, following the rules for a valid job name. You can use wildcard characters * and ? for pattern matching of job names. You can specify up to 32 job names, including all the pattern matches from wildcard characters. Note that using wildcard characters can result in requesting more than 32 physical jobs.

When you specify job names on MAPJOB, the system converts each active job name into one or more ASIDs. If the job is not active when the system does the load module mapping, HIS will not produce load module information for that job.

The system supports a total of up to 128 address spaces, including those specified on both the MAPASID and MAPJOB parameters. If you specify more than 128 address spaces, the system produces load mapping module data for the first 128 address spaces and ignores the rest. For example, MAPJOB=(*) produces load module mapping information for 128 active address spaces. If you need load module map data for more than 128 address spaces, specify MAPASID=ALL.

Examples:

MAPJOB=(task1,grs,omvs,db*,task2)

MAPJOB=(o*s)

MAPJOB=(JOB1??,JE*)

Default - None.

Example 1 - Start data collection using HIS defaults, including SMF data, and creates .CNT and SMP.cpu#. files. (Note that default collection does not include load module mapping.)

```
F hisproc,BEGIN
```

Example 2 - Start data collection that runs for 12 minutes, collects load module mapping information for jobs PGMA and PGMB. HIS collects SMF data and creates .MAP, .CNT and SMP.*cpu#*. files.

```
F hisproc,BEGIN,DURATION=12,MAPJOB=(PGMA,PGMB)
```

Example 3 - End HIS data collection with the following command:

```
F hisproc,END
```

8.4.2 Displaying hardware event data collection status

Use the **D HIS** command to display the results of the latest hardware event data collection run initiated by an **F hisproc,BEGIN** command; Figure 8-4 shows the command syntax. Hardware event data collection is performed by Hardware Instrumentation Services (HIS), which collects hardware event data for processors in SMF records type 113, subtype 2, as well as a UNIX System Services output file.

```
D HIS[,L=name|name-a|a]
```

Figure 8-4 Syntax of D HIS command

Note that the results displayed by the **D HIS** command in system message HIS015I message will only be as current as the last time you initiated a hardware event data collection run with the **F hisproc,BEGIN** command.

If you issue the **D HIS** command to display the results of the last hardware event data collection run while the collection is still in progress, the data displayed can be incomplete. The example in Figure 8-5 on page 208 shows output from a **D HIS** command issued while data collection was still running. Because data collection was not complete when the D HIS command was issued, the system had not yet converted the MAPJOB job names to MAPASID values at the time when the **DISPLAY** command was issued, so the MAPASID values are not displayed in the output.

```
HIS015I 18.41.33 DISPLAY HIS 201
HIS 0023 ACTIVE
COMMAND: MODIFY HIS,B,TT='Sampling',BUF=25,PATH='/user',MJOB=(OMVS,J*,GRS)
START TIME: 2007/11/08 18:41:29
END TIME: ----/--/-- --:--:--
COMPLETION STATUS: --------
FILE PREFIX: SYSHISS20071108.184129
LOST SAMPLES: 0 COUNTER VERSION NUMBER 1: 1 COUNTER VERSION NUMBER 2: 1
COMMAND PARAMETER VALUES USED:
TITLE= Sampling
PATH= /user
COUNTER SET= BASIC,PROBLEM-STATE
DURATION= 10 (MINUTES)
BUFCNT= 25 (PAGES/PROCESSOR)
SAMPTYPE= BASIC
SAMPFREQ= 800000 (SAMPLES/MINUTE)
DATALOSS= IGNORE
MAPJOB= OMVS J* GRS
```

Figure 8-5 MAPASID values are not displayed

In the example of Figure 8-6, the **DISPLAY** command was issued after a hardware event data collection run that specified MAPONLY as well as MAPASID and MAPJOBs.

```
HIS015I 18.41.33 DISPLAY HIS 201
HIS 0023 IDLE
COMMAND: MODIFY HIS,B,TT='map-only',MAPONLY,MAS=(1,EC),MJOB=(J*,GRS)
START TIME: 2007/11/08 13:41:29
END TIME: 2007/11/08 13:41:30
FILE PREFIX: SYSHISS20071108.134129
COMMAND PARAMETER VALUES USED:
TITLE= map-only
PATH= .
MAPONLY
MAPASID= 0001 0007 0010 0016 001A 001C 00EC
MAPJOB= J* GRS
```

Figure 8-6 MAPONLY, MAPASID, and MAPJOB are displayed

The **D HIS** command output in Figure 8-7 on page 209 shows a counters-only run, to collect only event counter set data, specifying ALL the counter sets. Note that the **DISPLAY** command was issued while the data collection was still ongoing.

```
HIS015I 19.18.33 DISPLAY HIS 201
HIS 0022 ACTIVE
COMMAND: MODIFY HIS,B,CTRSET=ALL,TITLE='Counters only',CTRONLY
START TIME: 2007/11/09 19:17:55
END TIME: ----/--/-- --:--:--
COMPLETION STATUS: --------
FILE PREFIX: SYSHISS20071109.191755
COUNTER VERSION NUMBER 1: 1 COUNTER VERSION NUMBER 2: 1
COMMAND PARAMETER VALUES USED:
TITLE= Counters only
PATH= .
COUNTER SET= BASIC, PROBLEM-STATE, CRYPTO-ACTIVITY, EXTENDED
DURATION= NOLIMIT
CTRONLY
```

Figure 8-7 Collecting only event counter set data - specifying ALL counter sets

8.4.3 Accessing the output from a hardware event data collection run

For each data collection run, the system generates UNIX System Services output files in the `/user HOME` directory that follow these naming conventions:

- SYSHISyyyymmdd.hhmmss.CNT
- SYSHISyyyymmdd.hhmmss.MAP
- SYSHISyyyymmdd.hhmmss.SMP.cpu#

The files that the data collection run generates will depend on the **`F hisproc`** parameters that were specified when the run was started.

Where:

- *yyyymmdd* is the year, month, and day when the **`F hisproc`** command was processed.
- *hhmmss* is the hour, minute, and second when the **`F hisproc`** command was processed.
- **CNT | MAP | SMP.***cpu#* identify the file, as follows:
  - CNT identifies a counter set data file.
  - MAP identifies a load module mapping output file.
  - SMP.cpu# identifies a sampling function data file. *cpu#* is the CPU number, in hexadecimal. There is one .SMP file for each active CPU.

For example, the system creates the following UNIX System Services files for a system with three CPUs at 11:30:16 on 2007/05/15:

```
SYSHIS20070515.113016.CNT
SYSHIS20070515.113016.MAP
SYSHIS20070515.113016.SMP.0 (for cpu 0)
SYSHIS20070515.113016.SMP.1 (for cpu 1)
SYSHIS20070515.113016.SMP.2 (for cpu 2)
```

Table 8-1 on page 210 shows when HIS generates the .MAP, .SMP, and .CNT files, based on the **`F hisproc,BEGIN`** parameters.

Table 8-1 HIS generation of output files

.CNT file	.MAP file	.SMP file
HIS generates a .CNT file if any of the following **F hisproc,BEGIN** parameters have been used:	HIS generates a .MAP file if any of the following **F hisproc,BEGIN** parameters have been used:	HIS generates a .SMP file if any of the following **F hisproc,BEGIN** parameters have been used:
► CTRSET ► CTRONLY - Note that HIS does not generate .SMP and .MAP output files if CTRONLY has been specified.	► MAPASID ► MAPJOBS ► MAPONLY - Note that HIS does not generate .SMP and .CNT output files if MAPONLY has been specified.	► Default to or specify SAMPTYPE, unless CTRONLY or MAPONLY has been specified.

To access the output that HIS generates in your UNIX System Services file in your `/user HOME` directory, use the **OBROWSE** command on the file, or else use OGET to copy the file to MVS and access the output there.

Example HIS output is shown here:

```
HIS019I EVENT COUNTERS INFORMATION
FILE NAME: SYSHIS20080115.093607.CNT
COMMAND: MODIFY HISPROC,B,TITLE='Test Run A',PATH='/u/hisuser'
COUNTER SET= BASIC
COUNTER IDENTIFIERS:
0: CYCLE COUNT
1: INSTRUCTION COUNT
2: L1 I-CACHE DIRECTORY-WRITE COUNT
4: L1 D-CACHE DIRECTORY-WRITE COUNT
START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 00:
0- 3 00015C4E2D5A9AFE 000052A9E5C5F17C 000000090C2B9D6C ----------------
4- 7 0000001558 93B7CC ---------------- ---------------- ----------------
START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 01:
0- 3 00015C4DF934F93F 000052B7005653F1 00000008EF94E3F9 ----------------
4- 7 0000001528FAEC68 ---------------- ---------------- ----------------

START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 02:
0- 3 00015C4E64C8BB2E 00005380E73CD936 000000070E31DAE5 ----------------
4- 7 0000001174DB114F ---------------- ---------------- ----------------

START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35 COUNTER VALUES (HEXADECIMAL) FOR CPU 03:
0- 3 00015C4E67F7501C 000053A1E7808B1F 00000006D28CFD0C ----------------
4- 7 0000001109BE6B77 ---------------- ---------------- ----------------

COUNTER SET= PROBLEM-STATE
COUNTER IDENTIFIERS:
32: PROBLEM-STATE CYCLE COUNT
33: PROBLEM-STATE INSTRUCTION COUNT
```

```
START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 00:
32- 35 00014797F6C56C78 00004F9FFA25194B ---------------- ----------------

START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 01:
32- 35 000147EE03F60B99 00004FB4110096E8 ---------------- ----------------

START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 02:
32- 35 00014C59FF8C7CE2 00005128F9333CDB ---------------- ----------------

START TIME: 2008/01/15 09:36:07
END TIME: 2008/01/16 09:45:35
COUNTER VALUES (HEXADECIMAL) FOR CPU 03:
32- 35 00014CBFF53BFF8F 0000515282E2F0F0 ---------------- ----------------
```

Interpreting load module output in a .MAP file

You can request load module mapping information in your data collection run using any of the following parameters on the **F hisproc,BEGIN** command:

- MAPONLY
- MAPASID
- MAPJOB
- SAMTYPE

HIS returns the load module mapping data in UNIX System Services SYSHISyyyymmdd.hhmmss.MAP output file in your `/user HOME` directory.

This load module mapping data contains information about the virtual address ranges of various modules loaded in private storage on the system. You can request this module address information for one address space, for several address spaces, or for all active address spaces by using the **F hisproc,BEGIN MAPASID** and **MAPJOB** parameters. The .MAP file also provides information about the virtual addresses of Nucleus CSECTs and modules loaded into LPA. Modules can also be further divided into CSECTs.

The data in a .MAP file is useful in exploiting the other information HIS returns in a data collection runs. For example, HIS generates .SMP files containing virtual addresses. The module map allows you to determine how many of these samples are associated with a specific module, which helps you estimate the relative amount of activity in the module.

For example, assume that module A is an LPA module that starts at x'00CC7000' and ends at x'00CC73FF'. When looking at the sample data provided by HIS, it is possible to see that 50,000 of the 1,000,000 samples provided by HIS have virtual addresses between x'00CC7000' and x'00CC73FF'. Based on this, it can be estimated that 5% of the CPU time is spent in module A during the time that HIS is capturing data.

Figure 8-8 on page 212, Figure 8-9 on page 212, and Figure 8-10 on page 213 show possible .MAP output.

```
        AREA        START ADR   END ADR     LENGTH      TYPE
        PRIVATE     00000000    008FFFFF    09437184    BOUNDARY
        CSA         00900000    00BB2FFF    02830336    BOUNDARY
        CSAALLOC    00029EA0    00E069A8    14535433    BOUNDARY
        CSACONVT    00000000    00000000    00000001    BOUNDARY
        MLPA        00BB3000    00BC9FFF    00094208    BOUNDARY
        FLPA        00000000    00000000    00000001    BOUNDARY
        PLPA        00BCA000    00D70FFF    01732608    BOUNDARY
        SQA         00D71000    00FD5FFF    02510848    BOUNDARY
        SQAALLOC    0006FA38    007894D0    07445145    BOUNDARY
        RWNUC       00FD6000    00FE336F    00054128    BOUNDARY
        RON         00FE4000    00FFFFFF    00114688    BOUNDARY
        ERON        01000000    019621BF    09839040    BOUNDARY
        ERWN        01963000    019DBFFF    00495616    BOUNDARY
        ESQA        019DC000    026C5FFF    13541376    BOUNDARY
        EPLPA       026C6000    05E49FFF    58212352    BOUNDARY
        EFLPA       00000000    00000000    00000001    BOUNDARY
        EMLPA       05E4A000    06592FFF    07639040    BOUNDARY
        ECSA        06593000    259FFFFF    24734464    BOUNDARY
        EPRV        25A00000    7FFFFFFF    16240896    BOUNDARY
        DONUC       075A8000    075ABFFF    00016384    BOUNDARY
```

Figure 8-8 .MAP output 1/3

```
LPA     IGG019FR    00BA7CC8    00BA7DBF    00000248    MODULE
LPA     IGG019FS    00BA7DC0    00BA81AF    00001008    MODULE
LPA     IGG019FU    00BA81B0    00BA836F    00000448    MODULE
LPA     IGG0197L    00BA8370    00BA877F    00001040    MODULE
LPA     IGG0197M    00BA8780    00BA8B7F    00001024    MODULE
LPA     IGG0197N    00BA8B80    00BA90A7    00001320    MODULE
LPA     IGG0197P    00BA90A8    00BA957F    00001240    MODULE
LPA     IGG0197Q    00BA9580    00BA9867    00000744    MODULE
LPA     IGG0201P    00BA9868    00BAA057    00002032    MODULE
LPA     IGG0201R    00BAA058    00BAA2F7    00000672    MODULE
LPA     IEFIB600    00BB3BC8    00BBF3BF    00047096    MODULE
LPA     CEEBINIT    00BBF3C0    00BC9FF7    00044088    MODULE
        CEEBINIT    00BBF3C0    00BC0A57    00005784    CSECT       24      ANY
        CEECPYRT    00BC0A58    00BC0B39    00000226    CSECT       24      ANY
        CEEANCH     00BC0B40    00BC0BBB    00000124    CSECT       24      ANY
LPA     IGG019T8    00BDE268    00BDE347    00000224    MODULE
LPA     IGG019TX    00BDE348    00BDE3E7    00000160    MODULE
LPA     IGG019JD    00BDE3E8    00BDE4C7    00000224    MODULE
LPA     IGG019BT    00BDE4C8    00BDE5B7    00000240    MODULE
```

Figure 8-9 .MAP output 2/3

```
          JOBNAME = OMVS      ASID = 000E

NAME        START ADR   END ADR     LENGTH      TYPE      AMODE  RMODE  COMME
BPXINPVB    00006000    00006BA7    00002984    MODULE
IEAVLSUP    2562A428    2562AFFF    00003032    MODULE
BPXINPVT    25A05000    25D326C7    03331784    MODULE
BPXINPV2    25D33000    25D75837    00272440    MODULE
BPXMIMSK    25D77000    25D7EBA7    00031656    MODULE
BPXMERNO    25D7F000    25DD0F47    00335688    MODULE
IGWASMS     266BF2F0    266BF4DF    00000496    MODULE
IGWASMS     266BF2F0    266BF4DF    00000496    CSECT       24     ANY
BPXVOSIT    26838000    2683ABCF    00011216    MODULE
BPXHCMSG    2684D000    26852DCF    00024016    MODULE
GFUAPFS1    268A9000    2698863F    00915008    MODULE
IGWLQ000    26989000    2698FF9F    00028576    MODULE
BPXTFS      26998000    269B25FF    00108032    MODULE
          JOBNAME = HIS       ASID = 0025

NAME        START ADR   END ADR     LENGTH      TYPE      AMODE  RMODE  COMME
AIRMFRR     251A5808    251A593F    00000312    MODULE
AIRMMOD     251A5940    251A5D37    00001016    MODULE
AIRMSRB     251A5D38    251A5FFF    00000712    MODULE
HISINIT     25A00000    25A02DDF    00011744    MODULE
HISINPVT    25A08000    25A1164F    00038480    MODULE
AIRMREQ     25A16000    25A25C8F    00064656    MODULE
AIRMRVA     36A88000    36A89207    00004616    MODULE
```

Figure 8-10 .MAP output 3/3

8.5 Installation considerations

The z10 hardware instrumentation and its support in z/OS facilitates software performance tuning for both system and application software with a broad scope. The Central Processor Measurement Facility is nondisruptive, has low overhead, and can run in multiple LPARs simultaneously. It provides an easy estimation of performance characteristics and a quick determination of potential performance problems. With adequate tooling, this provides a promising means to further tune software performance parameters, coding sequence, and data allocation at a fine granularity.

Significant performance improvement can be derived and achieved in client shops with immediate feedback and little effort.

9

Service aids enhancements

MVS service aids are a group of tools to aid in detecting problems, gathering documentation needed to solve the causes, and communicate with support locations remote from where materials can have originally been collected.

This chapter describes the following service aids enhancements:

- SDUMP storage management enhancement
- SDUMP timer DIE enhancement
- SDUMP start message
- Service aids health checks
- Service aids support of extended address volumes (EAV) in z/OS V1R11
- Removal of IPCS problem management

9.1 SDUMP storage management enhancement

The SDUMP storage management enhancement in z/OS V1R11 provides function to detect excessive auxiliary storage usage by SDMUP processing. The MAXSPACE value restricts the virtual storage available to the DUMPSRV address space. When you use MAXSPACE, you must tune for the worst case usage of real and auxiliary storage. The maximum size of a dump is determined by the MAXSPACE parameter specified by using the **CHNGDUMP** command:

```
CHNGDUMP SET,SDUMP,MAXSPACE=xxxxxxxxM
```

The default for MAXSPACE is 500 M.

Note: Use a multiple of the data set size to determine a MAXSPACE value. The installation must predict the size of the largest dump that it can configure for.

SDUMP dump sizes are increasing with applications like WebSphere and DB2 generating dumps in the 3 GB range. When a dump is captured it must be backed up by auxiliary storage and so the amount of page space defined needs to consider the MAXSPACE specified.

Prior to z/OS V1R11, the dump is truncated when the MAXSPACE was reached or when SRM detected that 85% of auxiliary storage has been used. The general guideline is that the ratio between MAXSPACE and auxiliary storage is to be 1:3. For example, if MAXSPACE=8000 M is specified then 24000 M of paging space is required to ensure that there is enough auxiliary storage to contain the worst case usage by captured dumps.

If there is insufficient auxiliary storage allocated, then the detection of a critical auxiliary storage shortage due to SDUMP processing will possibly not occur quickly enough. This can result in a system outage with a WAIT state 03C, reason code 01.

SDUMP processing in z/OS V1R11 has been enhanced to monitor auxiliary storage usage during SDUMP processing to allow detection of excessive auxiliary storage usage.

DUMPSRV address space

An SVC dump begins as a snapshot of volatile system data into DUMPSRV-owned virtual storage, which is represented by DUMPSRV dataspaces and DUMPSRV high virtual storage within the DUMPSRV address space. As such, the collection of data can represent an unusually heavy load upon the storage resources.

The virtual storage burden remains until the dump is written out to a target data set on DASD, which can be an SVC dump data set that is specified on the DCB parameter of the SDUMP or SDUMPX macro, a pre-allocated SYS1.DUMPxx data set, or an automatically allocated dump data set. The **DUMPDS** operator command allows the installation to manage the pre-allocated and automatically allocated data sets.

Important: If normal auxiliary storage utilization is above 30%, the system might be subject to severe performance impacts. The system might even experience a WAIT03C state when SVC dumping occurs, which indicates that the system ran out of available paging slots.

9.1.1 AUXMGMT keyword for the CHNGDUMP command

You can specify the MAXSPACE and the new AUXMGMT options on the **CHNGDUMP SET, SDUMP** command to manage the burden of taking SVC dumps on a system. However, using

them might not be sufficient to eliminate all problems associated with restricted auxiliary (paging) storage availability.

z/OS V1R11 has a new function to monitor auxiliary storage usage through the SDUMP options keyword AUXMGMT. However, the current documentation is unclear regarding the intended behavior when AUXMGMT is changed from ON to OFF.

In the case where AUXMGMT=ON and new SVC dumps are prevented because auxiliary storage usage has exceeded 50%, then the following message is issued:

```
IEA412I SLIP TRAP ID=GK01,    1 SDUMPS NOT SCHEDULED.  RETURN CODE=08, REASON
CODE=46.
```

The new keyword added to the **CHNGDUMP** command to control the SDUMP storage management enhancement is used as follows:

```
CHNGDUMP SET,SDUMP,AUXMGMT=ON|OFF
```

Where:

ON No new dumps are allowed when auxiliary storage usage reaches 50%. New dumps are allowed again only after the auxiliary storage usage drops below 35%. Current SDUMP data capture stops when auxiliary storage usage exceeds 68%, generating a partial dump.

For systems where large SVC dumps are typically generated, it is suggested to set MAXSPACE at 8000 megabytes.

OFF SVC dump virtual storage management is under the control of the MAXSPACE limitations. Dumps in progress are stopped when MAXSPACE is exceeded, or when auxiliary storage utilization exceeds 85%. See the topic “Obtaining SVC dumps” in *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589, for a more detailed discussion about using the AUXMGMT and MAXSPACE keywords.

Note: There are two points of auxiliary storage shortage detection.

- For AUXMGMT=ON, the shortage is considered to occur at 50%.
- For AUGMGMT=OFF, the shortage is considered to occur at 85%.

After it is detected, the condition can only be reset by managing the paging resources. It cannot be reset by changing the AUSMGMT value.

AUXMGMT considerations

Although it would seem that if SVC dump (SDUMP) processing detects an auxiliary storage shortage when AUXMGMT=ON, then setting AUXMGMT=OFF will reset the condition and allow SDUMP to use more auxiliary storage until the higher detection point. However, that is not the case. The default is AUXMGMT=ON, which makes availability a higher priority over first-failure data capture.

When AUXMGMT=ON or is defaulted to:

- New dumps will not be taken when auxiliary storage usage reaches 50%
- The current dump capture is truncated when the auxiliary storage usage reaches 68%.
- After the limit is exceeded, new dumps will not be allowed until the AUX storage utilization drops below 35%.

- MAXSPACE is honored when it is more restrictive than AUXMGMT. For example if MAXSPACE=35M was specified, then SVC dump processing stops even though auxiliary storage utilization is very low when MAXSPACE is exceeded.

To revert to SDUMP virtual storage management using MAXSPACE, only a CHNGDMP command specifying AUXMGMT=OFF must be issued:

```
CHNGDUMP SET,SDUMP,AUXMGMT=OFF
```

After this command is issued and a dump is in progress:

- SDUMP processing stops and the dump is flagged as partial when MAXSPACE is exceeded or the auxiliary storage utilization exceeds 85%.
- After exceeding MAXSPACE, there must be 35 MB of MAXSPACE available before dump capture can resume.
- After critical auxiliary storage shortage, auxiliary storage utilization must be 35% or less before dump capture can resume.

SDUMP processing stops

When SDUMP processing is stopped due to excessive auxiliary storage usage, the following new IEE711I messages are issued explaining why a dump is not taken:

```
IEE711I SYSTEM DUMP NOT TAKEN. A CRITICAL AUXILIARY STORAGE SHORTAGE EXISTS
```

Note: SVC dump will not allow another dump to be captured until the shortage of auxiliary storage is relieved. First you need to ensure that enough DASD resource is available for captured dumps to be written out. Then, consider adding additional auxiliary storage (paging) resources, because SVC dumps will not be allowed again until the auxiliary storage utilization drops below 35%.

The second possible IEE711I message follows:

```
IEE711I SYSTEM DUMP NOT TAKEN. DUMPSRV PROCESSING IS UNAVAILABLE
```

Note: DUMPSRV might have processed a CANCEL request and will restart shortly. Otherwise, DUMPSRV has either terminated or the primary task is unavailable. Notify the system programmer. SDUMP issuing applications will receive RC=08 RSN=0C.

The third possible new IEE711I message is:

```
IEE711I SYSTEM DUMP NOT TAKEN. DUMPSRV PROCESSING IS UNAVAILABLE
```

Note: DUMPSRV might have processed a CANCEL request and will restart shortly. Otherwise, DUMPSRV has either terminated or the primary task is unavailable. Notify the system programmer. SDUMP issuing applications will receive RC=08 RSN=0C.

System programmer response: If the reason field shows MAXSPACE LIMIT REACHED, take one of the following actions and then issue the **DUMP** command again:

- Add more dump data sets. Use the DUMPDS command or another utility to make more dump data sets available. The system writes captured dumps to available dump data sets, freeing storage for the next dump.
- Delete existing captured dumps. If any dump is captured but they are not required by the installation, reply D to message IEA793A to delete the dumps.
- Increase the MAXSPACE value. Enter the CHNGDUMP command to increase the value of MAXSPACE, after ensuring that the available auxiliary storage (paging slots) is at least three times the MAXSPACE space.

9.2 SDUMP timer DIE enhancement

The CHNGDMP keyword Q=YES specifies that the system is set to non-dispatchable while dumping common storage, SQA/ESQA and CSA/ECSA.

SDUMP establishes a timer Disabled Interrupt Exit (DIE) to protect the system and tasks of the address space from being left non-dispatchable due to an unrecoverable error in SDUMP processing. The DIE routine receives control (disabled) at a specified interval and determines if SDUMP processing is hung while the system or tasks of the address space are set non-dispatchable. This is done by examining a count of the number of records captured for an SVC Dump. If the count remains the same across the DIE time interval, SDUMP is considered hung and calls the STATUS service to reset the system and the tasks in the affected address spaces as dispatchable.

For large dumps on systems with large amounts of global storage, the global capture phase can take an excessive amount of time. If the system is non-dispatchable for an extended period of time many critical applications can time out and critical system functions can be delayed, thus causing a system outage.

The SDUMP timer DIE enhancement adds another factor in determining when the system is to be set dispatchable during global storage capture to avoid system impact.

9.2.1 MAXSNDSP keyword for the CHNGDUMP command

A new keyword, MAXSNDSP, is added to the **CHNGDUMP** command to limit the amount of time the system can remain non-dispatchable during SDUMP global storage capture:

```
CHNGDUMP SET,SDUMP,MAXSNDSP=nn
```

The default value is 15 seconds.

Note: If the count of the number of records captured for an SVC dump remains the same across the DIE time interval *or* the duration of the system being set non-dispatchable exceeds the MAXSNDSP specification, then SDUMP will reset the system to dispatchable and dump processing will continue.

SDRSNDSP bit

A new bit, SDRSNDSP, is set in byte SDRSDBND of IHASDRSN, if the DIE resets the system to dispatchable due to MAXSNDSP being exceeded. This is the first byte of word 3 in the

SDUMP reason codes. When SDRSNDSP is set, a SNAPTRC will be taken at the end of the global capture to allow diagnosis of the processing during global data capture.

To view the SNAPTRC, an **IPCS SYSTRACE** command must be issued for the corresponding system trace table snapshot copy header (TTCH). The TTCH can be found by issuing the following IPCS command:

```
IPCS STATUS WORKSHEET
```

The messages issued for a dump that exceeded the MAXSNDSP value are shown in Figure 9-1 on page 220. As shown in the figure, the first byte of word 3, SDRSDBND, = X'84'. SDRSNDSP is the X'04' bit, which shows the MAXSNDSP was exceeded.

From the IPCS STATUS display:

```
System reset nondispatchability Trace Table Control header address 7F541000
```

To format this TTCH, issue the following command:

```
SYSTRACE ALL TTCH(X'7F541000') TI(LO)
```

The following message is not related to MAXSNDSP being exceeded. It is issued when SDRTDISP is set in the first byte of word 3, SDRSDBND. SDRTDISP is the X'80' bit.

```
SYSTEM RESET DISPATCHABLE PRIOR TO DUMP COMPLETION
```

```
IEA045I AN SVC DUMP HAS STARTED AT TIME=09.06.14 DATE=05/05/2009 833
FOR ASIDS(0001,0022,0023)
QUIESCE = YES
IEA794I SVC DUMP HAS CAPTURED: 834
DUMPID=002 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=SVA TESTING
IEF196I IEF237I 0180 ALLOCATED TO SYS00005
IRA205I 50% AUXILIARY STORAGE ALLOCATED
IEF196I IEF285I   SYS1.DUMP.D090505.T130614.SP7E.S00002        CATALOGED
IEF196I IEF285I   VOL SER NOS= D16PK6.
IEA611I COMPLETE DUMP ON SYS1.DUMP.D090505.T130614.SP7E.S00002 839
DUMPID=002 REQUESTED BY JOB (*MASTER*)
FOR ASIDS(0001,0022,0023)
INCIDENT TOKEN: LOCAL    SP7E     05/05/2009 13:06:14
SDRSN = 00000000 00000000 00000000 84000000
SYSTEM RESET DISPATCHABLE PRIOR TO DUMP COMPLETION
```

Figure 9-1 Syslog messages for a dump with SDRSNDSP set

9.3 SDUMP start message

Prior to z/OS V1R11, SDUMP issued a message when data capture was complete but there was no message to indicate the start of data capture. With z/OS V1R11, the message shown in Figure 9-2 is issued when data capture starts.

```
IEA045I  AN SVC DUMP HAS STARTED AT TIME=hh:mm:ss DATE=mm/dd/yyyy
         FOR ASIDS(xx,xx,...,xx)
         ERRORID=SEQxxxxx CPUzz ASIDaa TIMEhh:mm:ss.t
            QUIESCE=YES/NO
```

Figure 9-2 SDUMP start message, IEA045I

9.4 AutoIPL health checks

New AutoIPL health checks in z/OS V1R11 provide early warning if AutoIPL is not specified. They validate the SAD and IPL devices if an AutoIPL policy is specified. This ensures that the AutoIPL function is available, thereby providing a RAS improvement and improving first failure data capture.

9.4.1 AutoIPL overview

AutoIPL quickly and automatically initiates a stand-alone dump (SAD), a reIPL of z/OS, or both when a z/OS system enters a disabled wait state. This enhancement was introduced in z/OS V1R10, as follows:

- An AutoIPL policy is specified using the DIAGxx parmlib member.
- A hardcoded wait state action table designates disabled wait state codes and reason codes for which AutoIPL processing is (or is not) to be performed, and which actions to be taken.
- Most non-restartable disabled wait states or reasons are eligible for AutoIPL processing.
- The VARY XCF commands support new options to IPL stand-alone dumps, reIPL 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.

DIAGxx parmlib member statement

The statement syntax is:

```
AUTOIPL SADMP(sadmp info) MVS(mvs info)
```

sadmp info This 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 This 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, then 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.

9.4.2 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 stand-alone dump.
- Initiate a stand-alone dump, followed by reIPL of z/OS.
- Initiate a reIPL 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 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 stand-alone dump, reIPL 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 through 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 another wait state, and has not already been system reset, manually reIPLed, or fenced at the time of the **VARY XCF** processing.

By default, systems removed from the sysplex using **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)
```

9.4.3 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.

9.4.4 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 stand-alone dump or initiate an IPL of z/OS.

Although the WSAT is not a programming interface, it will be externalized so that clients 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 is to be honored, and also a flag to indicate whether the MVS part of the policy is to 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 indicates it is to 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”, note the following information:

- Nibble 1 - contains the AutoIPL action in the last two bits:
  - b’00’ - no action
  - b’10’ - Stand-alone dump
  - b’01’ - IPL z/OS
  - b’11’ - Stand-alone 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 9-1 shows the entries in the WSAT and explains the action taken for each wait state and reason code combination.

Table 9-1 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 9-3. It can be found by 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 9-3 Viewing the WSAT in BLWWSATC using IPCS

Hardware and software requirements

Hardware support for DIAGNOSE 308 (program-directed IPL support) is required. This support is available on the z10 and z9 machines.

AutoIPL is supported with z/OS V1R10 and higher only. Rollback APARs are not provided for earlier releases.

9.4.5 AutoIPL availability health check for z/OS V1R11

The new health checks available with z/OS V1R11 can determine if AutoIPL is available by checking that the hardware requirements are satisfied, and by checking whether an AutoIPL policy has been specified in the client implementation. The following two health checks determine whether AutoIPL is applicable:

- Do not use AutoIPL in a GDPS environment because GDPS must be the sole manager of IPLing the GDPS systems.
  - The AutoIPL function uses DIAGNOSE 308, which is standard on z10 and optional on z9 hardware.
- If AutoIPL is applicable, then the AutoIPL health check issues a warning message if an AutoIPL policy has not been activated.

The syntax of the AutoIPL availability health check is shown in Figure 9-4.

```
CHECK(IBMSVA, SVA_AUTOIPL_DEFINED)
       Severity(MED)
       INTERVAL(02400)
       ACTIVE
       DEBUG(OFF)
       DATE(20081001)
       REASON('To ensure Auto-IPL function is in use when available')
```

Figure 9-4 Syntax of the AutoIPL availability health check

AutoIPL health check output messages for DIAGNOSE 308

An informational message is issued to indicate that DIAGNOSE 308 is not supported:

```
BLWH008I List-Directed IPL or Program-Directed IPL is not supported
```

If AutoIPL is supported, then an informational message is issued if an AutoIPL policy is active:

```
BLWH009I AutoIPL policy is active
Explanation: The check(IBMSVA, SVA_AUTOIPL_DEFINED)  found an active AutoIPL
policy.
```

If AutoIPL is supported, then an exception message is issued if an AutoIPL policy is not active:

```
BLWH001E AutoIPL policy is not active
Explanation: Check(IBMSVA, SVA_AUTOIPL_DEFINED) found no active AutoIPL policy.
IBM suggests activating an AutoIPL policy using a DIAGxx parmlib member.
Installations can activate the AutoIPL function so that the system will take
predefined actions automatically when it is about to enter certain disabled
wait states.  Action(s) can be to re-IPL z/OS, or to take a Stand  Alone dump
(SADMP), or to take a SADMP and have SADMP re-IPL z/OS when it has finished.

System Programmer Response: Specify an AutoIPL policy via a DIAGxx parmlib
member and activate it by issuing a SET DIAG=xx operator command.  Issue
DISPLAY DIAG to display information about the current DIAGxx parmlib settings.
```

AutoIPL device validation health check

If SADMP or MVS (or SADMP and MVS) are specified on the AutoIPL statement in the DIAGxx parmlib member, then the AutoIPL device validation health check will validate the devices specified. The following conditions are checked and an exception message is issued if any condition fails:

- Device is not available
- Device is not a disk device
- Device is defined as a secondary device in a metro mirror pair

The syntax of the AutoIPL device validation health check is shown in Figure 9-5 on page 226.

```
CHECK(IBMSVA, SVA_AUTOIPL_DEV_VALIDATION)
        Severity(MED)
        INTERVAL(02400)
        ACTIVE
        DEBUG(OFF)
        DATE(20081001)
        REASON('To validate the device information in the Auto-IPL policy')
```

Figure 9-5 Syntax of the AutoIPL device validation health check

The following message is issued if the AutoIPL device validation health check finds that the devices are valid:

```
BLWH010I AutoIPL policy device(s) are valid
Explanation: The check(IBMSVA, SVA_AUTOIPL_DEV_VALIDATION)  found the AutoIPL
policy device(s) to be valid
```

The following messages are issued if a problem is found with a device specified in the AutoIPL policy:

```
BLWH002E A problem was found for a device specified in the AutoIPL policy
Explanation: The check(IBMSVA, SVA_AUTOIPL_DEV_VALIDATION) found a problem
during device validation for a device specified in the AutoIPL policy This
message is followed by message BLWH901I, which lists information about invalid
device(s) specified in the AutoIPL policy.  All of the following conditions
must be met for the device to pass device validation:
 - Must be DASD
 - Must not be specified as a secondary device in a Metro Mirror pair.
 - Must be accessible
```

```
System Programmer Response: Examine logs to determine which AutoIPL policy
device(s) do not pass the device validation.  Resolve the problem either by
specifying a new device in the DIAGxx that meets the criteria described above
or by updating the existing device characteristics.  Cause MVS to read the
DIAGxx member by issuing a SET DIAG =xx operator command

BLWH901I A problem was found with the following AutoIPL device(s).
     AutoIPL action     = The AutoIPL action (SADMP or MVS).
     Device Address = The device address.
     Error =The description of the problem

Explanation: The check(IBMSVA, SVA_AUTOIPL_DEV_VALIDATION)  displays
information needed  to correct invalid AutoIPL device(s)
```

9.5 Dump analysis and elimination health checks

Dump analysis and elimination (DAE) suppresses dumps when enough symptoms are available to determine that the new request matches a dump already taken. This assists storage management by minimizing the system resources required to support first failure data capture. It reduces the processing resources required to collect the data, and reduces the disk resources required for preserving the data.

DAE uses the ADYSETxx parmlib member to determine the actions that DAE is to perform. Have the ADYSETxx member specify SUPPRESSALL, which requests that dumps be suppressed even though the component or program did not request dump suppression with a VRADAE key in the system diagnostic work area (SDWA). SUPPRESSALL is useful because it allows more dumps to be eligible for suppression.

For a sysplex, use the same ADYSETxx parameter values in each system. DAE can suppress a duplicate of a previous dump from any system when the systems in the sysplex share a DAE data set. Use a shared PARMLIB, or identical ADYSETxx members specifying SHARE(DSN), to share the same DAE data set.

9.5.1 DAE health checks

With z/OS V1R11, new health checks are introduced to ensure that DAE is active to prevent duplicate dumps and validate that DAE is properly configured:

- CHECK(IBMDAE,DAE_SUPPRESSING
- CHECK(IBMDAE,DAE_SHAREDSN)

DAE availability health check

The DAE availability health check raises an exception If DAE is not active. The syntax of the DAE availability health check is shown Figure 9-6.

```
CHECK(IBMDAE, DAE_SUPPRESSING)
        Severity(MED)
        INTERVAL(TIMER(24:00))
        ACTIVE
        DEBUG(OFF)
        DATE(20081201)
        REASON('To avoid duplicate dumps')
```

Figure 9-6 Syntax of the DAE availability health check

New messages issued by the DAE availability health check are shown in Figure 9-7 on page 228.

```
ADYH001E DAE is not active in the system
Explanation: CHECK(IBMDAE,DAE_SUPPRESSING) found that DAE cannot suppress dumps
because it is not active in the system.

ADYH002I DAE is active and has the following configuration: settings
Explanation: CHECK(IBMDAE,DAE_SUPPRESSING) found that DAE is active. This
informational message shows current configuration settings.

ADYH003I Neither SUPPRESS nor SUPPRESSALL for dump_type is specified. Duplicate
dumps will not be suppressed. This could negatively impact system performance.
Explanation: CHECK(IBMDAE,DAE_SUPPRESSING) found that although DAE is active,
dumps will not be suppressed because of configuration settings. dump-_type
fillin shows a type of a dump, either SVCDUMP or SYSMDUMP. The message may
appear up to twice with different 'dump_type' value.

ADYH004I SUPPRESSALL for dump_type is not specified. Some duplicate dumps
cannot be suppressed.
Explanation: CHECK(IBMDAE,DAE_SUPPRESSING) found that although DAE is active,
some dumps cannot be suppressed because SUPPRESSALL option is not specified in
the configuration. SUPPRESSALL allows more dumps to be eligible for
suppression. 'dump_type' fillin shows a type of a dump, either SVCDUMP or
SYSMDUMP. The message may appear up to twice with different 'dump_type' value.
```

Figure 9-7 Messages issued by the DAE availability health check

DAE shared data set health check

The DAE shared data set health check raises an exception if the system is in a sysplex environment and DAE is active but the **SHARE(DSN)** option is not specified. The syntax of the DAE availability health check is shown Figure 9-8

```
CHECK(IBMDAE, DAE_SHAREDSN)
        Severity(MED)
        INTERVAL(TIMER(24:00))
        ACTIVE
        DEBUG(OFF)
        DATE(20081201)
        REASON('To check DAE configuration in a sysplex environment')
```

Figure 9-8 Syntax of the DAE shared dataset health check

New messages issued by the DAE availability health check are shown in Figure 9-9 on page 229.

```
ADYH011E This system is in a sysplex but SHARE(DSN) is not in effect.
Explanation: CHECK(IBMDAE,DAE_SHAREDSN) found that this system is in a sysplex
but SHARE(DSN) is not in effect. This option is recommended because it
specifies DAE to suppress a duplicate of a previous dump from any system when
all systems in the sysplex share a common DAE data set.

ADYH012I This system is in a sysplex but DAE is not active so the SHARE(DSN)
option cannot be checked.
Explanation: CHECK(IBMDAE,DAE_SHAREDSN) found that DAE is not active in the
system, so the SHARE(DSN) option cannot be checked. When DAE is running, the
option will be checked again.

ADYH013I This system is in a sysplex, DAE is running and its SHARE(DSN) option
is set as recommended.
Explanation: CHECK(IBMDAE,DAE_SHAREDSN) found that SHARE(DSN) option is
specified in the ADYSETxx member used to start DAE so the DAE data set is
shared as IBM recommends.
```

Figure 9-9 Messages issued by the DAE shared dataset health check

9.6 Service aids support of extended address volumes

The first stage of extended address volume (EAV) support was shipped in z/OS V1R10. This support allowed most VSAM objects to reside in cylinder-managed space on extended access volumes. The second stage of EAV support in z/OS V1R11 adds the ability to place extended format sequential data sets in cylinder-managed space. This support is largely implemented by DFSMS.

The goal of service aids support is to provide continuity of support, allowing extended format sequential data sets to be supported in both track-managed and cylinder-managed space to the extent that track-managed data sets have been supported previously.

With z/OS V1R11, the following service aids components support EAV volumes:

- SPZAP
- SDUMP
- SNAP/ABDUMP
- Stand Alone DUMP
- CTRACE
- GTF
- AMATERSE
- IPCS

EAV support for SPZAP

SPZAP fully supports placement of SYSIN and SYSPRINT data sets in cylinder-managed space. SYSLIB references to extended format sequential data sets other than program libraries were not supported in prior releases, and they are not supported by z/OS V1R11 SPZAP.

EAV support for SDUMP

SDUMP fully supports placement of dump data sets in cylinder-managed space. This includes SYSMDUMPs and TDumps.

EAV support for SNAP/ABDUMP

SNAP and ABDUMP fully support the placement of dump data sets in cylinder-managed space.

EAV support for CTRACE/GTF

CTRACE and GTF support placement of NOWRAP traces in cylinder-managed space. Only NOWRAP traces can be recorded to extended format data sets in earlier releases.

CTRACE and GTF EAV1 support allowed both WRAP and NOWRAP traces to be written in cylinder-managed space occupied by VSAM linear data sets. That support remains in place.

AMATERSE

AMATERSE fully supports the placement of dump data sets in cylinder-managed space.

IPCS

IPCS fully supports the placement of dump data sets in cylinder-managed space.

9.7 Removal of IPCS problem management

The following IPCS subcommands have been removed:

- ADDDSN
- ADDPROB
- DELDSN
- DELPROB
- LISTDSN
- LISTPROB
- MODDSN
- MODPROB

The **SETDEF** subcommand and the option 0 panel of the IPCS dialog maintained a PROBLEM (problem-number) default for the subcommands. Support for PROBLEM and NOPROBLEM keywords has been removed. The keywords will continue to be parsed for compatibility but ignored.

In the IPCSPRnn parmlib member, keywords DSD, NODSD, PDR, NOPDR, PROBIDPREFIX, SYSTEM, GROUP, ADMINAUTHORITY, and DELETEAUTHORITY will continue to be parsed successfully but will be ignored. Only LINELENGTH and PAGESIZE keywords will remain functional.

The following groups of messages described in z/OS MVS dump output messages have been deleted:

- BLS03100I-BLS04016I
- BLS04040-BLS04066I
- BLS04068I-BLS06402I
- BLS21062I

10

Migration health checks

Beginning with z/OS V1R10, the IBM Health Checker for z/OS infrastructure is exploited for migration purposes. This chapter explains how to use the new migration health checks to help migrate from z/OS V1R9 and V1R10 to z/OS V1R11 conform to best practices. These checks provide better systems management and availability.

Before migrating to z/OS V1R11, use these new checks to assist with migration planning. After migration, rerun the checks to verify that the migration actions were successfully performed.

This chapter describes the new migration checks introduced with z/OS R1R11:

- STP (one new check)
- UNIX System Services (three new checks)
- Communications Server (two new checks)
- DFSMSrmm (three new checks)

10.1 Using IBM Health Checker for z/OS for migration purposes

As mentioned, migration checks are intended to be used on your current z/OS release and then again after you have migrated to your new z/OS release.The names of migration checks follow the convention *ZOSMIGVvvRrr_component_program_name* or *ICSFMIGnnnn_component_program_name* for ICSF. This convention tells you that the check helps with migration, as well as the release in which the migration action was introduced.

You can follow these steps.

On your current z/OS release

1. Install all migration health checks PTFs.

- Migration checks can be found using the funcional PSP bucket HCHECKER.

  `http://www14.software.ibm.com/webapp/set2/psp/srchBrocker`

- You can view all IBM Health Checker for z/OS checks at:

  `http://www.ibm.com/systems/z/os/zos/hchecker/check_table.html`

  Make sure you have received the APARs listed in Table 10-1.

Table 10-1 APARs number for migration health checks

Migration health check names	APAR numbers	Available releases
ZOSMIGREC_ROOT_FS_SIZE	OA28684 OA28631	z/OS V1R9 and higher
ZOSMIGV1R11_ZFS_*checks*	OA25026	z/OS V1R10 and higher
ZOSMIGV1R11_CS_RFC4301	PK84362 OA28605	z/OS V1R9 and higher
ZOSMIGV1R11_RMM_*checks*	OA26947	z/OS V1R9 and higher

2. Activate the migration health checks appropriate for your migration path, specifying releases you are migrating through and to. There are different ways to make a check active, as well as many ways of using wildcards to include specific checks. For example, you can active them using MODIFY command:

```
F HZSPROC,ACTIVATE,CHECK=(IBM*,*MIGV1*)
F HZSPROC,ACTIVATE,CHECK=(IBM*,ZOSMIGV1R11)
```

3. Migration health checks will run on your current release. Review the migration check output and rerun checks as appropriate. Any exceptions should be addressed in your migration plan.
4. Migrate to z/OS V1R11.

On your new z/OSV1R11

On the system that you are installing, consider the following options:

1. Install any updated Migration Health Check PTFs on your new release. New migration checks might be available for your new z/OS system since you installed it.
2. Activate the migration health checks appropriate for that release, specifying releases you migrated through to current.
3. Migration health checks will run on your z/OS V1R11. Review the migration check output and rerun checks as appropriate. Any exceptions, which could indicate that a migration

action was not performed correctly, should be addressed. Rerun the check after the corrections have been made.

4. Deactivate the migration health checks when you wish. After your migration verification is complete, deactivate the migration checks similar to the way you activated them. For example:

```
F HZSPROC,DEACTIVATE,CHECK=(IBM*,*MIGV1*)
F HZSPROC,DEACTIVATE,CHECK=(IBM*,ZOSMIGV1R11)
```

Note: Not all migration actions are addressed by checks; many migration actions do not lend themselves to programmatic checking.

10.2 New Server Timer Supervisor health check

There is one new IBM TIMER check for the Server Time Protocol.

- Check(ZOSMIGREC_SUP_TIMER_INUSE)

  This check verifies that Server Time Protocol (STP) is in use, when appropriate. Use Server Time Protocol because the Sysplex Timer (9037-002) has been withdrawn from marketing and STP is planned to be its replacement.

  SEVERITY(LOW) - INTERVAL(ONETIME) - INACTIVE

  This check applies to z/OS V1R11 and later.

10.3 New z/OS UNIX System Services health checks

There are three news UNIX System Services checks.

- ZOSMIGREC_ROOT_FS_SIZE check

  The ZOSMIGREC_ROOT_FS_SIZE check examines the volume that the version root file system resides on, and determines whether that volume has an acceptable number of available cylinders (currently a number greater than 500 free cylinders, but overridable by the client). This check will pass if the volume that the version root file system resides on has a number of available cylinders that is greater than the minimum required. This check will fail if the volume that the version root file system resides on has a number of available cylinders that is less than the minimum required.

  This check verifies the size accommodation for the z/OS root file system to prevent halt on installation.

  There is one parameter to this check, which is the number of cylinders available on the volume where the version root file system resides (defaulted at 500 cylinders, the acceptable value is in the range 500-1,000,000).

  SEVERITY(LOW) - INTERVAL(ONETIME) - INACTIVE - PARM:('MIN_CYLINDERS=500')

  This check applies to z/OS V1R9 and later. These system releases require APARs OA28684 and OA28631.

- Check(ZOSMIGV1R11_ZFS_INTERFACELEVEL)

  This check verifies that the system is running sysplex_admin_level=2 for zFS V1R11 toleration support.

All members of the sysplex should be running at ZFS sysplex_admin_level=2. After this is done, zFS V1R11 may be brought into the sysplex.

SEVERITY(LOW) - INTERVAL(ONETIME)

This check applies to z/OS V1R9 and higher with APARs OA25026 and OA27198.

- ZOSMIGREC_ZFS_RM_MULTIFS

  Beginning in z/OS V1R11, you can no longer attach zFS multi-file system aggregates that are shared across systems in a sysplex. Multi-file system aggregates should not be used. Compatibility mode aggregates should be used.

  The ZOSMIGREC_ZFS_RM_MULTIFS check verifies that the system has no multi-file system aggregates attached.

  SEVERITY(LOW) - INTERVAL(ONETIME)

  This check applies to z/OS V1R9 and higher with APARs OA25026 and OA27198.

10.4 New z/OS Communications Server health checks

There are two news checks for the z/OS Communications Server.

- ZOSMIGV1R11_CS_DNSBIND9 check

  The ZOSMIGV1R11_CS_DNSBIND9 check determines whether a BIND9 DNS server is in use on a system. The ability to run a DNS server on z/OS will no longer be supported in future releases.

  SEVERITY(LOW) - INTERVAL(24:00)

  This check applies to z/OS V1R11 and higher.

- ZOSMIGV1R11_CS_RFC4301 check

  This check verifies whether IPSec filter rules that are not in compliance with RFC4301 are in use on a system. IPSec filter rules that are not compliant with RFC4301 will no longer be supported in future releases of z/OS.

  SEVERITY(LOW) - INTERVAL(24:00)

  This check applies to z/OS V1R10 and V1R11. The APARs PK84362 and OA28605 are required.

10.5 New DFSMSrrm health checks

The DFSMSrmm has three new health checks to assist in the migration to z/OS V1R11.

- Check(ZOSMIGV1R11_RMM_DUPLICATE_GDG)

  Some clients have applied a USERMOD that provides functionality which is now covered with DUPLICATE(COUNT). The USERMOD allows you to activate the function separately for CYCLES and BYDAYSCYCLE retention types, unlike GDG(DUPLICATE()), which covers both retention types. However, the USERMOD is no longer supported.

  The ZOSMIGV1R11_RMM_DUPLICATE_GDG check verifies whether the USERMOD is currently applied.

  This check uses AMBLIST and AMASPZAP to verify the content of ZPSWITCH in load module EDGVREC, CSECT EDGVRECM.

  If EDGVREC is not in SYS1.LINKLIB, then specify the LINKLIB() parameter.

SEVERITY(MED) - INTERVAL(ONETIME) - PARM: LINKLIB(SYS1.LINKLIB)

This check applies to z/OS V1R9 and higher with APAR OA26947.

- ZOSMIGV1R11_RMM_REXX_STEM check

  The ZOSMIGV1R11_RMM_REXX_STEM migration health check determines whether installation-written REXX execs that issued RMM TSO subcommands use stem variables which are removed in z/OS V1R11 and later systems.

  For each library member that contains a reference to a dropped stem variable, the variables are listed.

  SEVERITY(MED) - INTERVAL(ONETIME) - PARM: LIBRARY(library_dsname)

  This check applies to z/OS V1R9 and higher with APAR OA26947.

- ZOSMIGV1R11_RMM_VRSEL_OLD check

  This check is used to determine whether an installation is still using OPTION VRSEL(OLD) despite migration actions in earlier z/OS releases.

  DFSMSrmm must be active for this check to be run. The check should be run once for each system to enable each of the parmlib settings to be verified.

  SEVERITY(MED) - INTERVAL(ONETIME)

  This check applies to z/OS V1R9 and higher with APAR OA26947.

11

z/OS BCP enhancements

The backbone of the z/OS system is the MVS Basic Control Program (BCP). This chapter discusses the following BCP enhancements:

- Display **ALLOC** and **SETALLOC** commands
- EDT size reduction
- Tape load balancing
- IEFBR14 to delete without recall
- Demand tape library allocation

11.1 DISPLAY ALLOC and SETALLOC commands

Device allocation is the assignment of input/output devices and volumes to job steps. Requests for device allocation come from data definition (DD) statements and dynamic device allocation requests.

MVS device allocation has been updated to allow many of the defaults and options to be set at IPL using the ALLOCxx parmlib member to be changed dynamically. This avoids an IPL to change the settings. In addition, parameters defined in the ALLOCxx parmlib member may be modified using operator commands.

The following items are described in this topic:

- MVS allocation device and groups locks.
- The **D ALLOC,GRPLOCKS** command is a diagnostic aid.
- The **D ALLOC,OPTIONS** command displays current ALLOCxx parmlib member options.
- The **SETALLOC** command allows for dynamic updates to current allocation settings.

11.1.1 MVS device allocation and group locks

An esoteric device group identifies the I/O devices that are included in that group. The name you assign to an esoteric device group is called the esoteric name. To request allocation of a device from an esoteric device group, specify the esoteric name on the UNIT parameter of a JCL DD statement. The name esoteric device group is often shortened to esoteric group or simply esoteric.

Allocation device group locks are an internal serialization mechanism, used by allocation to serialize devices from other allocations going on in parallel. Allocation groups devices with similar attributes and esoteric requirements, allowing the possibility to serialize the one group rather than many devices.

Figure 11-1 on page 239 shows the following information:

- Devices 180-186 are of the same device type and are defined as being only in the ALLDASD esoteric group.
- In the second row, devices 180-186 are defined as being in two esoterics, DASD1 and DASD2 groups. If there is an allocation request to DASD1, then devices 180-183 are locked when determining the device to allocate to.
- In the third row, all three esoterics are defined. ALLDASD is a superset of DASD1 and DASD2, so still only two groups are defined. For example, if there are two allocation requests, one to DASD2 and then one to ALLDASD, the DASD2 case will serialize the selection of 184-186. The allocation that requests ALLDASD must wait for the other to release the group locking resources because 184-186 are contained within ALLDASD.

The group lock contention is a difficult situation to debug especially in a dump or after a job has been hanging for a time and is cancelled.

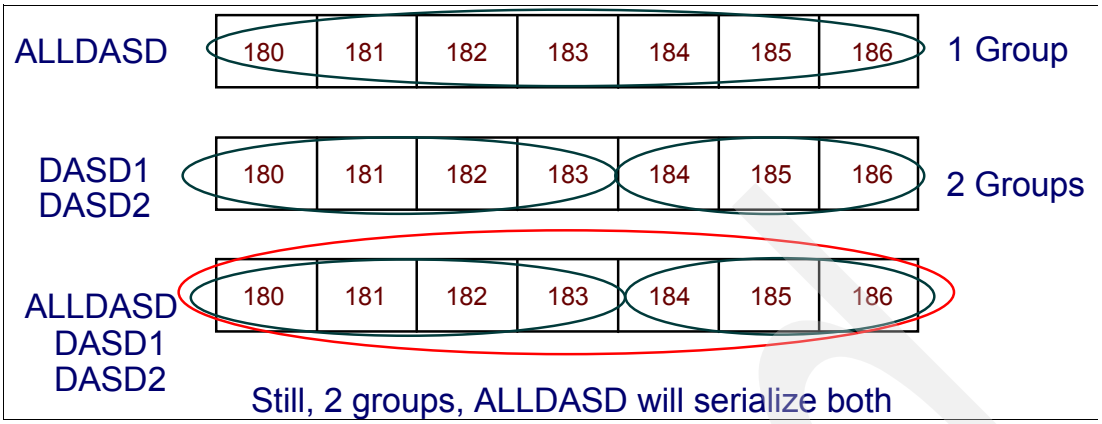


Figure 11-1 Group lock contention

11.1.2 D ALLOC, GRPLOCKS command

Use the **DISPLAY ALLOC,GRPLOCKS** command to display information about the current device allocation group locks that are being held as shown in Figure 11-2. The command is used to diagnose potential causes for a job hang and to make it easier and faster to discover how to fix the problem.

The operator can specify one of the four options:

- Specify all group locks
- Group locks for which contention exists
- Group locks associated with a particular device
- Group locks associated with a particular jobname

Filters the operator can specify with this command include the following:

ALL	Displays all groups in which grouplocks are either being held or waited on
CONTENTION | C	Displays all groups that are in grouplock contention (groups that have both owners and waiters on their grouplocks)
JOBNAME | J=jobname	Displays all devices in which grouplocks are either held or waited on by the particular jobname
DEVICE | D=device	Displays all groups that are either holding or waiting on grouplocks for a particular named device

```
D ALLOC,GRPLOCKS
      { ,ALL }
      { ,CONTENTION | C }
      { ,DEVICE | D = d }
      { ,JOBNAME | J = j }
  [,L={a|name|name-a}]
```

Figure 11-2 D ALLOC,GRPLOCKS command

Note: If the input has an invalid jobname (containing a non-alphanumeric or non-national character) or an invalid device number (one that is not 3 or 4 characters), then message IEFA002I is issued. Simply enter the device number; the forward slash mark (/) is not allowed. Only 1 device or jobname is allowed as input, not a list.

In each of those groups, the group numbers, the devices, the jobnames, the asids, and the status of the jobs are displayed. If no groups match the criteria, then IEFA001I displays the messages shown in Figure 11-3 indicating that there is nothing to display.

```
D ALLOC,GRPLOCKS,ALL
IEFA001I 13.53.21 ALLOC GROUPLOCKS 647
  THERE ARE NO OWNERS/WAITERS FOR GROUP LOCKS.
```

Figure 11-3 D ALLOC,GRPLOCKS,ALL command

D ALLOC,GRPLOCKS,CONTENTION command

Figure 11-4 displays information when CONTENTION is the filter. This command indicates that in all groups that are in device allocation group lock contention (groups that have both owners and waiters on their group locks), the group numbers, the devices, the jobnames, the asids, and the status of the jobs are to be displayed.

```
D ALLOC,GRPLOCKS,CONTENTION
IEFA001I 15.31.25 ALLOC GROUPLOCKS 817
DISPLAYING GROUP LOCKS FOR JOBS IN CONTENTION ONLY.
GROUP 2 HAS THE FOLLOWING DEVICE(S)
00C0-00C5,02E0-02FF,0340-034F,0370-037F,0980-098F
JOBNAME   ASID  STATUS
JOB1       002C  OWNER
JOB2       002D  WAITER
DISPLAYING GROUP LOCKS FOR JOBS IN CONTENTION ONLY.
GROUP 12 HAS THE FOLLOWING DEVICE(S)
0593
JOBNAME   ASID  STATUS
JOB1       002C  OWNER
JOB2       002D  WAITER
```

Figure 11-4 Display group locks contention

D ALLOC,OPTIONS command

This command is used to display the current MVS device allocation settings that are in use, as follows:

- As set by the ALLOCxx parmlib member at IPL
- As modified by the **SETALLOC** command operator
- Using the system defaults if no ALLOCxx parmlib member has been specified or if no **SETALLOC** command has been processed

The IEFA003I message displays the current options, as shown in Figure 11-5 on page 241.

Certain parmlib options are only displayed when they are applicable to the settings that the system is using. For example, if the SPEC_WAIT POLICY is CANCEL or WTOR, then the MAXNWAIT, and POLICYNW are not displayed. Figure 11-5 on page 241 displays the device allocation settings that are in use.

Only allocation options which are applicable to the current settings are allowed to be set. Likewise, all applicable allocation options are required when changing to such a setting that has dependent keywords. For example, when setting the ALLC_OFFLN POLICY to

WAITHOLD, the POLICYNW must also be set in the same command; otherwise, the new value is rejected and IEFA011I is issued.

As another example, when the SPEC_WAIT POLICY is not WAITNOH, then the operator may not change the SPEC_WAIT MAXNWAIT value because it only applies to the WAITNOH POLICY. Change the SPEC_WAIT POLICY to WAITNOH to change the value of MAXNWAIT. After the SPEC_WAIT policy is WAITNOH, then MAXNWAIT can be changed by itself, without having to specify those other options.

```
D ALLOC,OPTIONS
IEFA003I 15.45.46 ALLOC OPTIONS 754
SPACE           PRIMARY:          4
                SECONDARY:        24
                DIRECTORY:        0
                MEASURE:          AVEBLK
                BLKLNGTH:         8192
                ROUND:            NOROUND
                RLSE:             RLSE
UNIT            NAME:             SYSALLDA
                UNITAFF:          SYSTEM DEFAULT
                REDIRECTED_TAPE:  TAPE
TIOT            SIZE:             32 (MAX DDS: 1635)
SDSN_WAIT       WAITALLOC:        NO
VOLUME_ENQ      POLICY:           WTOR
VOLUME_MNT      POLICY:           WTOR
SPEC_WAIT       POLICY:           WTOR
ALLC_OFFLN      POLICY:           WTOR
CATLG_ERR       FAILJOB:          NO
                ERRORMSG:         NO
2DGT_EXPDT      POLICY:           ALLOW
VERIFY_VOL      POLICY:           YES
SYSTEM          IEFBR14_DELMIGDS: LEGACY
                TAPELIB_PREF:     EQUAL
                REMIND_INTV:      90
```

Figure 11-5 D ALLOC,OPTIONS command

SETALLOC command

The **SETALLOC** command is used to dynamically modify device allocation settings. The syntax is borrowed from the ALLOCxx parmlib member syntax, with the exception of using commas (,) and equal (=) signs instead of spaces and parenthesis.

Before using this command, however, see *z/OS MVS Initialization and Tuning Reference*, SA22-7592, for an explanation of all of the acceptable parameters and their functions and possible values.

Message IEFA010I displays each option that was set in the command when all input is valid. The operator may set more than one parameter in the same category, but may not specify the same option twice. If a parameter is specified twice, then the command is rejected and no parameters are set.

Note: The **SETALLOC** command does not support the 2DGT_EXPDT POLICY keyword.

If any of the values specified for an option are not valid, then the command is rejected and IEFA011I is issued.

Figure 11-6 on page 242 shows that the primary space shown in Figure 11-5 has been changed from 4 to 5.

```
SETALLOC SPACE,PRIMARY=5
IEFA010I SETALLOC COMMAND SUCCESSFUL
PRIMARY SET TO 5.

D ALLOC,OPTIONS
 IEFA003I 09.05.45 ALLOC OPTIONS 524
 SPACE           PRIMARY:          5
                 SECONDARY:        24
                 DIRECTORY:        0
                 MEASURE:          AVEBLK
                 BLKLNGTH:         8192
                 ROUND:            NOROUND
                 RLSE:             RLSE
 UNIT            NAME:             SYSALLDA
                 UNITAFF:          SYSTEM DEFAULT
                 REDIRECTED_TAPE:  TAPE
 TIOT            SIZE:             32 (MAX DDS: 1635)
```

*Figure 11-6 Example **SETALLOC** command*

The complete syntax for the **SETALLOC** command is shown in Figure 11-7.

```
SETALLOC {SPACE[,PRIMARY=n]
              [,SECONDARY=n]
    [,DIRECTORY=n]
    [,MEASURE={TRK|CYL|AVEBLK}]
    [,BLKLNGTH=n]
    [,ROUND={ROUND|NOROUND}]
    [,PRIM_ORG={CONTIG|MXIG|ALX}]
    [,RLSE={RLSE|NORLSE}] }
        {UNIT[,NAME=group]
   [,UNITAFF=unit]
   [,REDIRECTED_TAPE={TAPE|DASD}] }
        {TIOT,SIZE=n}
        {SDSN_WAIT,WAITALLOC={YES|NO}}
        {VOLUME_ENQ,POLICY={WTOR|CANCEL|WAIT}}
        {VOLUME_MNT,POLICY={WTOR|CANCEL}}
        {SPEC_WAIT[,POLICY={WTOR|CANCEL|WAITHOLD|WAITNOH}]
        [,MAXNWAIT=n]
        [,POLICYNW={WTOR|CANCEL}] }
        {ALLC_OFFLN[,POLICY={WTOR|CANCEL|WAITHOLD|WAITNOH}]
         [,MAXNWAIT=n]
         [,POLICYNW={WTOR|CANCEL}] }
        {CATLG_ERR[,FAILJOB={YES|NO}]
        [,ERRORMSG={YES|NO}] }
        {VERIFY_VOL,POLICY={YES|NO}}
        {SYSTEM[,IEFBR14_DELMIGDS={LEGACY|NORECALL}]
               [,TAPELIB_PREF={EQUAL|BYDEVICES}]
               [,REMIND_INTV=intv] }
```

Figure 11-7 Parameters of the **SETALLOC** *command*

Note: Settings that are changed using a **SETALLOC** command take effect on the next allocation request (or in the case of TIOT SIZE, the next job started after the command was completed). See *z/OS MVS System Commands,* SA22-7627, for additional information about the **SETALLOC** command parameters.

11.2 EDT size reduction

An installation's device groups are defined in the eligible device table (EDT). The EDT is built during system initialization. An EDT is an installation-defined and -named representation of the devices that are eligible for allocation. This table also defines the relationship of generic device types and esoteric group names. The term “generic device type” refers to the general identifier IBM gives a device, for example, 3380. An esoteric device group is an installation-defined and -named collection of I/O devices; TAPE is an example of an esoteric group name.

The EDT is a collection of tables that describe the I/O configuration from the operating system perspective. This information is built from information in the active IODF and used by MVS device allocation to select and serialize devices used by batch jobs, TSO users, and subsystems. It includes all devices defined, regardless of their state. It also includes the necessary “esoteric device type” information defined by the installation.

Current EDT design

When IODF support was added originally, the EDT was built entirely in common storage. However, at that time, there was a limit of 4096 devices. In addition, the internal “group” processing was limited to 64 K. So even though our control blocks were built to allow any combination of devices, esoterics and “groups”, the storage needed was relatively small.

Over time, as constraints have been lifted (such as the 4 K device limit in MVS SP510, and the 64 K group limit in z/OS V1R9), the storage needed to hold any combination of devices, esoterics, and groups has greatly increased.

In z/OS V1R9, internal groups were increased to 2 G and a much larger amount of CSA/ESQA was required. Common storage is often cited as a constraint to workload growth and consolidation. The changes with z/OS V1R11 are designed to help alleviate constraints by implementing the EDT to consume less common storage.

11.2.1 z/OS V1R11 EDT enhancements

z/OS V1R11 builds the EDT in private storage, where you can obtain the storage needed to contain any combination of devices, esoterics, and groups. Private storage is available earlier in an IPL than when the EDT is first created. The system then obtains only the common storage needed to hold the EDT tables that actually have data. From an external point of view, nothing about the EDT has changed, so no other MVS or vendor products need to change. The impact of this is to realize a great deal of common storage savings to hold the EDT and to build it during an IPL or during **ACTIVATE** command processing.

This increases the amount of common storage to be available for new workloads and consolidation. Installations can even reduce the storage dedicated to common areas using the CSA and SQA parameters in the IEASYSxx parmlib member. This allows more private storage for every address space for installations where private storage is severely constrained.

Note: In previous releases of MVS, the SCHEDxx parmlib member allowed you to define attributes for the EDT through the EDT statement. Now, HCD is used to define attributes for the EDT in the IODF. For more information about defining the EDT, see *z/OS Hardware Configuration Definition Planning*, GA22-7525.

Figure 11-8 shows the command that causes a rebuild of an EDT from an IODF in SYS1.IODF42.

```
ACTIVATE IODF=42,SOFT

IOS500I ACTIVATE COMPLETED - WARNING MESSAGE(S) ISSUED WARN=0209, NEW EDT COULD
NOT BE BUILT.EDT BUILD FAILED, ESQA STORAGE NOT AVAILABLE.

IOS500I ACTIVATE COMPLETED - WARNING MESSAGE(S) ISSUED WARN=0210, NEW EDT COULD
NOT BE BUILT.   EDT BUILD FAILED, PRIVATE STORAGE NOT AVAILABLE.
```

Figure 11-8 Example EDT build failed

11.3 Tape load balancing

When automatic tape libraries were introduced the devices were all similar, containing a limited number of devices. As such, balancing involved simply choosing one of the libraries available. No one library was more capable than another. A limited number of tape devices were supported (16), and selecting a library for a scratch request was easy.

As newer libraries were introduced, however, they contained more tape drives and supported more volumes. As such, users increased their tape library workload to take advantage of the tape libraries available. As a consequence, tape libraries began supporting many more devices, and this increased the tape library workload.

With the newer libraries, today's support does not take into account the number of devices in a library. Non-specific allocations, like scratch requests, for tape are simply randomized equally across the available libraries, then randomly assigned to a tape device within that library. This assumes that the libraries were all above the scratch threshold.

As shown in Figure 11-9 on page 245, the requests are balanced equally across the three libraries, even though library LIBVTS2 has more devices and will handle more workload than the other two.

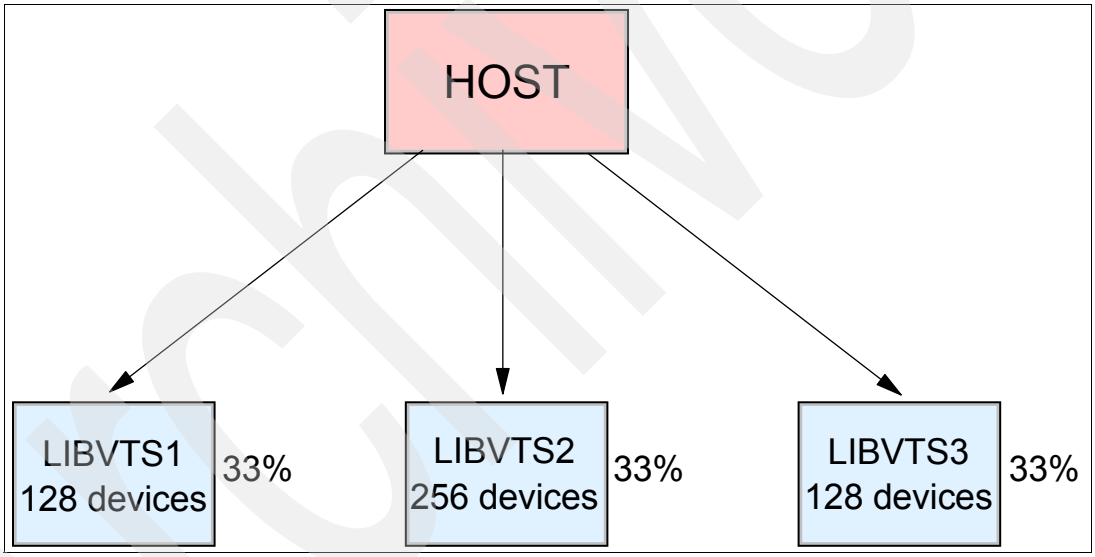


Figure 11-9 Randomized across the libraries

The problem has been seen enough that various users have created elaborate interventions to bypass the problem. However, these solutions have typically been tied to specific workloads and require making error-prone, manual updates. In addition, this problem is not seen until the tape library is already in use by the system.

11.3.1 z/OS V1R11 allocation improvements

In z/OS V1R11, allocation will allocate libraries with more tape drives, taking into account the number of devices in each library. The solution included is to change MVS device allocation to support two tape load balancing algorithms. The new one will randomize across the set of available tape library devices. By removing the step of randomizing across libraries and only randomizing across the set of suitable and available devices, you better balance the workload across the set of devices. As shown in figure 12-8, library LIBVTS2 will now get 50% of the

scratch tape requests because it has twice as many devices to handle that workload than either of the other two libraries.

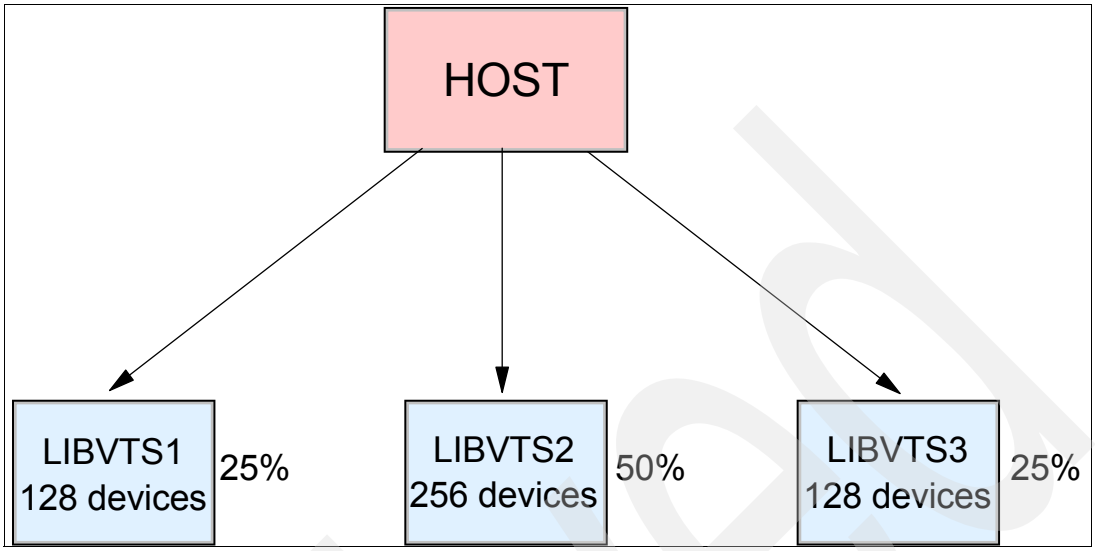


Figure 11-10 Balancing across libraries by number of tape devices

11.3.2 Tape load balancing enhancements

As a prerequisite to exploiting this new function, you set the SYSTEM TAPELIB_PREF keyword in the ALLOCxx parmlib member.

This support is included in APAR OA26414 and PTF UA46842 for V1R11. The ALLOCxx parmlib member supports a TAPELIB_PREF keyword, with values of EQUAL or BYDEVICES.

EQUAL This indicates that for non-specific tape library requests, all tape libraries must be treated as equal and receive an equal share of the requests.

BYDEVICES This indicates that non-specific tape library requests must be balanced across all tape libraries according to the number of tape devices in the tape library. Tape libraries with more tape devices receive more non-specific tape requests than libraries with fewer devices, when all devices have the same attributes.

Note: The support (except for the **SETALLOC** operator command) is rolled back to z/OS releases V1R10 and V1R9 with PTFS UA46841 and UA46840 respectively.

In addition, a new **SETALLOC** command will allow the setting to be changed dynamically, as shown in Figure 11-11.

```
-SETALLOC SYSTEM,TAPELIB_PREF=EQUAL
IEFA010I SETALLOC COMMAND SUCCESSFUL
TAPELIB_PREF SET TO EQUAL.

-D ALLOC,OPTIONS
 IEFA003I 15.27.26 ALLOC OPTIONS 504
SYSTEM          IEFBR14_DELMIGDS: LEGACY
                TAPELIB_PREF:     EQUAL
                REMIND_INTV:      90
```

Figure 11-11 All tape libraries must be treated as equal

11.3.3 Load balancing considerations

For non-specific (scratch) allocations, by default, MVS device allocation will first randomize across the eligible libraries. After a library is selected, it will then randomize on the eligible devices within that library. This works well if the libraries under consideration have an equal number of eligible devices. However, if the libraries have an unequal number of devices, scratch allocations will still be evenly randomized across the libraries.

For example, if two libraries are eligible for a scratch allocation (LIBVTS1 and LIBVTS2) and one of the libraries has 128 devices and the other library has 256 devices, as shown in Figure 11-9 on page 245, then over time each of the libraries will receive half of the scratch allocations before this new support.

MVS device allocation support for the BYDEVICES load balancing option will more evenly distribute the scratch allocations across all of the eligible devices. Thus, in the example shown in Figure 11-10 on page 246, the library with 256 device will receive twice as many scratch allocations.

Before enabling the new load balancing support, care must be taken to ensure that the desired results will be achieved. This is particularly important if the libraries are being shared across multiple systems and the systems can be at various levels of support, or if manual actions had already been taken to account for the existing algorithm's behavior.

Installation considerations

In terms of planning, note that changes to the balancing of scratch requests is likely to change the number of scratch volumes available to handle that change, as follows:

- More tape volumes can be required on tape libraries that have more tape devices, because they will start receiving more tape requests.
- Conversely, tape libraries with fewer tape devices will start receiving fewer scratch tape requests and may not need as many tape volumes defined.

If you choose to use the BYDEVICES function, examine the SMS storage group settings to ensure that the storage groups do not try to manually balance scratch workload. Or, you can always use TAPELIB_PREF(EQUAL) until ready.

11.4 IEFBR14 to delete without recall

With z/OS, there is no shortage of ways to delete data sets. ISPF panel actions, a TSO command, batch JCL, and JCL utilities such as IDCAMS each allow users to delete data sets.

Of these possibilities, using batch JCL with PGM=IEFBR14 is one of the simplest ways to delete data sets. In fact, it has several advantages. The JCL statements are very short and easy to code; they are easy to repeat and update; and they are easy to create using scripts and REXX execs.

However, a main disadvantage of this method is that the allocation component, which runs before the IEFBR14 program gets control, has no knowledge of what the program is going to do. Allocation always gathers all requested resources before starting the batch program, even a program like IEFBR14, which is merely coded to drive either allocation or unallocation.

Most of the time, recalling data sets that have been migrated is the right thing to do. For batch jobs that initiate programs that act on the DDs coded in the JCL, it ensures that the program will run without delay. However, IEFBR14 is unique because it never uses the data sets coded in the DD statements. In short, if the data set is being deleted using IEFBR14, then recalling the data set first wastes time and resources, such as CPU and library or tape operator resources.

z/OS V1R11 IEFBR14 implementation

The solution is to allow jobsteps with IEFBR14 to bypass the recall for existing data sets being deleted and to simply delete the data set. Because this is a change in behavior, perform it under installation control. The benefit is an elapsed time reduction for IEFBR14 jobs that are initiated to delete migrated data sets. It also reduces HSM and tape workload, because the DELETE does not require that the tapes be mounted to delete the data set. Finally, job throughput is increased because the overhead of the recall is eliminated.

This support is invoked by batch JCL that includes an IEFBR14 program, if the following conditions are met:

- The JCL includes DDs for DASD data sets with DISP=(OLD,DELETE) or DISP=(MOD,DELETE).
- The data set is migrated (and note that it can be any type: SMS or non-SMS, VSAM or non-VSAM).
- No VOL=REF or UNIT affinities, which require the data set to exist on DASD, are coded.
- The installation has updated the ALLOCxx parmlib member or executed a **SETALLOC** command that enables the NORECALL behavior.

Note: The support is not called directly; only JCL invocation of the step is supported.

11.4.1 Deleting migrated data sets

When a migrated data set is deleted without recall, IEF285I will indicate that the data set has been HDELETED.

Failure messages (IEF233I, IEF237I) describing actions that cannot be completed will continue to indicate NOT DELETED because the data set remains exactly where it was.

Parmlib definition for delete without recall

As a prerequisite to exploiting this new function, you set the SYSTEM IEFBR14_DELMIGDS keyword in the ALLOCxx parmlib member. This specifies the policy on whether to recall a migrated data set when you use an IEFBR14 JCL program with DD DISP=(x,DELETE) to delete the data set. In most cases the recall is unnecessary because the data set is being deleted anyway.

The ALLOCxx parmlib member supports a IEFBR14_DELMIGDS keyword, with values of LEGACY or NORECALL.

LEGACY This indicates that the system is to recall HSM-migrated data sets before deletion. For minimal impact, LEGACY is the shipped default.

NORECALL This indicates that the system can delete (through HSM HDELETE processing) the data set without first recalling the data set to the primary storage.

Command to update parmlib member

The new **SETALLOC** command will allow the setting to be changed dynamically, as shown in Figure 11-12.

```
SETALLOC SYSTEM,IEFBR14_DELMIGDS=NORECALL

IEFA010I SETALLOC COMMAND SUCCESSFUL
 IEFBR14_DELMIGDS SET TO NORECALL.
```

Figure 11-12 Changed to delete without recall

12

SDSF enhancements

This chapter discusses the enhancements to SDSF for JES3 and JES2 introduced in z/OS V1R11.

The following topics are described:

- Extended support for the JES3 environment
- Elimination of the HASPINDX data set
- zIIP processors support
- Authorization messages in the user session log
- Health Checker enhancements
- Unconditional confirmation messages
- SDSF/REXX enhancements

12.1 Expanded support for the JES3 environment

SDSF support for the JES3 environment was introduced in z/OS V1R10. The initial SDSF support for JES3 only provided data for JES3 jobs in the active users (DA), input queue (I) and status of jobs (ST) panels. Other SDSF panels not dependent on the type of JES, such as the scheduling environments (SE) and WLM resources (RES) panels, were also supported under JES3.

In z/OS V1R11, SDSF expands its support for JES3. The job class, spool volumes, SYSLOG, multi-access spool and output descriptor panels are now supported by SDSF in a JES3 environment. Figure 12-1 shows the new SDSF primary option menu, for a user with full authority, when running in a JES3 environment.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
HQX7760 -----------------  SDSF PRIMARY OPTION MENU  -------------------------
COMMAND INPUT ===>                                          SCROLL ===> CSR

DA    Active users                      SP    Spool volumes
I     Input queue
ST    Status of jobs                    CK    Health checker

LOG   System log                        ULOG  User session log
SR    System requests
JP    Members in the JESPlex
JC    Job classes
SE    Scheduling environments
RES   WLM resources
ENC   Enclaves
PS    Processes

END   Exit SDSF
```

Figure 12-1 SDSF primary option menu under JES3

12.1.1 Supported JES3 releases

Starting with z/OS V1R9, each release of SDSF supports only a single release of JES. SDSF on z/OS V1R11 only supports JES3 V1R11. SDSF V1R11 can be invoked either on a JES3 global or local. If the JES3 processor is not at the required level, SDSF proceeds with initialization, but not all JES3 functions are available.

Furthermore, when SDSF V1R11 is invoked on a JES3 V1R11 local, but the JES3 global is still at V1R10, the new SDSF V1R11 JES3 functions are not available. This includes SYSLOG support, output descriptors, and so on. If the global is still running at a release level lower than V1R10, SDSF proceeds initialization, but no JES3 functions are available at all.

12.1.2 SDSF panels in support of JES3

The following SDSF panels have been updated in z/OS V1R11 to support the JES3 environment:

- System log (LOG)
- Job classes (JC)

- Multi-access spool (MAS), also known as JESPLEX (JP)
- Spool volumes (SP)
- Output descriptors

The LOG panel

The **LOG** command is used to access the SDSF SYSLOG and OPERLOG display. The full syntax of the **LOG** command is shown in Figure 12-2.

```
LOG [OPER|O | SYSLOG| S]
```

Figure 12-2 Syntax of the SDSF LOG command

LOG with no parameters displays the default log panel. Use the **OPER** or **O** parameter to display the OPERLOG panel. Use the **SYSLOG** or **S** parameter to display the SYSLOG panel. When no parameter is specified, the OPERLOG panel is displayed if the OPERLOG is active. Otherwise, the SYSLOG panel is displayed.

On the SYSLOG panel, the **SYSID** command can be used to display the SYSLOG from a specific system in the sysplex. The **SYSID** command is updated to support a system name with 1 to 8 characters, as supported by JES3. In JES2, the system name remains 1 to 4 characters long. Use the **SYSID *** command to display the SYSLOG of the JES3 global.

For the JES3 DLOG to be displayed, it must be active in JES3. Issue the ***I O DLOG** command to check whether the JES3 DLOG is active. The ***F O DLOG=ON** and ***F O DLOG=OFF** commands are used to activate and deactivate the JES3 DLOG. It can be useful to change the SDSF default to always display the SYSLOG panel instead of the OPERLOG panel. In conjunction with the **SYSID *** command, this will cause the JES3 DLOG to be displayed.

The JC panel

The job class (JC) panel allows authorized users to display and control job classes in a JES2 MAS. The panel is enhanced to display and control job classes in a JES3 JESPLEX. Both JES- and WLM-managed classes are supported. Figure 12-3 on page 254 shows an example of displaying JES3 classes in SDSF.

The JC panel is updated to display class names of 1 to 8 characters, as supported by JES3. Other JES3 job class information such as group name, TDEPTH, setup depth, primary allocation tracks, secondary allocation tracks and priority is also available.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF JOB CLASS DISPLAY ALL CLASSES                    LINE 53-78 (78)
COMMAND INPUT ===>                                       SCROLL ===> CSR
ACTION=//,=,+,D,DC,DG,ST
NP   CLASS    Status   Member   Group    Mode Xeq-Cnt  TDepth Log   Jrnl Rst Je
     A        ACTIVE   SC75     A        JES         1   NONE STD   NO   NO  NO
     B        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     C        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     D        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     E        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     F        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     G        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     H        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     I        ACTIVE   SC75     A        JES             NONE STD   NO   NO  NO
     J        ACTIVE   SC75     J        JES             NONE STD   NO   NO  NO
     K        ACTIVE   SC75     J        JES             NONE STD   NO   NO  NO
     L        ACTIVE   SC75     J        JES             NONE STD   NO   NO  NO
     M        ACTIVE   SC75     J        JES             NONE STD   NO   NO  NO
```

Figure 12-3 SDSF JC panel under JES3

The available action characters in the JC panel under JES3 are:

D Display information about a job class in the logs and ULOG.

DC Display status for the class in the logs and ULOG.

DG Display status for the group in the logs and ULOG.

ST Display the ST panel for all jobs in the class.

In addition, the **JC** command accepts a parameter of a single class name from 1 to 6 characters. When a class name parameter is specified on the **JC** command, it is regarded as a class name filter for the JC panel. For example, to display the JC panel with a class name filter of FASTN, enter a **JCFASTN** command. To filter additional classes or class names longer than 6 characters, use the **FILTER** command in the JC panel.

The MAS and JP panels

The **JP** command is a new SDSF command used under JES3 to display the members of the JESPLEX. The **MAS** command is used in SDSF under JES2 to display the members of a JES2 multi-access spool environment. When running in a JES3 environment, the **MAS** command is a synonym of the **JP** command. Figure 12-4 shows an example of the JP panel under JES3.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF JP DISPLAY  SC75               77% SPOOL         LINE 1-3 (3)
COMMAND INPUT ===>                                       SCROLL ===> CSR
ACTION=//,=,+,C,D,DL,F,JS,P,S,SM,V,VF,ZM
NP   NAME     Status       SysName  Version  C        JESN SLevel Global Start-
     SC74     DOWN         SC74              *                0 NO     LOCAL
     SC70     DOWN         SC70              *                0 NO     LOCAL
     SC75     ATTACHED     SC75     z 1.11.0 *        JES3      2 YES    HOT
```

Figure 12-4 SDSF JP panel under JES3

The available action characters in the JP panel under JES3 are:

C Connect a member.

D Display a member of the JESPLEX in the log.

DL Display a member of the JESPLEX in the log, long form.

F Flush jobs currently running on the main.

JS Display the current status of JES3.

P Stop a member of the JESPLEX.

S Start a member of the JESPLEX.

SM Start the JES3 monitor.

SX Start scheduling jobs for the member.

V Start scheduling jobs for the member.

VF Stop scheduling jobs for the member.

ZM Stop the JES3 monitor.

The SP panel

The spool volumes (SP) panel is updated to support JES3. Although the same column names from the JES2 environment are used the JES3 environment, additional columns are provided for data specific to JES3. For example, a spool data set name column is available under JES3 but not under JES2. Figure 12-5 shows an example of the SP panel.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF SPOOL DISPLAY SC75      77% ACT  15000 FRE   3396 LINE 1-4 (4)
COMMAND INPUT ===>                                        SCROLL ===> CSR
ACTION=//,=,+,A,D,DL,H,HC,HP,J,P,U
NP   NAME     Status       TGPct TGNum TGUse Ext LoCyl    LoTrk            LoHe
     DRAINED  INACTIVE       100     0     0 00  00000000 0000000000000000 0000
     UNAVAIL  INACTIVE       100     0     0 00  00000000 0000000000000000 0000
     JES3PART ACTIVE          77 15000 11604 00  00000000 0000000000000000 0000
     SPOOL1   ACTIVE          77 15000 11604 02  00000014 000000000000012C 0000
```

Figure 12-5 SDSF SP panel under JES3

The available action characters in the SP panel under JES3 are:

A Release the spool data set and all jobs that have data on spool for scheduling.

D Display the status of a spool volume.

DL Display the long form of status.

H Hold the spool data set and further scheduling for jobs with data on the data set.

HC Hold the spool data set and cancel all jobs using it.

HP Hold the spool data set and hold further scheduling of jobs with data on it. Cancel jobs active on the main and using the data set.

J Display all jobs using the spool volume.

P Drain a spool volume.

U Resume allocating space on the spool data set.

Under JES3, the SP panel displays both spool volumes and spool partitions. For example, Figure 12-6 shows a spool volume named SPOOL1 and the three default JES3 spool partitions, named DRAINED, UNAVAIL, and JES3PART. The Type column on the SP panel is used to differentiate between spool volumes and spool partitions. Moreover, for each spool volume, the PartName column indicates the spool partition it is contained in. Figure 12-6 shows an example of the Type and PartName columns.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF SPOOL DISPLAY SC75      77% ACT  15000 FRE   3396 LINE 1-4 (4)
COMMAND INPUT ===>                                        SCROLL ===> CSR
ACTION=//,=,+,A,D,DL,H,HC,HP,J,P,U
NP   NAME     rTG Type      PartName OverFNam OverAllow OverOccur OverInto PTra
     DRAINED    0 PARTITION DRAINED           NO        NO        NO
     UNAVAIL    0 PARTITION UNAVAIL           NO        NO        NO
     JES3PART   0 PARTITION JES3PART          NO        NO        NO          1
     SPOOL1     1 EXTENT    JES3PART                                         1
```

Figure 12-6 Type and PartName columns on the SP panel

The output descriptors panel

The output descriptors (OD) panel is not directly accessible from the SDSF primary option menu. The OD panel is accessible using the **O** action character under the DA, I, ST, O, H, and JDS panels. Modification of output descriptors in a JES2 environment must be done at the job output group level. Therefore, when SDSF is running in a JES2 environment, the OD panel allows modification of output descriptors only when the OD panel is accessed from the O and H panels, or when invoking the JDS panel through the O and H panels.

However, in a JES3 environment, modification of output descriptors is done at the spool data set level. Therefore, under JES3, SDSF allows modification of output descriptor when the OD panel is accessed from the DA, I and ST panels, or when invoking the JDS panel through the DA, I, and ST panels. Note that modification of output descriptors is only allowed for spool data sets not actively in use by JES3. Figure 12-7 shows an example of the OD panel, accessed through the DA panel. The fields shown in bold are modifiable.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF OUTPUT DESCRIPTORS - JOB PELEGENR (JOB14667)      LINE 331-366 (535)
COMMAND INPUT ===>                                        SCROLL ===> CSR
NP  DDNAME   Output Descriptors
    SYSPRINT Building
             008 - 2ND FLOOR

    SYSPRINT Department
             ITSO

    SYSPRINT Room
             C3

    SYSPRINT Address
             2455 SOUTH ROAD
             POUGHKEEPSIE, NY 12601
```

Figure 12-7 SDSF OD panel under JES3

12.2 Elimination of the HASPINDX data set

Prior to z/OS V1R11, the SDSF SYSLOG function is depended on a single, sysplex-scope, HASPINDX data set to access SYSLOG data sets across the MAS. The HASPINDX data set was maintained by SDSF running in various TSO/E address spaces as users browse the JES SYSLOG data set, which caused a number of issues.

Serialization and data corruption is one issue. Every user of the HASPINDX data set must serialize their updates, and data corruption can occur if the user abends while accessing the data set. A second problem with the HASPINDX data set is performance. The first user to access the SYSLOG after a long period of inactivity, for example the first user who uses SDSF in the morning, can experience a significant delay waiting for the HASPINDX data set to be updated.

Starting with z/OS V1R11, JES2 maintains its own index of the SYSLOG spool data set. Using job and spool data set indexing, JES2 in z/OS V1R11 provides support for viewing the SYSLOG as a single logical data set and allows rapid access to SYSLOG records by record number and time. SDSF in z/OS V1R11 exploits the new JES2 logical SYSLOG support and eliminates the use of the HASPINDX data set, along with its well-known problems.

Moreover, by using the new JES2 support for logical SYSLOG, SDSF removes the requirement to set up WebSphere MQ in order to gather cross-sysplex SYSLOG information. WebSphere MQ is still needed to provide cross-sysplex information for SDSF device and resource panels. In a JES3 environment, the HASPINDX data set and WebSphere MQ are not used.

Authorization changes for the logical SYSLOG

A new SAF resource controls access to the JES logical log. The resource name is nodeid.+MASTER+.SYSLOG.SYSTEM.sysid in the JESSPOOL class. READ access is required. This resource is in addition to the ISFCMD.ODSP.SYSLOG resource in the SDSF class that is already used.

To simplify migration to the new logical SYSLOG, a new custom property Security.Syslog.UseSAFRecvr may be set to TRUE, to cause the SAF check for the logical SYSLOG to always pass. An example of using the new property is provided in Figure 12-8.

```
PROPLIST NAME(LOGMIGR)
    PROPERTY NAME(Security.Syslog.UserSAFRecvr) VALUE(TRUE)
```

Figure 12-8 SDSF custom property for logical SYSLOG migration

Formatting changes for the logical SYSLOG

The SYSLOG records are displayed in a slightly differing format when SDSF is using the JES logical SYSLOG support. This can affect SDSF batch scripts relaying on the old format. See *SDSF Operation and Customization,* SA22-7670, for an explanation of the exact changes required for SDSF batch processing.

Coexistence with earlier releases of SDSF and JES2

When SDSF is using the JES2 logical SYSLOG support, the ISFPARMS settings related to HASPINDX, NIDBUF, INDEX and INDXVOL are ignored. They are retained in ISFPARMS if it is shared with systems running SDSF releases lower than V1R11. When a user on a V1R11 SDSF system issues the **SYSID** command to view the SYSLOG of a lower level system, SDSF still uses the HASPINDX data set. Only when all systems in the sysplex are running at z/OS V1R11, the HASPINDX data set it not required any more.

12.3 zIIP support

In z/OS V1R11, the SDSF DA panel now shows zIIP usage and an LPAR view of CPU usage for each system, as follows:

zIIP-Time Accumulated zIIP service time, in seconds.

zICP-Time Accumulated general processor service time that was eligible for a zIIP, in seconds.

zIIP-NTime Normalized zIIP service time, in seconds.

SLCPU% Percentage of time the LPAR is busy for the system, in the most recent interval. The value for SLCPU% is the same for all rows for a system.

Note: RMF monitor I must be active for SDSF to display data in the DA panel columns specified.

12.4 Health Checker panel enhancements

Several enhancements to the SDSF Health Checker (CK) panel are available in z/OS V1R11.

Overtypeable check parameters field

The **Parameters** field is now modifiable, making it easier to change check parameters. After a check's parameters are update, subsequent invocations of the check use the updated parameters. Figure 12-9 shows an example of the CK panel. The fields in bold are modifiable.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HEALTH CHECKER DISPLAY  SC75                     LINE 1-37 (139)
COMMAND INPUT ===>                                       SCROLL ===> CSR
NP   NAME                              ParmLen Parameters                     S
     ASM_LOCAL_SLOT_USAGE                   14 ('THRESHOLD(30%)')             z
     ASM_NUMBER_LOCAL_DATASETS              12 ('MINLOCALS(3)')               z
     ASM_PAGE_ADD                           10 ('MINADDS(2)')                 z
     ASM_PLPA_COMMON_SIZE                   15 ('THRESHOLD(100%)')            z
     ASM_PLPA_COMMON_USAGE                  14 ('THRESHOLD(80%)')             z
```

Figure 12-9 Modifiable parameters in the CK panel

RexxIn and RexxOut fields

In addition, two new columns are added to the CK panel, RexxIn and RexxOut. The fields are useful for Health Checker checks that invoke REXX execs using System REXX. RexxIn is equivalent to the REXXINDSN= parameter of the AXREXX assembler macro and RexxOut is equivalent to the REXXOUTDSN= parameter of the AXREXX assembler macro.

The REXXINDSN= parameter is used to specify an optional input parameter containing the name of the data set that the PARSE external function will read data from. The exec can obtain the DDNAME associated with this data set by accessing the AXRINDD variable. The default is NO_REXXINDSN.

The REXXOUTDSN= parameter is used to specify an optional input parameter containing the name of the data set that the exec will direct the output from SAY, error messages and tracing

to. The REXX exec can obtain the DDNAME associated with this data set by accessing the AXROUTDD variable. The default is NO_REXXOUTDSN.

New action characters

As shown in Figure 12-10, the following new action characters are added to the CK panel:

SBI Browse REXX input data set using ISPF browse

SBO Browse REXX output data set using ISPF browse

SEI Edit REXX input data set using ISPF Edit

SEO Edit REXX output data set using ISPF Edit

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HEALTH CHECKER DISPLAY  SC65                      LINE 1-16 (134)
COMMAND INPUT ===>                                       SCROLL ===> HALF
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=
ACTION=//-Block,=-Repeat,+-Extend,A-Activate,D-Display,DD-DisplayDiag,
ACTION=DL-DisplayLong,DP-DisplayPolicies,DPO-DisplayOutdatedPolicies,
ACTION=DS-DisplayStatus,E-Refresh,H-Deactivate,P-Delete,PF-DeleteForce,R-Run,
ACTION=S-Browse,SBI-ISPFBrowseIn,SBO-ISPFBrowseOut,SB-ISPFBrowse,
ACTION=SEI-ISPFEditIn,SEO-ISPFEditOut,SE-ISPFEdit,U-RemoveCat,X-Print,
ACTION=XC-PrintClose,XD-PrintDS,XDC-PrintDSClose,XF-PrintFile,
ACTION=XFC-PrintFileClose,XS-PrintSysout,XSC-PrintSysoutClose
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
```

Figure 12-10 New action characters in the SDSF CF panel

12.5 Authorization messages in ULOG

Starting with z/OS V1R11, SDSF includes SAF authorization messages in the user session log (ULOG), making it easier to identify authorization problems. An example is shown in Figure 12-11 on page 260. The authorization messages are included in the ULOG even when a console is not active, as seen in the example.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
 SDSF ULOG  CONSOLE PELEG                          LINE 0       COLUMNS 42- 121
 COMMAND INPUT ===>                                          SCROLL ===> CSR
******************************** TOP OF DATA *********************************
   ICH408I USER(PELEG   ) GROUP(SYS1    ) NAME(GIL PELEG           )
     ISFCMD.DSP.ACTIVE.JES2 CL(SDSF    )
     INSUFFICIENT ACCESS AUTHORITY
     FROM ISFCMD.DSP.ACTIVE.* (G)
     ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
   ISF031I CONSOLE PELEG ACTIVATED
******************************** BOTTOM OF DATA ******************************
```

Figure 12-11 SAF authorization messages in SDSF ULOG

When using SDSF/REXX, the authorization messages are included in the **isfulog** stem variables.

12.6 Unconditional confirmation function

By default, SDSF displays a confirmation panel for destructive action characters, such as the cancel action. The **SET CONFIRM OFF** command is used to disable the confirmation panel. When the SDSF confirmation function is disabled, a user might accidently perform a destructive action without warning. The status of the SDSF confirmation function can be checked from any SDSF panel using the **SET CONFIRM ?** command.

Starting with z/OS V1R11, the ability to disable the confirmation function in SDSF can be prevented through a setting of a new SDSF configuration parameter in ISFPRMxx or ISFPARMS.

The ISFPRMxx setting is shown in Figure 12-12.

```
GROUP CONFIRM(ON | OFF | ALWAYS)
```

Figure 12-12 Unconditional confirmation setting in ISFPRMxx

The ISFPARMS setting is shown in Figure 12-13.

```
ISFGRP CONFIRM=ON | OFF | ALWAYS
```

Figure 12-13 Unconditional confirmation setting in ISFPARMS

When CONFIRM=ALWAYS is in effect, the **SET CONFIRM OFF** command fails with a PARM NOT ACCEPTABLE error.

Important: Use ISFPRMxx for setting SDSF parameters. A migration tool, ISFACP, is provided in ISF.SISFEXEC, and it automatically converts ISFPARMS syntax to ISFPRMxx syntax.

Coexistence with previous releases

Toleration APAR PK66761 is provided to allow SDSF running on z/OS V1R9 and V1R10 systems to share ISFPRMxx with z/OS V1R11 systems. With the APAR installed, z/OS V1R9 and V1R10 systems ignore the new parameter in ISFPRMxx.

12.7 SDSF REXX enhancements

SDSF provides access to SDSF functions through the REXX programming language. SDSF REXX was introduced in z/OS V1R9. Using SDSF REXX provides a simpler and more powerful alternative to using SDSF in batch. In this section we describe the enhancements to SDSF REXX in z/OS V1R11.

You invoke the SDSF function with a new host command environment, SDSF, new **ISFEXEC** and **ISFACT** commands, and REXX variables. Data and SDSF messages are returned in REXX variables.

You must be authorized to use SDSF from REXX, and to the SDSF functions that you invoke from REXX. Depending on how your SDSF security is implemented, when you use SDSF from REXX you can be placed in an SDSF user group that is other than the user group you are in when you use SDSF interactively. Invoking an SDSF function from REXX when you are not authorized to the function can cause the exec to fail and the invocation of SDSF to end.

12.7.1 ISFACT enhanced syntax

Use the **ISFACT** command to invoke SDSF action characters and modify column values in panels. Use the **TOKEN** parameter to identify the rows to act upon. A token consists of a variable-length string that may contain special characters. You must not modify it. When you call the **ISFEXEC**, **ISFACT** and **ISFGET** commands, SDSF automatically sets tokens to identify the returned rows. These tokens are then used in following calls to **ISFACT**, and so on. Figure 12-14 shows the two forms of passing a tokens to **ISFACT** in z/OS V1R11.

```
address SDSF "ISFACT command TOKEN((stem-name)) PARM(parms)"
address SDSF "ISFACT command TOKEN(token-list) PARM(parms)"
```

Figure 12-14 ISFACT syntax

Using a stem name to act on multiple rows

The example shown in Figure 12-15 on page 262 illustrates how to use the stem name form of **ISFACT** to act upon all rows in a panel. In the example, first **ISFEXEC** is issued to set tokens for all jobs with prefix MYJOBS* in the ST panel. Then, **ISFACT** is used to issue the purge (P) action character against all returned jobs. This is a quick and simple alternative to iterating over all tokens and purging them one by one.

```
/* REXX */

rc = isfcalls('ON')

isfprefix = 'MYJOBS*'

address SDSF
"ISFEXEC ST"
"ISFACT ST TOKEN((token.)) PARM(NP P)"

rc = isfcalls('OFF')
```

Figure 12-15 Using a stem name to act on multiple rows

Using a token list to act on multiple rows

The example in Figure 12-16 shows how to use the token list form of **ISFACT** to act upon specific returned rows. In the example, only rows 1, 3, and 6 are purged.

```
/* REXX */

rc = isfcalls('ON')

isfprefix = 'MYJOBS*'

address SDSF
"ISFEXEC ST"
"ISFACT ST TOKEN('"token.1"','"token.3"', '"token.6"') PARM(NP P)"

rc = isfcalls('OFF')
```

Figure 12-16 Using a token list to act on multiple rows

Using a token list is useful when it is required to perform an action for specific rows according to a condition. The token list can by dynamically created, as shown in Figure 12-17 on page 263. In the example, the **ST** command is issued to receive a list of all jobs with prefix MYJOBS*. Then, a token list is built. Only tokens of jobs with a non-zero completion code are added to the token list. In the end, the **P** action character is issued for each row in the token list, to purge the jobs in the list.

```
/* REXX */

rc = isfcalls('ON')

isfprefix = 'MYJOBS*'

address SDSF
"ISFEXEC ST"

/* purge all jobs with non-zero completion code */
purgeTokens = ""
do i = 1 to isfrows
    if RETCODE.i ¬= 'CC 0000' then
    do
        purgeTokens = purgeTokens "'"TOKEN.i"'"
    end
end
"ISFACT ST TOKEN("purgeTokens") PARM(NP P)"

rc = isfcalls('OFF')
```

Figure 12-17 Building a token list according to a condition

12.7.2 The ISFSLASH command

ISFSLASH is a new SDSF/REXX command in z/OS V1R11that is used to issue system commands through SDSF. It is similar to using **ISFEXEC** with the forward slash (/) command. Using **ISFSLASH** offers several benefits:

- You can issue multiple commands in one call to **ISFSLASH**.
- Commands are passed to **ISFSLASH** in a stem variable or a list of commands.
- Responses from all issued commands are returned together in the isfulog stem variable.

Figure 12-18 on page 264 shows how to use **ISFSLASH** to issue two system commands **D A,L** and **D R,R**. The commands are passed to **ISFSLASH** in a stem variable. Command responses are then printed from the ULOG.

```
/* REXX */

rc = isfcalls("ON")

syscmd.0 = 2
syscmd.1 = "D A,L"
syscmd.2 = "D R,R"

address SDSF "ISFSLASH ("syscmd.") (WAIT"

do i = 1 to isfulog.0
    say isfulog.i
end

rc = isfcalls("OFF")
```

Figure 12-18 Using ISFSLASH with a stem variable

The next example, in Figure 12-19, shows how to pass commands to ISFSLASH as a list of commands instead of a stem variable. This time three commands are issued, namely **D A,L**, **D R,R** and ***I S**.

```
/* REXX */

rc = isfcalls("ON")

address SDSF "ISFSLASH 'D A,L','D R,R','*I S' (WAIT"

do i = 1 to isfulog.0
    say isfulog.i
end

rc = isfcalls("OFF")
```

Figure 12-19 Using ISFSLASH with a list of commands

12.7.3 The isfreset() function

The **isfreset()** function is a new function in SDSF/REXX in z/OS V1R11. It is used to un-assign the special SDSF variables and restore them to their original undefined state. **isfreset()** does not require access to SDSF and so no authorization is required to use it. **isfreset()** is not dependent on **isfcalls()** and can be issued at any point in the REXX exec. It is most useful when issued prior to an address SDSF command.

The syntax of the isfreset() function is as follows:

```
rc = isfreset("ALL"|"INPUT"|"OUTPUT"|"INOUT")
```

Figure 12-20 Syntax of the SDSF/REXX isfreset() function

A return code is set in the REXX variable RC. A return code of zero (0) indicates the request completed successfully.

The meaning of the parameters is as follows:

ALL	Drops all special variables except the variables for printing. This is the default.
INPUT or **I**	Drops all input special variables.
OUTPUT or **O**	Drops all output special variables.
INOUT or **IO**	Drops all input/output special variables.

12.7.4 COLSHELP enhancements

The **COLSHELP** command describes the columns in SDSF panels. For each column **COLSHELP** lists the panel it is available on as well as the column name, column title, and description. It is most useful when writing SDSF/REXX execs, because SDSF/REXX requires the column name, rather than the column title. In addition, **COLSHELP** indicates which columns are delayed-access columns (columns for which I/O must be performed to retrieve the data, instead of accessing JES2 control blocks in storage) and whether columns are overtypeable or not. The **COLSHELP** command is only available in SDSF under ISPF.

The following enhancements are made to **COLSHELP** in release 11:

- **COLSHELP** now only lists the columns for the panel it is invoked from. This saves time searching for the required columns.
- A new locate command provides the ability to quickly find columns in the list.
- A new filter command allows filtering by panel, column name or column description.

Figure 12-21 shows the new **COLSHELP** panel. In this example, **COLSHELP** is invoked from the status (ST) panel.

```
                          Columns on SDSF Panels      Row 874 to 882 of 919
 Command ===> ------------------------------------------------------------

 Sort with F5 (panel), F6 (column), F10 (title). Use Filter to filter rows.

_   All panels           Descriptions

 Panel     Column      Title                Delayed?  Overtype?
 ST        JNAME       JOBNAME
 ST        JTYPE       Type
 ST        JNUM        JNum
 ST        JOBID       JobID
 ST        OWNERID     Owner
 ST        JPRIO       Prty                           X
 ST        QUEUE       Queue
 ST        JCLASS      C
 ST        POS         Pos
  F1=Help        F5=SortPnl     F6=SortFld     F7=Backward    F8=Forward
 F10=SortTtle   F11=SortDesc   F12=Cancel
```

Figure 12-21 New COLSHELP panel

12.7.5 REXXHELP command enhancements

The **REXXHELP** command is used to display online help on SDSF REXX from SDSF under ISPF. In addition to examples and usage information, the online help also includes links to descriptions of commands, action characters and overtypeable columns, which is not included in this information. In z/OS V1R11, REXXHELP now also provides an index which makes it easier to find help about specific terms.

```
                          Using REXX with SDSF


Tab to a topic and press F1, or press Enter to view the topics in order.

o   Introduction                                          >> Index <<
o   Programming practices
o   Quick start
o   Add the SDSF host command environment
o   Issue SDSF commands
    - Panel commands (ISFEXEC)
    - Slash (/) commands (ISFSLASH)
    - Other commands (ISFEXEC)
    - Filter commands (special variables)
    - Options commands (special variables)
o   Take actions and modify columns on SDSF panels
o   Browse output and Print output
o   Examples
o   Diagnose errors in a REXX exec
o   Special variables

 F1=Help     F5=Exhelp   F7=Up       F8=Down    F10=Back    F12=Cancel
```

Figure 12-22 Using the REXXHELP command for the index

SDSF REXX Index

The SDSF online help provides examples and usage information. The online help also includes links to descriptions of commands, action characters, and overtypeable columns.

Using the Index shown in Figure 12-22, place the cursor under Index and press the F1 key.

Figure 12-23 on page 267 is then displayed. This panel allows you tab to a topic and press F1 again.

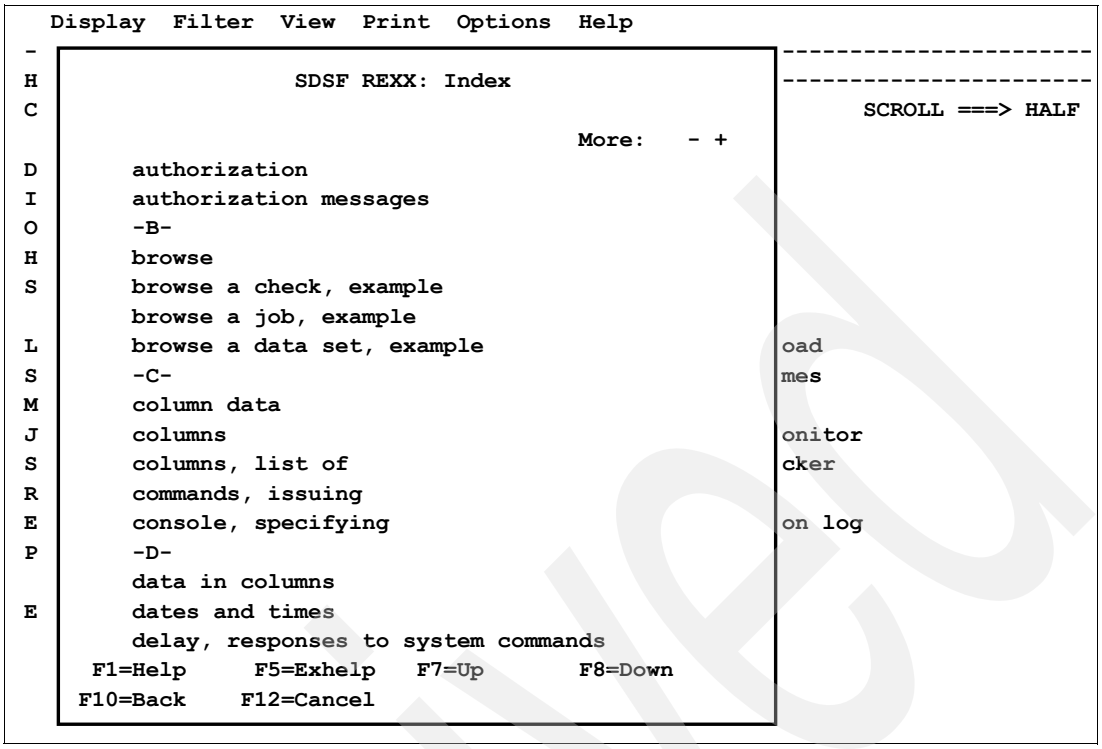


Figure 12-23 SDSF REXX Index

Placing the cursor under the field **browse a job, example**, which is tabbing to a topic, press F1 and Figure 12-24 is displayed.

```
SDSF REXX Example: Browse Job Output

                                                              More:     +
Using ISFEXEC, access the ST panel to create the row variables for jobs.
Then, for each job with a name that matches a desired string (RJONES1),
use ISFACT to issue the SA action character. SA allocates the job data
sets and sets the ISFDDNAME special variable to the DDNAME for each data
set that has been allocated.  Use the ISFDDNAME variable as input on the
EXECIO command and list the data sets line by line.

  /* REXX */

  rc=isfcalls('ON')

       /*************************/
       /* Access the ST display */
       /*************************/
  Address SDSF "ISFEXEC ST"
  lrc=rc
  call msgrtn
  if lrc<>0 then
 F1=Help     F5=Exhelp   F7=Up       F8=Down    F10=Back    F12=Cancel
```

Figure 12-24 SDSF example of how to browse job output using REXX

13

BCP supervisor updates

This chapter discusses new BCP supervisor functions for z/OS V1R11. The following new functions and enhancements are explained:

- zAAP on zIIP support
- Support for greater than 64 CPUs
- SRB priority adjustment
- XM Post SRB without local lock
- Cross memory TCB and SRB wait
- Moving of fields for processor cache improvements
- SPIN system trace entries
- Recovery termination manager (RTM) health check for IEAVTRML

13.1 zAAP on zIIP support

Various clients do not have enough offload-eligible capacity to warrant a “whole number” of both zAAPs and zIIPs. A new function, zAAP on zIIP, has been added to z/OS V1R11 to allow zAAP-eligible work to run on zIIPs. With this support, clients can avoid buying zAAPs if they already have zIIPs and also have the advantage of fewer processor types to manage.

This new function is only available when the machine does not have any zAAPs installed. Even if an LPAR does not have zAAPs defined, this new function is not available if there are zAAPs in the machine.

Restriction: When Java requests to switch to zAAP and there are no zAAPs installed on the entire machine, the requests are treated as a request to switch to a zIIP.

IEASYSxx parmlib member

This support is invoked implicitly when work requests to switch to zAAP. Applications that call the zAAP switch service include Java and the XML parser.

The zAAP on zIIP function is managed through a new IEASYSxx parameter ZAAPZIIP (or ZZ, as its alias). The ZAAPZIIP determines whether the system can run zAAP processor-eligible work on zIIP processors when no zAAP processors are installed on the machine. The IEASYSxx parmlib member specification is:

```
ZAAPZIIP=NO|YES
```

Where:

ZAAPZIIP=YES The system can run zAAP processor-eligible work on zIIP processors when no zAAP processors are installed on the machine.

Default Value: YES

ZAAPZIIP=NO The system cannot run the zAAP processor-eligible work on zIIP processors when no zAAP processors are installed on the system.

Note: To correct an error you make in specifying ZZ, you must specify the value on ZAAPZIIP instead of ZZ. The system issues message IEA341A to indicate this situation.

Coexistence support

This support is available for z/OS V1R9 and z/OS V1R10 through APAR OA27495. For z/OS V1R9 and z/OS V1R10 you must specify ZZ=YES in IEASYSxx to enable this support. The ZAAPZIP parameter in only available on z/OS V1R11.

- The default on z/OSV1R9 and z/OS V1R10 is ZAAPZIIP=NO or ZZ=NO.

Note: The enablement state of zAAP on zIIP cannot be changed after IPL.

SMF records

Timing fields within SMF will show offload time under zIIP, not partly under zIIP and partly under zAAP. There is no mechanism to determine what portion of zIIP time originated due to a zAAP request.

Attention: If you have SMF-based procedures that examine zAAP time and want to use zAAP on zIIP, then these procedures will have to be changed.

13.2 Support for greater than 64 CPUs

As new machines are released, more CPUs are typically supported at the hardware level. For every change in hardware, the software must be enhanced to keep up. Historically, as shown in Figure 13-1, z/OS releases have supported from less that 16 CPUs to up to 64 CPUs:

- Prior to z/OS V1R6, z/OS supported up to 16 CPUs
- z/OS V1R6 to z/OS V1R8 supported up to 32 CPUs
- z/OS V1R9 to z/OS V1R10 supported up to 64 CPUs

In anticipation of new machines that will support greater than 64 CPUs, new function has been added to z/OS V1R11 to support up to 100 CPUs. The software infrastructure in z/OS V1R11 supports 128 CPUs but the maximum allowed is limited to 100 to ensure that 3-digit decimal CPU IDs are not used in system messages.

CVT_G64CPU_INFRASTRUCTURE is a new CVT bit which says that the greater than 64 CPUs infrastructure is in use. This is a bit declare over CVTZOS_V1R11, the X'04' bit of CVTOSLV5.

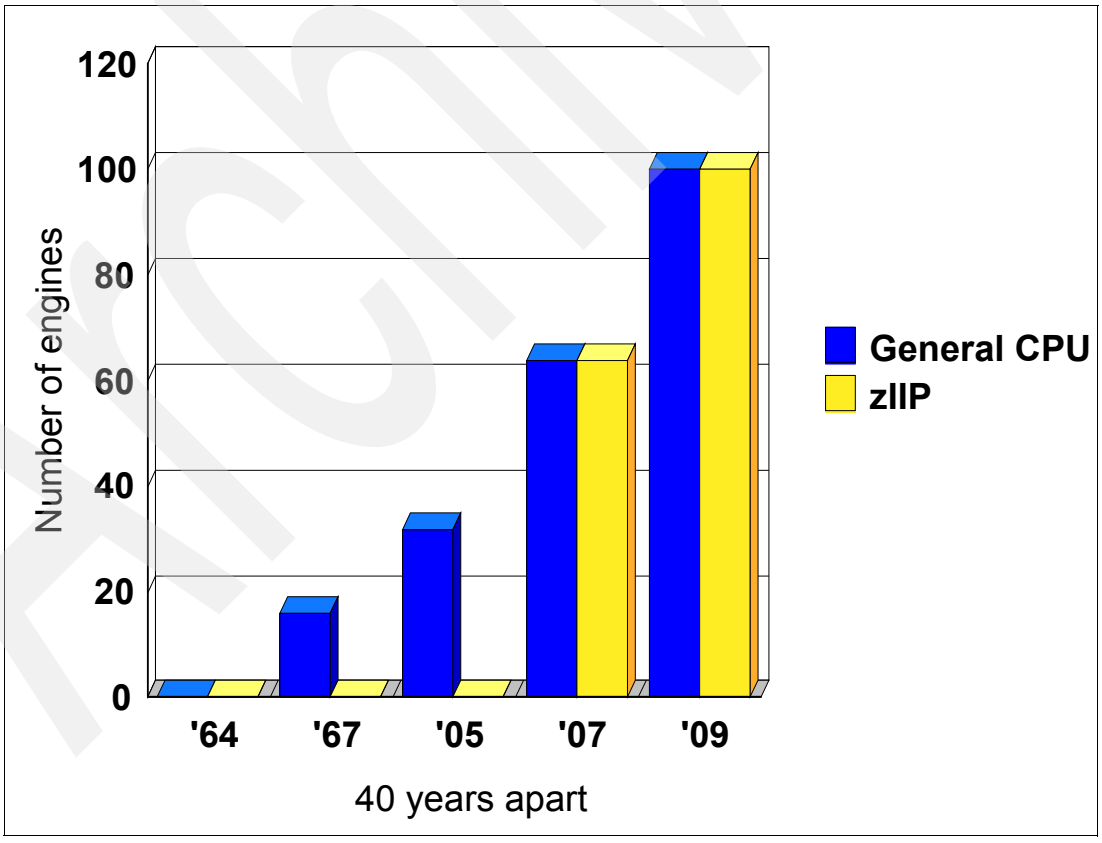


Figure 13-1 An order of magnitude in 40 years

New function details

The support for greater than 64 CPUs is available.

With a maximum of 64 CPUs, a 64-bit CPU mask can conveniently fit in an 8-byte register. Updating the software infrastructure to support up to 128 CPUs required that CPU bit masks were changed to pointers to a new control block where arrays of masks can be stored.

With this support, the conversion of physical CPU IDs to logical CPUIDs and visa versa changes to use the X'80' bit rather than the X'40' bit. New fields have been added to the ECVT for use in physical-to-logical and logical-to-physical conversion, as follows:

- ECVTPhysicalToLogicalMask = X'0080'
  This is ORed to the physical CPU ID to obtain the logical CPU ID.
- ECVTLogicalToPhysicalMask = X'007F'
  This is ANDed to the logical CPU id to obtain the physical CPU ID.

The following fields have been added to the ECVT for use during CPU manipulation:

- ECVT_max_highestCPUID = 99
- ECVT_max_CPUMaskSizeInBits = 128
- ECVT_max_CPUMaskSizeInBytes = 16
- ECVT_zOSR11_highestCPUID = 99
- ECVT_zOSR11_CPUMaskSizeInBits = 128
- ECVT_zOSR11_CPUMaskSizeInBytes = 16

Changes to CPU masks

The following masks will remain 64 bits long:

- LCCA_CPU_ADDRESS_MASK
- PCCA_CPU_ADDRESS_MASK
- SI00PCCA_CPU_Address_Mask

To support greater than 64 CPUs, new offset fields have been created associated with each of the preceding fields:

- LCCA_CPU_ADDRESS_MASK_OFFSET
- PCCA_CPU_ADDRESS_MASK_OFFSET
- SI00PCCA_CPU_Address_Mask_Offset

These fields contain the 8-byte block that the CPU mask is in and are a multiple of 8, representing which double word of the mask this CPU is in. For example, Figure 13-2 shows the mask (either LCCA_CPU_ADDRESS_MASK, PCCA_CPU_ADDRESS_MASK, or SI00PCCA_CPU_Address_Mask) and its offset, as well as the actual mask it represents.

CPU ID	64 Bitmask	Offset	Actual Mask
7	**0100000000000000x**	**0**	**0100000000000000 0000000000000000 x**
71	**0100000000000000x**	**8**	**0000000000000000 0100000000000000 x**

Figure 13-2 Examples of 64-bit CPU masks with the corresponding block offset

The rest of the masks and arrays shown in Figure 13-3 will no longer be available in their respective control blocks. To obtain the mask or array, a new field is defined in the same control blocks named <fieldname>_ADDR as a pointer type.

Macro	Field name
IHACSD	CSD_CPU_ALIVE
IHACSD	CSD_zIIP_MASK
IHACSD	CSD_BYLPAR_ZIIP_MASK
IHACSD	CSD_SUP_MASK
IHACSD	CSD_BYLPAR_SUP_MASK
IHACSD	CSD_CP_MASK
IHACSD	CSD_BYLPAR_CP_MASK
IHACSD	CSD_IFA_MASK
IHACSD	CSD_BYLPAR_zAAP_MASK
IHACSD	CSD_BYLPAR_IFA_MASK
CSRSIIDF	SIOOPCCA_CPU_Address_Mask
IEEMRCPT	RCPT_ARRAY (and all sub structures)
IHACSD	CSD_CPUS_MANIPULATED_BY_WLM*
IHACSD	CSD_CPUS_GOING_ON_OR_OFF_MASK*
IHACSD	CSD_VARIED_OFFLINE_BY_OPERATOR*
IHACSD	CSD_TAKEN_OFFLINE_BY_ACR*
IHASVT	SVT_CPUS_CAUSING_SPIN*
IHASVT	SVT_WAITING_PROCESSOR_MASK*
IHASVT	SVTWAS*
IHASVT	SVT_Short_Wait_CPUPA*
IHASVT	SVT_Short_Wait_IFAPA*
IHASVT	SVT_Short_Wait_SUPPA*
IHALCCA	LCCA_CPU_ADDRESS_MASK*
IHAPCCA	PCCA_CPU_ADDRESS_MASK*
IGFRWA	RWAPCCAR*
IHATFWA	TFWAPDS (and all sub structures)*
IHATOB	TOBPE (and all sub structures)*
IHATTCH	TTCHPH (and all sub structures)*
IHALCCX	LCCX_CPU_Excluded*
IHALCCX	LCCX_CPU_Excluded_Partial*
IHALCCX	LCCX_NeedHelp_SIGP_Counters (and all substructures)*
IHALCCX	LCCX_Previous_SIGP_Counters (and all substructures)*

Figure 13-3 Masks and arrays affected by support for greater than 64 CPUs

Note: An asterisk (*) denotes a field that is not declared a programming interface by IBM. Contact IBM if this field is being used.

If the field is a bit mask, the pointer will reference an area defined at runtime to contain bits 1 through (CVTMAXMP+1), rounded up to a double word boundary. A new field in the ECVT is defined called ECVTMaxMPNumMaskBytes. This field will be the number of bytes (CVTMAXMP+1) rounded up to a double word boundary. For example, if CVTMAXMP is 27 then ECVTMaxMPNumMaskBytes will be 8. The area referenced by the pointer will be 8 bytes long; however, only bits 0 to 27 will be valid.

If the field is an array, the pointer will reference an area defined at runtime to contain (CVTMAXMP+1) number of entries, unless otherwise noted. In the assembler mapping there will be a DSECT produced for each field, representing an array entry. The total size of this

new data area will be (CVTMAXMP+1) * SIZE(<fieldname>). The pointer will reference the first entry of the array.

If local storage is needed for a CPU mask at assemble time, use a new constant of ECVT_kRecCPUMaskSizeInBits or ECVT_kRecCPUMaskSizeInBytes. These constants represent a useful size such that if the number of CPUs supported by z/OS needs to increase, the increase will be easier for IBM and its clients.

To minimize changes required to your code if the number of CPUs increases, only use ECVT_kRecCPUMaskSizeInBits and ECVT_kRecCPUMaskSizeInBytes when absolutely necessary, such as when determining how much storage to allocate. To copy a CPU mask, IBM requires clearing the storage allocated for the mask, and copying it at runtime using the runtime fields CVTMaxMP or ECVTMaxMPNumMaskBytes.

Figure 13-4 lists fields that contain a logical CPU ID. They are equivalent to the physical CPU ID with the X'80' bit turned on rather than the X'40' bit.

Macro	Field name
IOSDCHRB	CHRACRW
IOSDCRWQ	CRWQCP
IOSDIOWA	IOWCPUA
IHALCCA	LCCACPUA
IHAPSA	PSACPULA
IHAPSA	PSALCPUA
IHASDWA	SDWACPUA
IHASDWA	SDWACPUI
IHASDWA	SDWALCPU
IHASLPL	SLPLLKCP
IHASLPL	SLPLWCPU
IHASTCB	STCBSLID
IKJTCB	TCBXSCT
IHAWEB	WEBLogicalCpuId

Figure 13-4 Fields that contain a logical CPU ID

Note: The fields shown in Figure 13-4 are not declared a programming interface by IBM. Contact IBM if any of these fields are being used.

Samplib member IEESCPUM

A new SYS1.SAMPLIB member called IEESCPUM has been added to demonstrate coding techniques using the new level of indirection introduced by the CPU mask pointers. It contains sample code that uses the new z/OS V1R11 CPU infrastucture to count the number of offline standard CPs on the system. The program shows how to write code that does not have to be modified or recompiled when the number of CPUs increases.

13.3 SRB priority adjustment

Many system outages have been caused by a flood of cross-memory (XM) POST SRBs. This typically occurs when an application running in an address space with a higher priority than the address space being posted continually issues an XM POST regardless of whether the ECB has already been posted. This can cause the number of SRB work element blocks

(WEBs) on the work unit queue (WUQ) to become so long that SPIN loops occur when a new WEB needs to be added or deleted.

New function details

The changes to SRB priority adjustment is designed to avoid a system outage due to a flood of SRBs. It does this by detecting that a flood of SRBs is being scheduled and increasing their dispatching priority to the priority of the scheduler. This means the SRBs are dispatched as often as they are being scheduled, thereby avoiding a large buildup of SRB WEBs on the work unit queue.

Normally, SRBs are dispatched with the priority of the address space in which they execute. If a program in a high priority address space schedules SRBs to a lower priority address space, SRBs might be scheduled faster than they can be dispatched.

When an excessive number of SRBs for an address space are found on the dispatching queue, subsequent SRBs to that address space are given the priority of the scheduling address space if that priority is higher than that of the target address space.

13.4 XM post SRB without local lock

This new function is similar to SRB priority adjustment. It is also designed to eliminate a system outage due to a flood of XM POST SRBs.

When the scheduler of XM POST SRBs runs on each of many CPUs and the target address space runs on one CPU, the scheduler can outpace the rate that the target address space can process the SRBs even when they are the same priority. This results in a large and growing number of suspended SRBs even though the ECB might already be posted.

In general, because ECBs do not have to be in common storage, POST cannot look at the ECB for the XM POST because it is running in the wrong address space and if the ECB was in common storage, then the poster should have checked before issuing the XM POST.

When the number of suspended SRBs in the target address space exceeds a threshold, the XM POST is scheduled without the LOCAL lock so that it can check to see if the ECB has been posted. If it has been posted, then processing is complete; otherwise, it joins the normal path and must wait for the LOCAL lock.

For cases where the ECB has already been posted, this avoids having a flood of XM POST SRBs become suspended by having to queue for the LOCAL lock.

13.5 Cross-memory TCB and SRB WAIT

A common system function is to issue a STIMER macro to wait for a period of time before continuing processing. However, an SRB or a TCB running in cross-memory mode is restricted from issuing a STIMER WAIT.

The need to wait in cross-memory mode is commonly required by the GRS component, and is provided through an internal GSR service, ISGXSRBW. Instead of issuing a STIMER WAIT, a disabled interrupt exit (DIE) routine is scheduled to run when the desired time interval expires and the task suspends itself. When the DIE routine runs it schedules another SRB, which performs a RESUME for the suspended unit of work. The result is that the task was in a WAIT for the requested interval.

This function is now available for general use in z/OS V1R11 through the Timed wait processing service, IEAVXTSW. This provides the access to the existing ISGXSRBW service. Module IEAVXTSW has a CSECT name of ISGXSRBW so that existing users of ISGXSRBW will not be affected.

A new field called ECVTXTSW has been added to the ECVT which is mapped by IHAECVT. ECVTXTSW contains the address of the cross-memory TCB or SRB wait routine. No executable macro is provided.

The caller must be AMODE 31, key 0, supervisor state, enabled for I/O and external interrupts, holding no locks.

Follow these steps to use this new function:

1. Load the address of ECVTXTSW into register 15.
2. Issue a BASR 14,15.

On entry, R1 contains the address of a standard parameter list. The parameter list consists of a fullword that is the address of an 8-byte area that contains the wait time in TOD clock format.

On exit, GPR 15 contains the return code:

- 0 = Wait successfully completed.
- 16 = Unable to obtain storage to perform the suspend operation.
- 20 = SUSPEND w/TOKEN is prohibited for this SRB. A PURGEDQ might already have been issued for this SRB.
- 24 = An unrecoverable error occurred in SUSPEND processing.

Potential abend codes:

- AC7 REASON-CODE 00410001: RESUME request did not complete normally
- AC7 REASON-CODE 00410002: An error occurred during the timer DIE execution and the FRR abended the owning task for the SRB that was to be resumed.

Figure 13-5 shows how to use the IEAVXTSW service.

```
         USING ECVT,Regx
         L     15,ECVTXTSW
         BASR  14,15
* Place code to check return code here
```

Figure 13-5 Sample code to use the IEAVXTSW service

13.6 SPIN system trace entries

There are many situations where a CPU spins for an event to occur. This occurs most commonly for a lock. Routines that SPIN enable excessive SPIN detection so that action can be taken if the SPINTIME specified in the EXSPATxx parmlib member is reached. When this occurs, the first action is to SPIN and issue a restart, which generates a LOGREC entry for S071-10. However, issuing a restart requires that CVTRSTWD can be obtained and therefore, in many cases a LOGREC entry for S071-10 is not generated. When this occurs it is difficult to determine what caused the SPIN because there are no system trace entries produced.

There are nine modules that commonly SPIN that have been enhanced in z/OS V1R11 to produced SPIN system trace entries to help determine why the SPIN occurred and aid problem diagnosis. The modules are:

- BLWRESRC - Obtain/Release Restart Resource
- IEAVEAC0 - known as ASCBCHAP, manipulates ASCBs on the swapped-in queue and the work unit queue
- IEAVEBBR - BIND BREAK service routine
- IEAVEINT - Intersect SPIN routine
- IEAVELKX - Lock manager service routine extension
- IEAVERI - Interprocessor communication remote immediate Signal (RISGNL) service routine.
- IEAVESGP - SIGP routine
- IEAVESPN - SPINLOOP service routine
- IEAVTMTC - Memory termination controller

These modules now generate SPIN system trace entries at the start of the SPIN and at the end of the SPIN. Actually IEAVERI also generates a SPIN system trace entry in the middle of the SPIN. A SPIN trace entry is generated when the SPIN exceeds 2**20 microseconds which is approximately one second.

The SPIN system trace entries contain a three-character identifier of the module that is SPINning, the return address to identify which module made the request, the duration of the SPIN, and other information depending on which module generated the SPIN system trace entry.

BLWRESRC - Obtain/Release restart resource

BLWRESRC manages obtaining and releasing the restart resource, CVTRSTWD. If CVTRSTWD is held by another CPU it will SPIN, waiting for it to be free. SPIN system trace entries are generated with the following information:

- CD/D: SRC/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time stamp since the start of the SPIN)
- Unique2: current holder (value in CVTRSTWD: high two bytes are CPU address, low two bytes are restart resource ID)
- Unique3: restart resource ID of the caller (input parameter 1)

Figure 13-6 shows examples of start and stop SPIN trace entries generated by BLWRESRC. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
00-0026 005E6D90  SPIN SRC/S          811559EE  01000347 00810000 0000E6E3
00-0026 005E6D90  SPIN SRC/P          811559EE  6453F4A5 00810000 0000E6E3
```

Figure 13-6 Start and stop SPIN trace entries generated by BLWRESRC

IEAVEAC0 - aka ASCBCHAP

IEAVEAC0 manipulates ASCBs on the swapped-in queue and the work unit queue (WUQ). It SPINs when the ASCB being deleted is still in PSAAOLD on another CPU. SPIN system trace entries are generated with the following information:

- CD/D: EAC0/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since start of spin)
- Unique2: CPU that is a current "holder"
- Unique3: the input function code
- Unique4: current_ASCB
- Unique5 input_parm:

Figure 13-7 shows examples of start and stop SPIN trace entries generated by IEAVEAC0. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
00 0027 005E6D90  SPIN AC0/S          8A3F8F4C  010222DE 00000001 00000002
                                                00FCB180 00FCB180
00 0027 005E6D90  SPIN AC0/P          8A3F8F4C  6B7FBD34 00000001 00000002
                                                00FCB180 00FCB180
```

Figure 13-7 Start and stop SPIN trace entries generated by IEAVEAC0

IEAVEBBR - BIND BREAK service routine

IEAVEBBR is the BIND BREAK service routine. It is called by routines that change cross-memory environments to force a retranslation of the cross-memory environment on the other processors in the configuration. IEAVEBBR needs to signal the other CPUs and SPINs while waiting for a response. SPIN system trace entries are generated with the following information:

- CD/D: EBBR/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since start of spin)
- Unique2: CPU address that you are spinning for
- Unique3: input reg 0: the function code
- Unique4: input reg 1: when applicable, the ASID in bits 16-31

Figure 13-8 on page 279 shows examples of start and stop SPIN trace entries generated by IEAVEBBR. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
62 0001 005EBA28  SPIN BBR/S          859CD75C  038BEA39 00000002 0000000C
                                                0000001A
62-0001 005EBA28  SPIN BBR/P          859CD75C  3CE144C7 00000063 0000000C
                                                0000001A
```

Figure 13-8 Start and stop SPIN trace entries generated by IEAVEBBR

IEAVEINT - Intersect SPIN routine

IEAVEINT is the dispatcher intersect SPIN routine. It SPINs on the dispatcher active bits waiting for the dispatcher to exit on each CPU. SPIN system trace entries are generated with the following information:

- CD/D: EINT/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: spin duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: CPU address that you are spinning for found from LCCA/PCCA)
- Unique3: input reg 0: the function code
- Unique4: input reg 1: when applicable, the ASID in bits 16-31

Figure 13-9 shows examples of start and stop SPIN trace entries generated by IEAVEINT. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
01 0001 00000000  SPIN INT/S          83FD4528  010011C2 00000000
01-0001 00000000  SPIN INT/P          83FD4528  7662A234 00000000
```

Figure 13-9 Start and stop SPIN trace entries generated by IEAVEINT

IEAVELKX - Lock manager service routine extension

IEAVELKX is an extension of the spin lock manager service routine IEAVELK. IEAVELKX is called by IEAVELK when a lock is not available, and it manages the SPIN processing while waiting for the lock to become available. SPIN system trace entries are generated with the following information:

- CD/D: ELKX/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: lock word (contains the CPU address that you are spinning for)
- Unique3: input reg 11 (lockword address)
- Unique4: Lock held obtain mask (PLLHSOM, through input reg 12)
- Unique5: Lock held string pointer (PLCLHSP, through input reg 12)
- Unique6: input reg 13 (lock routine entry point address)

Figure 13-10 on page 280 shows examples of start and stop SPIN trace entries generated by IEAVELKX. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD. and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
62 0001 005EBA28  SPIN LKX/S          8113D764  02FD35D9 000000C1 00FD9800
                                                00001000 000002F8 00FEB21A
62 0001 005EBA28  SPIN LKX/P          8113D764  0F0768D4 000000C1 00FD9800
                                                00001000 000002F8 00FEB21A
```

Figure 13-10 Start and stop SPIN trace entries generated by IEAVELKX

Note: IEAVELKX is always called by IEAVELK to perform SPIN processing, so the caller of IEAVELK is placed in ADDRESS.

IEAVERI - RISGNL service routine

IEAVERI is the interprocessor communication remote immediate signal (RISGNL) service routine. It provides the issuer of the RISGNL macro with the necessary interface to signal (SIGP) an alive CPU. It spins while waiting for a response from the CPU being signalled. SPIN system trace entries are generated with the following information:

- CD/D: ERI /{S | M | P}source identifier with start, middle or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: CPU address that you are spinning for
- Unique3: input register 0 (request code and, when appropriate, ASID)
- Unique4: input reg 12 receiving routine's entry point address
- Unique5: input reg 1 (PCCA address of receiving CPU). If this identifies the same CPU as Unique2, simply use Unique2.

Figure 13-11 shows examples of start and stop SPIN trace entries generated by IEAVERI. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
01 001A 005FF890  SPIN RI/S           89F06560  01000022 000000E3 40000000
                                      81212BD8  00FBE3A8
01 001A 005FF890  SPIN RI/M           89F06560  149315FA 000000E3 40000000
                                      81212BD8  00FBE3A8
01 001A 005FF890  SPIN RI/P           89F06560  55A205BD 000000E3 40000000
                                      81212BD8  00FBE3A8
```

Figure 13-11 Start and stop SPIN trace entries generated by IEAVERI

Note: IEAVERI is the only routine that produces middle trace entries, because it can SPIN for an extended period. A middle trace entry is generated when half the SPINTIME specified in EXSPATxx parmlib member is reached. In this example, SPIMNTIME=40 and therefore a middle SPIN trace entry was generated after 20 seconds. Also notice that the stop SPIN trace entry shows that the SPIN duration was about 86 seconds. This is because there are two SPIN intervals before the ABEND action is taken because an automatic SPIN is performed in addition to the SPIN action specified in EXSPATxx.

IEAVESGP - SIGP routine

IEAVESGP is the interprocessor SIGP routine. It spins while waiting for a response from the CPU being signalled. SPIN system trace entries are generated with the following information:

- CD/D: ESGP/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: CPU address that you are spinning for (came from input reg 3)
- Unique3: input reg 1 (parameter reg for status and prefix order codes)
- Unique4: input reg 2 (SIGP order code)
- Unique5: Status returned from the last SIGP

Figure 13-12 shows examples of start and stop SPIN trace entries generated by IEAVESGP. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
50-0001 00000000  SPIN SGP/S          812EE4C2  02508926 00000062 00000000
                                                00000002 00000080
50-0001 00000000  SPIN SGP/P          812EE4C2  3D093941 00000062 00000000
                                                00000002 00000080
```

Figure 13-12 Start and stop SPIN trace entries generated by IEAVESGP

IEAVESPN - SPINLOOP service routine

IEAVESPN is the SPINLOOP service routine. It spins until a user-specified resource is available, which is typically called by dispatcher processing and global recovery. SPIN system trace entries are generated with the following information:

- CD/D: ESPN/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: CPU address that you are spinning for
- Unique3: input reg 0
- Unique4: input reg 1
- Unique5: input reg 3

Figure 13-13 shows examples of start and stop SPIN trace entries generated by IEAVESPN. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6
00 0009 005F56F8  SPIN SPN/S          816B0E60  0100007A 00000081 00000002
                                                0200A700 00000000
00-0009 005F56F8  SPIN SPN/P          816B0E60  2AE54509 00000081 00000002
                                                0200A700 00000000
```

Figure 13-13 Start and stop SPIN trace entries generated by IEAVESPN

IEAVTMTC - Memory termination controller

IEAVTMTC is the memory termination controller. It controls the process of responding to requests for address space termination. It spins while waiting for other CPUs to complete process work for the terminating address space. SPIN system trace entries are generated with the following information:

- CD/D: TMTC/{S | P}: source identifier with start or stop indicator
- PSW/Address: the caller's return address
- Unique1: SPIN duration so far (bytes 3, 4, 5, 6 of time since the start of the spin)
- Unique2: physical CPU number of a CPU that is still running with the terminating ASCB for "S", 0 for "P". This can also be 0 for "S" if an ACR condition is encountered.
- Unique3: ASID being MEMTERMed.

Figure 13-14 shows examples of start and stop SPIN trace entries generated by IEAVTMTC. It shows the leftmost portion of the trace entry. To the right are PSACLHS, PSACLHSE, PSALOCAL, PASD, SASD, and Timestamp.

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
01 0001 005EBC58  SPIN MTC/S          00000000  01000061 00000080 00000028
01 0001 005EBC58  SPIN MTC/P          00000000  65902A9C 00000080 00000028
```

Figure 13-14 Start and stop SPIN trace entries generated by IEAVTMTC

13.7 RTM health check for IEAVTRML

The old or original way to define a resource manager to recovery termination manager (RTM) was to place its name into CSECT IEAVTRML and link IEAVTRML into LPA load module IGC0001C. End Of Task (EOT) and End Of Memory (EOM) resource managers defined to RTM by IEAVTRML run for every task and address space termination in the system. This can impact system performance.

Use the RESMGR service instead of IEAVTRML. The RESMGR service allows EOT and EOM resource managers to be dynamically defined for just the environments where they are required. The RESMGR service has been available since MVS/ESA was released.

In the past, several IBM products required clients to place the names of their resource managers into IEAVTRML as part of the product's installation procedure. There are no longer any IBM products that do this, although several products continue to support their former resource managers defined this way for compatibility.

RTM_IEAVTRML

New health check RTM_IEAVTRML warns when IEAVTRML is still being used and provides a mechanism for accepting legitimate usage situations; see Figure 13-15.

```
UPDATE CHECK(IBMRTM,RTM_IEAVTRML) ACTIVE
SEVERITY(LOW) INTERVAL(ONETIME) DATE(20081015)
PARM('ALL') VERBOSE(NO)
REASON('IEAVTRML should not be used if possible.')
```

Figure 13-15 Syntax for the RTM_IEAVTRML health check

The following parameters are accepted:

- VERBOSE(YES) specifies that all of the resource managers found in IEAVTRML will be listed in the message log whether or not an exception is raised.
- PARM('ALL') specifies that exceptions should be issued for all module names specified in IEAVTRML. This is the system default.
- PARM('NEW(value)') specifies that the current contents of IEAVTRML are to be treated as correct and exceptions should be issued only for new module names added to IEAVTRML after this time. This specification persists across restarts of the Health Checker, including IPLs. Note that the system only recognizes changes to IEAVTRML by an IPL with CLPA.

Tip: Use PARM('NEW(value)') to specify the date and time that the current contents were accepted - PARN('NEW(yyyy/mm/dd hh:mm)')

Starting RTM_IEAVTRML

RTM_IEAVTRML runs automatically every time that the IBM Health Checker is started. To invoke RTM_IEAVTRML manually, enter the following command while the Health Checker is active:

```
F HZSPROC,RUN,CHECK(IBMRTM,RTM_IEAVTRML)
```

Like other health checks, RTM_IEAVTRML can also be invoked by using the CK panel under SDSF.

Note: The RTM_IEAVTRML health check uses the Health Checker HZSPDATA persistent dataset so that it is always activated on Health Checker startup.

Output from RTM_IEAVTRML

If this health check is successful (no exceptions were found), it will issue the following message in the Health Checker message log:

```
IEAVTRH01I CHECK(IBMRTM,RTM_IEAVTRML) was successful.  IEAVTRML contains no module
names.
```

If the module names in IEAVTRML have been accepted, it will issue the following message:

```
IEAVTRH02I CHECK(IBMRTM,RTM_IEAVTRML) was successful.  IEAVTRML contains no new
module names.
```

Important: IGC0001C is loaded into (LPA) storage during IPL and does not change for the life of the IPL. For that reason, this health check cannot detect changes to IEAVTRML until after the following IPL.

If RTM_IEAVTRML finds a module name that has not been accepted by check parameter 'NEW(value)') in IEAVTRML, it issues the following exception message:

```
IEAVTRH04E IEAVTRML contains module names
```

The Health Checker message log will then contain a message indicating up to 16 of these names:

```
IEAVTRH03I The following module names in IEAVTRML caused an exception:
IEFBR14  IEFBR15  IEFBR16
```

If this check finds resource manager names in IEAVTRML, then investigate to determine whether they are still appropriate. In most cases, they will be found to have been superseded by the use of RESMGR and can be removed from IEAVTRML.

Examples of this type of resource manager name include:

- DFSMRCL0 - no longer required as of IMS Version 9
- MVPTTRML - no longer required in any supported release
- BNJMTERM - no longer required in any supported release

If a resource manager module name is still required in IEAVTRML, use check parameter 'NEW(value)' to set the check to accept any current module names and to only flag future additions to IEAVTRML as exceptions.

Controlling RTM_IEAVTRML

If a resource manager module name is still required in IEAVTRML, the RTM_IEAVTRML health check parameter 'NEW(value)' should be set to accept any current module names and to only flag future additions to IEAVTRML as exceptions.

The 'NEW(value)' parameter and the resource manager module names that it accepted persist across restarts of this check including across IPLs. For example:

```
F HZSPROC,UPDATE,CHECK(IBMRTM,RTM_IEAVTRML),PARM('NEW(2009/02/27 16:49)')
```

RTM_IEAVTRML supports VERBOSE mode. When VERBOSE=YES is specified after accepting resource manager names with NEW(value), all of the module names found in IEAVTRML will be written to the Health Checker message log whether they caused an exception or not. VERBOSE mode is useful for auditing which resource manager names have been accepted with PARM('NEW(value)').

An example of requesting VERBOSE mode is:

```
F HZSPROC,UPDATE,CHECK(IBMRTM,RTM_IEAVTRML),VERBOSE=YES
```

If an exception is raised (after NEW(value) was specified) and VERBOSE mode is active, then there will be messages in the Health Checker message log for both the exception and for the VERBOSE output. For example, if module name IEFBR14 had been accepted in IEAVTRML and the check now also finds IEFBR15, it issues the IEAVTRH04E IEAVTRML contains module names exception message followed by:

```
IEAVTRH03I The following module names in IEAVTRML caused an exception:
IEFBR15

IEAVTRH11I IEAVTRML contains the following module names:
IEFBR14  IEFBR15
```

14

JES2 and JES3 enhancements

This chapter describes the functional changes made to JES2 and JES3 in z/OS V1R11.

In this chapter we introduce the following JES2 enhancements:

- New z11 checkpoint activation level
  - JOAs – Artificial JOE
  - $ACTIVATE and $D ACTIVATE
    - New z11 functions provided including SSI improvements
  - Infrastructure changes in support of z11 activation
    - $DOGJOE
    - Macro changes - $#JOE, $#BLD, $#ADD, $#REP, $#BUSY, $#ALCHK and others
  - JES2 exit changes
  - Support for more checkpoint versions
  - Health Checks in support of z11 checkpoint readiness

In this chapter, the following JES3 enhancements are implemented:

- JES3 dump exit
- JES3 SYSLOG browse
- JES3 SDSF SSI enhancements

14.1 JES2 V1R11 enhancements

JES2 V1R11 has added support for selecting SYSOUT by using the SYSOUT application programming interface (SAPI) and extended status by transaction job names and job IDs. To do this, the JOE data structure is extended with JES2 BERTs.

To coordinate when this change occurred, a new $ACTIVATE level was added to JES2. It is this $ACTIVATE level and the service changes it required that are the main and most important changes in this release.

Additionally, there is a small enhancement to the **$S SPOOL** command. In addition, the current job (JQE) and SYSOUT (JOE) limits are too small. Increasing the limits requires all members to be at a release that supports the new limits.

Also, the number of checkpoint versions that can be defined is too small for many clients. The number of versions that can be supported is dictated by how many versions fit in a single data space. This release increases the size of a checkpoint version and thus further reduces the maximum number of versions that are supported.

14.2 SAPI and extended status enhancements

Transaction data is any SYSOUT data that has job level information associated with a SYSOUT data set. These are always SPIN data sets. The original creator of these data sets was APPC initiators. Later BPXAS address spaces exploited the same support. The support was generalized so that any address space can create transaction output by creating a task level JSAB using the IAZXJSAB macro. This is how “server” address spaces such as InfoPrint Server can create transaction output.

Currently, SAPI and extended status do not support selection by APPC (transaction) job name or job ID. This change has been requested by a number of vendors. As more print work moves to z/OS and OMVS (BPXAS) applications increase, the problem of selecting work by transaction identifier instead of address space name increases. Also, the transaction information is only available in extended status if a verbose request is made. This implies that spool I/O for every SYSOUT element is to be returned.

Because the needed information is not in the checkpoint (associated with the JOE), it cannot be added to the JOE because the current structure has no space to place the needed data into it. Therefore, the JOE needs to be extended so that the transaction data can be added.

This requires that code which updates a JOE must first check the JOE out, update the checked-out copy, and check it back in. This changes many services that manipulate JOEs, including certain changes to basic interfaces to the service.

14.2.1 Transaction SYSOUT selection

The SYSOUT level job name and job ID can now be used to select SAPI output by setting the SSS2STPN and SSS2STPI selection bits. Job name supports generics and job ID supports both generics and ranges. Similarly, SYSOUT level job name and job ID can be used to select SYSOUT using extended status.

The new fields that support these changes are:

- The STATSTPN field enables transaction job name

  The job name supports generics and the job name list.
- The STATSTPI field for transaction job ID

  Transaction job ID supports generics and a range.
- The STATSTPU field for transaction owner

  Transaction owner supports generics. The transaction owner allows you to select all jobs and transaction output owned by the same user ID.

Transaction data section

The SYSOUT terse data returned from extended status now includes a new transaction data section. If the SYSOUT has transaction data, this section (STSATERS section type) contains the transaction job name and job ID. All the transaction data is only available if the JESPLEX is in z11 checkpoint mode (z11 activation level).

14.3 $JOE data structure extended with BERTs

The current structure of the $JOE data area (represents a group of SYSOUT that is ready to print) cannot be easily expanded without a cold start. This limits what can be done now and into the future. A solution is needed that allows additional data to be easily added to the checkpoint.

With this release, BERTs are now being associated with JOEs. This change is oriented also to be able to deal with BERT shortages. The JES2 $JOE data area has been enhanced to support BERT extensions. A new $DOGJOE service and macro has been created to encapsulate the expanded JOE. With the new $ACTIVATE level, the JOE fields needed by both SAPI and extended status to do filtering on transaction data are now available. Extended status can also now return transaction data in terse sections. At this point, JES2 has the flexibility to easily add additional fields to the $JOE.

A new health check was added to display if you have met all the requirements for z11 mode.

Note: Because all members must be at the z/OS V1R11 level of JES2 for these services to be available, a new checkpoint level, z11, is created to allow installation to decide when the new processing is enabled. This can have implications for certain exits and user modifications.

JOX and BERTs added to JOEs

The existing work and characteristics JOE still exist in this release. The additional data that is needed to be associated with the work JOE has been added to a new type of JOE BERT. To make room for the larger MQTR pointers, a new section has been added to the CKPT known as the JOX, as shown in Figure 14-1 on page 289. There is one JOX for every JOE, but only the JOX for the work JOE is currently used. MTTR fields in the work JOE have been moved to MQTR fields in the JOX. All these fields are then mapped into a single data structure, the artificial JOA, by the $DOGJOE service.

In z2 mode, the JOEs look the same as they have in previous releases. There is no JOX nor data in BERTs. In z11 mode, the three MTTR fields in the work JOE are moved into new MQTR fields in the JOX, one of the old MTTR fields is reused as the BERT token, and new data is placed in the BERTs. Regardless of mode, the JOA always appears to be in z11 mode

(that is, MTTR fields moved to MQTR field). This insulates code from the current $ACTIVATE level.

Artificial JOEs called JOAs

There are three types of artificial JOEs called JOAs. They all contain the same data, but are used differently. JES2 provides several types of artificial JOEs (that is, JOAs) to installation exits and processes them in differing ways. Consider the following use of the types of JOAs:

- Read-mode JOA

  An exit receives an artificial JOE that is a temporary block of storage. This storage contains information about the work JOE, the characteristics JOE, and the JOE extension (JOX). Information about BERTs (another checkpointed area) is data that is owned by JOEs and is new for JES2 V1R11 code running in z11 checkpoint activation mode. The first two are created by the $DOGJOE service directly and use storage key 0. This JOA is used JOA when issuing display commands or while a JOE is printing.

- Update-mode JOA

  User exits receive an artificial JOE as a temporary block of storage. This storage is similar to the read-mode JOA. This JOA can be updated.

- Work area that contains a prototype JOA

  In certain circumstances, user exits might be passed the address of a work area that contains a working copy of a JOA. For example, a prototype JOA is embedded in the JOE information block ($JIB). They are often imbedded in other data areas and might not contain all the BERT data that a regular JOA has. They can be used as input to services like $#ADD, but in general only exist to save JOA information where a read mode JOA is not practical. See User Exit 23 for more information.

Note: The BERT tables ($BERTTABs) add extensions to existing JES2 checkpointed control blocks such as job queue elements (JQEs), and now in z/OS V1R11, JOEs.

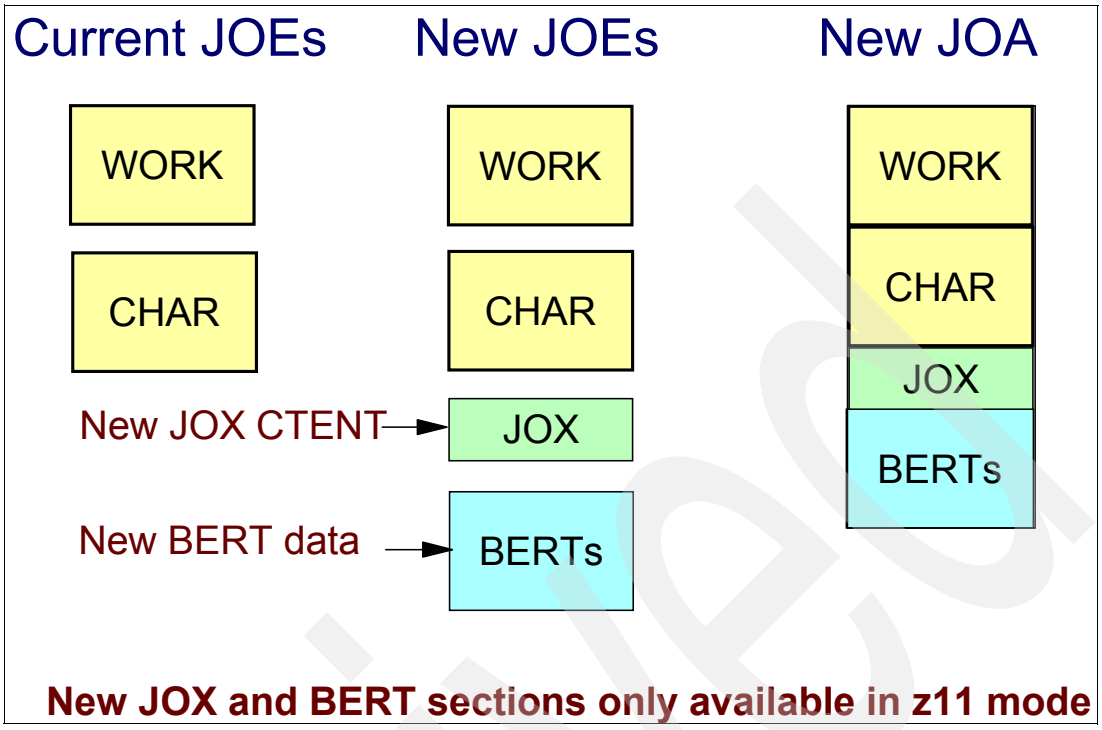


Figure 14-1 $JOE data area changes

Important: The mapping of the JOE DSECT has changed with this release. In previous releases, the characteristic (char) JOE was mapped over the work JOE in the DSECT. Because of this, and to simplify accessing the new JOA structure, the $JOE data area has been re-mapped. The char JOE now follows the work JOE in the mapping, as shown in Figure 14-1 on page 289. New JOX and BERT fields follow those areas.

This requires a change to any USINGs that are used to address the characteristics JOE. Use the JOE (or JOA) label in general to address an artificial JOA. Use the JOEWSTRT and JOECSTRT to address the work and characteristics JOEs. It is strongly suggested that the USINGs specify the proper limits so that they are not used accidentally to access data beyond the end of the particular JOE to which you are pointing.

14.4 z11 checkpoint activation

The PARM=UNACT JES2 start parameter has been reintroduced in this release. If PARM=UNACT is specified, then JES2 starts in z2 mode. It overrides the default and the new OPTSDEF COLD_START_MODE statement. PARM=UNACT can be used on an all-member warm start, an all-member hot start, and a cold start.

A new JES2 initialization option has been added to the OPTSDEF initialization statement. COLD_START_MODE= specifies what checkpoint mode ($ACTIVATE LEVEL) JES2 is to be in when it cold starts. If not specified, JES2 will cold start in z11 mode. However, if specified, JES2 will not only start in the specified mode, but will issue messages if the specified mode does not match the mode of the current warm start. The specified mode is also verified after a $ACTIVATE to ensure it matches the mode that was $ACTIVATEd.

Specify this keyword in each JES2 member's initialization deck (or in a deck common to all members). By specifying COLD_START_MODE, you ensure that an emergency cold start does not result in any surprises.

14.4.1 $ACTIVATE command with JES2 V1R11

Because all members must be at the z/OS V1R11 level of JES2 for these enhancements to be available, a new checkpoint level was created to allow installations to decide when the new processing is enabled. Similar to previous releases, the **$ACTIVATE** command has been reintroduced in this release. The new checkpoint level is for z11 mode (the current checkpoint level is known as z2 mode). $ACTIVATE can be used to move from the z2 level to z11 mode and back.

Attention: If the current mode is z2, you can start pre-z/OS V1R11 levels of JES2 if you have not increased any of the JQE, JOE, or BERT limits above the z/OS V1R10 supported levels. If the new limits have been exploited, then you must reduce the limits before starting an older release (can be done while in z2 mode).

Use the **$ACTIVATE** command to activate to the z11 checkpoint level. The **$ACTIVATE, LEVEL=Z11** command expands the JES2 checkpoint to support functions that are enabled with the z11 checkpoint level. JES2 rejects the **$ACTIVATE** command if certain conditions are not met.

Note: The LARGEDS support must be active (not set to FAIL) to activate z11 mode.

The **$ACTIVATE** command verifies that the current environment supports the target level (all members running z/OS V1R11, no job or output queue errors, and LARGEDS is active). If a problem is found, $ACTIVATE reports the first problem it encounters in the command response. If you need a complete list, then use a **$D ACTIVATE** command. In general, issue a **$D ACTIVATE** command as part of the planning for a **$ACTIVATE command**.

14.4.2 $D ACTIVATE command

The **$D ACTIVATE** command informs you of the current checkpoint level. The **$D ACTIVATE** command provides an exhaustive list of reasons that block checkpoint activation to the z11 checkpoint level. Use the **$D ACTIVATE** command before attempting an activation to the z11 checkpoint level.

The command output is shown in the Notes in Figure 14-2:

1. Current checkpoint mode (z2 or z11 mode).
2. Current checkpoint configuration (z2 or z11 mode):
   - Number of BERTs in the checkpoint along with current utilization.
   - Current checkpoint size in 4 K pages.
3. Future checkpoint configuration after transitioned to z11 mode by using the **$ACTIVATE,LEVEL=z11** command. This includes the number of additional BERTs that will be used after activation with the projected BERT utilization and the projected checkpoint size.

Note: This section only appears when the current checkpoint is z2.

4. Indication of whether a **$ACTIVATE,LEVEL=z11** will succeed or not, along with ALL issues preventing checkpoint activation to the z11 level. (This section only appears when the current checkpoint is z2.)

With the new **$ACTIVATE** level, the JOE fields needed by both SAPI and extended status to filter on transaction data are now available. Extended status can also now return transaction data in terse sections. Therefore, JES2 has the flexibility to easily add additional fields to the $JOE. With the **$ACTIVATE**, the limits for JQEs, JOEs, and BERTs have all been doubled. A number of changes to the JES2 checkpoint structure now allow continued growth of the data that is stored.

```
$HASP895 $DACTIVATE
$HASP895 JES2 CHECKPOINT MODE IS CURRENTLY Z2                    Note 1
$HASP895 THE CURRENT CHECKPOINT:                                 Note 2
$HASP895  -- CONTAINS 2100 BERTS AND BERT UTILIZATION IS 12
$HASP895     PERCENT.
$HASP895  -- CONTAINS 356 4K RECORDS.
$HASP895 z11 CHECKPOINT MODE ACTIVATION WILL:                    Note 3
$HASP895  -- EXPAND CHECKPOINT SIZE TO 409 4K RECORDS.
$HASP895  -- REQUIRE 0 ADDITIONAL BERTS AND UTILIZATION
$HASP895     WOULD REACH 12 PERCENT.
$HASP895 z11 ACTIVATION WILL FAIL IF ISSUED FROM THIS MEMBER.    Note 4
$HASP895     THE FOLLOWING ISSUES PREVENT ACTIVATION:
$HASP895  -- LARGEDS SUPPORT MUST BE ACTIVATED.
```

Figure 14-2 $D ACTIVATE command messages

Figure 14-3 displays the possible reasons for a checkpoint migration failure. In this example, **$D ACTIVATE** displays the current checkpoint level and information, which shows that the **$ACTIVATE,LEVEL=z11** command will fail due to CKPT1 and CKPT2 data sets being too small. The amount too small is displayed in the record. You can use **$D CKPTSPACE** to view the current size of the data sets and to determine how much bigger they must be.

```
----------Example of a possible failures to migrate checkpoint----------------

$D ACTIVATE
$HASP895 $DACTIVATE
$HASP895 JES2 CHECKPOINT MODE IS CURRENTLY z2
$HASP895 THE CURRENT CHECKPOINT:
$HASP895 -- CONTAINS 500 BERTS AND BERT UTILIZATION IS 26
$HASP895     PERCENT.
$HASP895 -- CONTAINS 143 4K RECORDS.
$HASP895 z11 CHECKPOINT MODE ACTIVATION WILL:
$HASP895 -- EXPAND CHECKPOINT SIZE TO 398 4K RECORDS.
$HASP895 -- REQUIRE 200 ADDITIONAL BERTS AND UTILIZATION
$HASP895     WOULD REACH 66 PERCENT.
$HASP895 z11 ACTIVATION WILL FAIL IF ISSUED FROM THIS MEMBER.
$HASP895     THE FOLLOWING ISSUES PREVENT ACTIVATION
$HASP895 -- CKPT1 IS TOO SMALL BY 255 4K RECORDS
$HASP895 -- CKPT2 IS TOO SMALL BY 255 4K RECORDS.
```

Figure 14-3 $D ACTIVTAE migrate checkpoint failure messages

14.4.3 Converting the checkpoint to z11 mode

In Figure 14-4, JES2 attempts activation to z11 but fails because an extreme BERT shortage has previously been encountered. An extreme BERT shortage will block both z11 and z2 activation. The $HASP895 message indicates that the checkpoint level has remained at z2.

Note: Use the **$D ACTIVTAE** command to display a complete list of situations that might be preventing z11 activation.

```
$ACTIVATE,LEVEL=z11
      $HASP003 RC=(123),ACT
$HASP003 RC=(123),ACTIVATE  - BOTH z2 AND z11 CHECKPOINT
$HASP003              ACTIVATION ARE DISALLOWED DUE TO EXTREME
$HASP003              BERT SHORTAGE. JES2 IS IN A NON-STABLE
$HASP003              STATE. RESTART JES2 AS SOON AS MORE BERTS
$HASP003              HAVE BEEN MADE AVAILABLE.
      $HASP895 $ACTIVATE,LEVEL=z11
$HASP895 JES2 CHECKPOINT MODE IS CURRENTLY z2
$HASP895 CURRENT CHECKPOINT SIZE IS 148 4K RECORDS.
$HASP895 CURRENT NUMBER OF BERTS IS 500.
$HASP895 PERCENT BERTS UTILIZED IS 27 PERCENT.
```

Figure 14-4 Convert the checkpoint to z11 mode

Checkpoint level requirements

One or more of the following messages can appear in the $HASP895 message to indicate what prohibits z11 mode activation, as shown in Figure 14-5. An indication is given of whether **$ACTIVATE,LEVEL=z11** succeeds or not, along with all issues preventing checkpoint activation to z11 level. The issues which can prohibit z11 activation are described as follows:

- LARGEDS SUPPORT MUST BE ACTIVATED.
- RC=(94),JOB/OUTPUT QUEUE ERROR DETECTED - A RC=xx. says that z11 activation is not allowed, and refers you to HASP003 reason code 94 for more information.
- ALL MAS MEMBERS ARE NOT AT THE REQUIRED PRODUCT/SERVICE LEVEL - A level to support activation of the MAS. All members must be at the z/OS V1R11 level.
- AN EXTREME BERT SHORTAGE EXISTS - This prevents a z11 checkpoint activation. JES2 is in a non-stable state. Restart JES2 as soon as possible because more BERTs have been made available.

  JES2 displays the current checkpoint level and information, which shows that the **$ACTIVATE,LEVEL=z11** command will fail due to an extreme BERT shortage that was previously encountered. See the description of message $HASP051 for more information about extreme BERT shortages. An extreme BERT shortage prevents **$ACTIVATE,LEVEL=z11** and **$ACTIVATE,LEVEL=z2**.
- NOT ENOUGH FREE BERTS FOR z11 ACTIVATION - A minimum of xxxx additional BERTs are required to avoid a critical BERT shortage.

```
LARGEDS SUPPORT MUST BE ACTIVATED
JOB/OUTPUT QUEUE ERROR DETECTED
ALL MAS MEMBERS ARE NOT AT JES2 z/OS V1R11
AN EXTREME BERT SHORTAGE EXISTS
NOT ENOUGH FREE BERTS FOR z11 ACTIVATION
CKPTn IS NOT ACCESSIBLE
CKPTn IS TOO SMALL BY xxxx 4K RECORDS
```

Figure 14-5 $HASP895 messages

An extreme BERT shortage and a job or output queue error can also prevent activation to both z2 and z11 mode.

Note: All systems must be at the z/OS V1R11 JES2 level to migrate the checkpoint.

A success conversion of the checkpoint to z11 mode is shown in Figure 14-6.

```
$ACTIVATE,LEVEL=z11
      $HASP895 z11 CHECKPOINT MODE IS NOW ACTIVE
      $HASP895 $ACTIVATE, LEVEL=z11
$HASP895 JES2 CHECKPOINT MODE IS CURRENTLY z11
$HASP895 CURRENT CHECKPOINT SIZE IS 155 4K RECORDS.
$HASP895 CURRENT NUMBER OF BERTS IS 500.
$HASP895 PERCENT BERTS UTILIZED IS 27 PERCENT.
```

Figure 14-6 Successful checkpoint conversion to z11 mode

14.4.4 BERT usage

To assist in determining which control blocks are using BERT, a new operand BERTUSE has been added to the **$D CKPTSPACE** command, as shown in Figure 14-7. This operand must be explicitly specified (it does not appear in a normal **$D CKPTSPACE** command). For each control block type, it displays the current number of BERTs assigned to that control block type and how many control blocks are represented in that count. In the example shown, there are nine BERTs being used by six JQEs.

```
$D CKPTSPACE,BERTUSE
$HASP852 CKPTSPACE
$HASP852 CKPTSPACE  CURRENT BERT UTILIZATION
$HASP852            TYPE         COUNT  CB COUNT
$HASP852            -------- --------- ---------
$HASP852            INTERNAL        11         1,
$HASP852            JQE              9         6,
$HASP852            CAT            114        38,
$HASP852            WSCQ             0         0,
$HASP852            DJBQ             0         0,
$HASP852            JOE              0         0,
$HASP852            FREE           966         0
```

Figure 14-7 $D CKPTSPACE,BERTUSE command

New BERT shortage message

In the past, many instances have occurred where a BERT shortage resulted in a cold start. There is an (mostly disregarded) existing $HASP050 message that warns of the problem, but there was no appreciation of what damage a BERT shortage can cause. To address this problem, a new $HASP052 message that is explicit about the implications of a BERT shortage is issued when BERT utilization drops below 256 free BERTs. The $HASP050 message is now issued on all members of the MAS, as shown in Figure 14-8.

When BERTs are in a shortage situation, another change is to suspend JOE creation, input processing for JOBs, and input processing STCs/TSUs as the number of free BERTs drops to 32, 16, and 8, respectively. This slows down consumption of BERTs at a time when the limits are critical.

```
$HASP052 JES2 BERT resource shortage is critical -- immediate action required
   DO NOT TERMINATE JES2 OR IPL
   Doing so may result in a COLD start.
   Current BERTNUM=nnnnn  Free BERTs = nnn
   [JOE creation processing suspended]
   [No batch jobs can be submitted]
   [No batch jobs, STCs or TSUs can be submitted]
```

Figure 14-8 $HASP052 message for JES2 BERT shortage

BERT shortages during warm start

With z/OS V1R11, JES2 will internally fail processing that cannot obtain the needed BERTs during warm start processing. JES2 will continue to initialize. This implies that several data structures may not be updated correctly or may not be created. Some JES2 functions may not work correctly due to this. However, because JES2 is allowed to initialize, it is possible to free sufficient BERTs to allow a normal JES2 start to continue. In this state, JES2 is essentially broken and must be restarted as soon as the BERT shortage has been relieved. JES2 issues the $HASP051 message, as shown in Figure 14-9, to inform the operator that this state exists.

The only way to restore normal operations is to restart JES2 (a hot start is all that is required). Note that in this state:

- Jobs with duplicate jobs names may run at the same time.
- WLM initiators may not work.
- There may be problems with submitting jobs, creating output, and logging on a TSO user.

```
$HASP051 EXTREME BERT SHORTAGE DETECTED.
   JES2 PROCESSING IS IN A NON-STABLE STATE.
   RESTART JES2 AS SOON AS MORE BERTS HAVE BEEN
   MADE AVAILABLE.
```

Figure 14-9 $HASP051 message for BERT shortage during WARM start

Attention: z/OS V1R11 has added more cases when BERTs are used. This release has addressed all the issues with BERT shortages. It is possible to start JES2 when there is a BERT shortage.

Although this release has provided a measure of safety for the situation, the best advice is to avoid running out of BERTs, and thereby avoid testing the safety net. To display what BERTs are being used for, issue the **$D CKPTSPACE,BERTUSE** command.

Activating z11 checkpoint mode

Activating z11 checkpoint mode enables the following function:

- SAPI (SSI 79) and Extended Status (SSI 80) can now support selection by transaction job name and transaction job ID.
- Extended Status (SSI 80) can now return transaction information (job name, job ID and owner) within the terse SYSOUT section.
- JOE data area extensions supported by BERTs. Initial JOE data area extension includes transaction job name and transaction job ID.
- Fixed JOE extension termed the JOX. JOX only exists in an artificial JOE returned by the new $DOGJOE service.
- Increased the size of the JQX by 32 bytes.
- Added job reader on time to the JQX (used in ENF 70).
- Increased limits for JQEs, JOEs, BERTs, and TGNUM.

  Another problem several large clients reported is that the current job (JQE) and SYSOUT (JOE) limits are too small. The concern is that a problem with their printing subsystem can cause the current limits to be reached within an hour or two. Increasing the limits requires all members to be at a release that supports the new limits. New limits are as follows:

  - JQEs = 400,000
  - JOEs = 1,000,000
  - BERTs = 1,000,000
  - TGNUM MAX = 132,649,472

Additional data in the checkpoint puts pressure on the number of checkpoint versions that can be defined, which is too small in certain clients. The number of versions that can be supported is dictated by how many versions fit in a single data space. This release increases the size of a checkpoint version and thus further reduces the maximum number of versions that are supported; see 14.6, “Checkpoint versions” on page 296.

14.5 JES2 exit considerations

Prior to JES2 V1R11, a real work and characteristics JOE were passed to the exits 1, 15, 38, 46, and 56. Starting with JES2 release V1R11, an artificial JOE (JOA) is now passed to each of those exits. Exits will assume that a read mode JOA is passed and obtain an update mode, if required.

JES2 exit migration considerations

Exits normally use JOAs in read mode (data in the JOA is used but not modified) or in write mode (data in the JOA is modified). The exit will always obtain either a READ or an UPDATE mode JOA. Avoid using the real JOE, if possible.

Note: For JES2 exit writers that need to access these new control blocks, see *z/OS JES2 Installation Exits*, SA22-7534.

Exit 7 changes

Prior to z/OS V1R11 JES2, $CBIO used 4-byte MTTRs internally to determine what record is being read or written. Starting with z/OS V1R11 JES2, $CBIO uses 6-byte MQTRs internally. Though not formally passed to exit 7 ($CBIO exit), several exits are known to locate the track

address being processed. If you have exit routines that examine field CBMTTR, they must be changed to examine field CBMQTR (and realize that it is now a 6-byte MQTR).

14.6 Checkpoint versions

Prior to JES2 V1R11, all checkpoint versions were housed in a single data space. This was plenty of space when checkpoint versions were introduced in JES2 SP 4. However, over the years the JES2 checkpoint has grown and two full checkpoint versions do not fit into a single data space.

With z/OS V1R11, the potential size of a checkpoint has increased significantly and in turn checkpoint version size has also increased. To avoid a capacity issue, multiple data spaces are now used to house checkpoint versions. Sufficient data spaces are created to contain at least 50 checkpoint versions. These changes will have no impact on any applications that are currently using checkpoint versions.

Finally, the total number of JES2 checkpoint versions supported has been increased by allowing multiple data spaces to be used to store the versions. This, combined with performance improvements in extended status, reduces the problems that several large clients have experienced with checkpoint version shortages.

14.7 JES2 health check for $ACTIVATE

A new JES2 health check was added to display whether you have met all the requirements for z11 mode, and it allows for a proper decision to issue a **$ACTIVATE**. This check determines if the system is ready to upgrade the JES2 checkpoint to the z11 level. If the necessary preconditions are already satisfied on all systems, then a message is issued stating that the system can be upgraded to z11 mode:

```
HASPH022E JES2 address space JESA can activate checkpoint mode z11.

Explanation: All requirements for activating to checkpoint mode z11 have been
satisfied. The checkpoints are large enough and there are enough free BERTs.
LARGEDS support is activated and all MAS members are at z/OS V1R11.
```

If prerequisite conditions have not been satisfied, then an exception message is issued indicating that the system is not ready to upgrade to checkpoint level z11, as follows:

```
HASPH028E JES2 address space JESA cannot activate checkpoint mode z11.

Explanation: One or more requirements for activating to checkpoint
mode z11 have not been satisfied. Refer to the additional messages following
this message to determine the specific requirements that have not been
satisfied.
System Programmer Response: Before you can successfully activate to checkpoint
mode z11, the size of the JES2 checkpoint needs to be increased. More
information on checkpoint reconfiguration can be found in z/OS JES2 Commands,
SA22-7526 and z/OS JES2 Initialization and Tuning Reference, SA22-7533.
```

14.8 Checkpoint migration considerations

Fully exploiting z11 checkpoint mode involves these basic steps,:

- Activate LARGEDS support (this can be done before or after migrating to z/OS 1.11 JES2).
- Migrate to z/OS V1R11 JES2.
- $ACTIVATE z11 mode.
- Stabilize your system in z11 checkpoint mode.
- Exploit new functions (such as new limits) only after fallback from z11 mode is not needed.
  - Falling back to a release prior to z/OS V1R11 after the new z11 mode limits have been activated requires that the limit be reset to the z/OS 1.10 maximum (or less).

JES2 code is designed to make it easy to fall back one step. However, falling back two steps may require additional effort, so ensure that your system is stable before moving to the next step.

14.9 JES2 $S SPOOL command to create volume

Currently, if you want to start a spool volume, you must have a volume set up with an appropriately named spool data set. If the spools are full, then this may not be easy to do. To simplify the process of adding a spool volume, the **$S SPOOL** command now supports a SPACE= operand to specify that the data set is to be created. JES2 uses all the appropriate attributes when it creates the data set (further simplifying the process).

Command operand SPACE=

The SPACE= operand indicates that a data set is to be created and specifies the size of that data set. The size can be in cylinders or tracks, or you can specify to allocate the largest size that the volume and JES2 supports. The SPACE= parameter can only be specified on the initial start of a volume. If the volume exists (that is, it can be displayed on **$D SPOOL**), then specifying SPACE= results in an error. The start will also fail if the data set already exists on the volume. The CANCEL and DRAIN operands are not allowed and the FORMAT operand is assumed (because this is always a new data set).

SPACE= can be specified as follows:

MAX This allocates the largest SPOOL data set possible considering the free space on disk.

CYL This allocates a SPOOL data set with nnnn cylinders.

TRK This allocates a SPOOL data set with nnnn tracks.

This function is only valid on the initial start of a volume. You will receive an error if it is specified for a volume that is already known to JES2 (an existing volume). The command will fail if the data set already exists on the volume. As mentioned, you cannot specify the CANCEL or DRAIN parameters. The data set will be formatted even if FORMAT is not specified.

Command example

In Figure 14-10, a **$S SPOOL** command starts spool volume SPOOL7. If SPOOL7 is a new volume, a 100 cylinder-sized data set is allocated. However, if SPOOL7 is already defined to JES2, the command fails and the HASP003 error message is issued.

```
$sspl(spool7),space=(cyl,100)
$HASP893 VOLUME(SPOOL7) STATUS=INACTIVE,COMMAND=(START)
$HASP646 1.3333 PERCENT SPOOL UTILIZATION
$HASP423 SPOOL7 IS BEING FORMATTED
$HASP630 VOLUME SPOOL7 ACTIVE 0 PERCENT UTILIZATION
```

Figure 14-10 Command for adding a spool volume

14.10 JES2 SYSLOG browse

Currently, the SDSF SYSLOG function depends on a single sysplex scope data set, HASPINDX, to access SYSLOG data sets across the MAS. This data set is maintained by SDSF running in various TSO address spaces as users look at the SYSLOG data set. This causes a number of problems, as follows:

- Serialization and data corruption is one problem (each updater of the data set must serialize the update, but there can be data corruption if a user ABENDs while accessing the data set).
- Another problem is performance; in the morning, the first person who accesses SYSLOG can experience a significant delay waiting for the HASPINDX data set to be updated.
- There are also performance issues related to the size of the SYSLOG data sets. These problems are well known among SDSF users.

HASPINDX data set

Additionally, the HASPINDX data set only supports JES2. SDSF needs to recreate the function to support JES3. To address this, JES (both JES2 and JES3) will now maintain an index for the SYSLOG data set as it is created. The index tracks both the job's data sets that make up the logical SYSLOG data set, and the individual records in the data set. The index allows rapid access to records in the SYSLOG by record number and by time. The SPOOL data set browse interface was updated to allow browsing of the logical SYSLOG data set for a system using this index.

JES2 V1R11 HASPINDX enhancements

The HASPINDX support that currently exists in SDSF has been replaced with a new mechanism within JES2. As part of this support, JES2 maintains a chain of the SYSLOG JQEs that exist for each SYSTEM in the MAS. There are two new index structures on spool that track the SPIN data sets that the spool job creates, and the buffers with each data set. These indexes are maintained as the SYSLOG data sets are created (and not built by the application, as was done with the HASPINDX data set). The buffer index maps a record number and time stamp to a spool data record.

These new structures are used when the SYSLOG is accessed by a spool data set browse. Allocation is the same as any other spool data set; however, a special data set name triggers the SYSLOG allocation. After it is allocated and opened, the logical SYSLOG can be accessed using the existing spool browse access methods. In addition, a new format for the argument to POINT is supported to locate a record by time or a specific record. GET processing can also return additional information about the record returned.

Spool data set browse allocation

As mentioned, access to the SYSLOG data set is by a spool data set browse using a special data set name as follows:

```
sysname.SYSLOG.SYSTEM
```

Instorage buffers (including those on other members) can be obtained by setting the ASID in the browse token.

Access to the SYSLOG is protected by a SAF/RACF call made during OPEN processing, The resource name passed on the AUTH call is as follows, where `sysname` is the MVS system name of the system creating the SYSLOG. READ access is required to access the data set.

```
+MASTER+.SYSLOG.SYSTEM.sysname
```

After allocation is complete, processing of the data set uses the same services as any other spool data set browse application.

Note: SYSLOG browse is only supported on z/OS V1R11 JES2 for SYSLOG data sets created on z/OS V1R11 JES2.

If a SYSLOG data set was created on a prior release of JES2, then it will not be returned on the SYSLOG browse request. SYSLOG support is not related to the $ACTIVATE level of JES2.

After a SYSLOG data set has been deleted, it will no longer be returned by SYSLOG browse.

14.11 JES3 dump exit

With JES3 V1R11, when an SVC dump is about to occur and JES3 detects that a subsystem interface call was active when the SVC dump was initiated, the dump exit now includes additional address spaces in the SVC dump data. This new support improves diagnostic and serviceability for JES3 subsystem interface failures by capturing additional data which may be relevant to solving the problem.

Often when an SSI call failure occurs in a user address space, capturing additional JES3 address spaces can help with the failure analysis. JES3 now creates an SVC tailored dump that detects when an address space has an outstanding SSI request and adds the JES3, JES3AUX, and JESXCF address spaces to the dump. This eliminates depending upon an operator collecting the information or service support asking an installation to recreate the failure to capture the data.

JES3AUX address space

The JES3 auxiliary address space (JES3AUX) prior to z/OS V1R8 JES3 was started only when the PRTPAGE= parameter on the MAINPROC statement included a non-zero auxpages subparameter. Starting with z/OS V1R8 JES3, JES3AUX is always started.

JES3AUX provides an alternate place to allocate protected page buffers to optionally reduce the dependence of JES3 on ECSA for these buffers. In addition, JES3 uses the JES3AUX address space for the following purposes, even when auxpages is zero (0):

- The JES3AUX address space provides a cross-memory post for JES3 after an I/O request fails.

- The JES3AUX address space provides a space switching PC routine to make unauthorized application code capable of using various types of Subsystem Interface calls.

14.11.1 SVC dump exit

The SVC dump exit (IATABTDX) is created during JES3 initialization. The IATABTDX dump exit is a new JES3 module that is a dynamic LPA module. The MVS service CSVDYNEX is used to establish the dump exit when JES3 is started. If JES3 is ended, then MVS service CSVDYNEX is used to delete the dump. The delete allows for the dump exit module to be updated when JES3 is started.

IATABTDX exit

During initialization, if the call to the MVS service fails and the dump exit cannot be established, initialization continues and message IAT3207 is written, as shown here:

```
IAT3207 EXIT IATABTDX NOT ESTABLISHED, CSVDYNEX  RETURN CODE rc REASON CODE rsn
```

Because JES3 attempted to establish the tailored dump exit IATABTDX by using the CSVDYNEX service, and because the call to the CSVDYNEX service failed, the message is issued. JES3 initialization continues without the dump exit established. If dumps that are needed later were not requested by JES3, the JES3 address space is not included. The JES3 address space is included if the processing that initiated the dump specified that the JES3 address space is to be included.

Operator response

Notify the system programmer to verify that module IATABTDX is properly installed in LPA. If no errors are found, then search problem reporting databases for a fix to the problem. If no fix exists, contact the IBM Support Center.

This new dump exit conditionally adds the address spaces for JES3, JES3AUX, and JESXCF to an SVC dump. The address spaces are added if at least one ASID, included in the dump, has an outstanding SSI request. Identification of the address spaces with outstanding SSI requests is based upon the same internal activity table counters used to control the setting and resetting of the IAZJSAB activity flags.

Note: Adding the spaces to an SVC dump produces the same results as specifying a **DUMP** command option **JOBNAME=(JES3,JES3AUX,JESXCF)**. The dump exit does not request a cross-system dump because cross-system dumps cannot be requested from a tailored SVC dump exit.

14.12 JES3 enhancements for z/OS V1R11

The JES3 enhancements are for the SDSF version are:

- Multi-system active buffer support
- A new way to browse SYSLOG plus five new relative buffer address formats for POINT requests

JES3 enhancements for the SSI are:

- Addition of SSI 70 to display and alter subsystem maintained scheduler data
- Addition of SSI 82 for JES3 property Information

14.12.1 Multi-system active buffer support

The JES3 spool browse functionality in z/OS V1R10 included the ability to read core buffers of the job writing to a data set. The limitation of this support was that active buffers of SYSOUT data sets can be browsed only when the browser and data set owning job are on the same system.

JES3 V1R11 enhancements

z/OS V1R11 now uses JESXCF as a data path from one system to another to retrieve active buffers from a data set browser. This allows active buffers of SYSOUT data sets to be browsed regardless of on which system the data set owner resides. SDSF at R11 specifies to read active buffers by default, regardless whether the data set browser and owner are on the same or separate systems.

14.12.2 SYSLOG browse

A SYSLOG often consists of multiple SYSOUT data sets, thus making it cumbersome to browse its contents. z/OS V1R10 had a limitation that SDSF users cannot browse the entire SYSLOG using JES3 SDSF.

With JES3 V1R11, the SYSLOG stream is merged into a single logical SYSLOG data set. This allows all the SYSLOG data sets on a system to be logically concatenated into a single logical SYSLOG data set for SYSLOG browse requests.

With this new support, JES3 users can now access the SYSLOG as a single entity. Instead of searching multiple SYSLOG data sets, searching can be performed on a single, logical SYSLOG data set.

New SYSLOG support

On the main SDSF panel, selecting LOG now allows access to all the concatenated SYSLOG data sets in a seamless manner. For example, instead of searching for text in each SYSLOG data set one by one, only one search is necessary. The logical concatenation of the SYSLOG data sets is named as follows, where `sysname` is the main processor name:

```
sysname.SYSLOG.SYSTEM
```

14.12.3 SSI 70 support

Prior to JES3 z/OS V1R11, users were unable to display or alter many SYSOUT data set characteristics. With this release, the newly accessible information includes most of the characteristics that can be specified on the JCL OUTPUT statement, and that include:

```
SWB: Scheduler Work Block
SWBTU: Scheduler Work Block Text Unit
```

The scheduler facility services call (function code 70) was introduced in 1991 to provide the ability to change information carried in the SWB for SYSOUT in the HOLD and WTR queues. JES3 now completely supports the scheduler facility services call, including the ability to add, modify or delete all current and future “SWB maintained” fields including but not limited to the following fields:

```
PAGEDEF, FORMDEF, TITLE, NAME, BUILDING, DEPARTMENT, ROOM,  ADDRESS, OUTBIN,
COMSETUP, FORMLEN, COLORMAP, INTRAY, OVERLAYF,  OVERLAYB, OFFSETXF, OFFSETXB,
```

```
OFFSETYF, OFFSETXB, PORTNO, NOTIFY,  USERLIB, RETAINF, RETYL, RETRYT, PRTOPTNS,
and PRTQUEUE.
```

User code requirements

SSI 70 provides the ability to modify or obtain characteristics of SYSOUTdata sets that are controlled by subsystem-maintained scheduler data such as SWBTU data. Most of these characteristics appear exclusively on an OUTPUT JCL statement.

When issuing the IEFSSREQ macro, the caller must ensure that R1 points to a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits, and that R13 points to a standard 18-word save area.

The caller must meet the following requirements:

Minimum authorization:	Problem state, any PSW key
Dispatchable unit mode:	Task mode
AMODE:	24-bit or 31-bit
Cross memory mode:	PASN=HASN=SASN
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	The SSOB, SSIB, and SSSF control blocks can reside in storage above or below 16 megabytes.

Scheduler facility functions

Three scheduler facilities functions are supported by SSI 70:

- SWB merge

  This is the SWBTU retrieval function. The current SWBTU information associated with the sysout data set is returned. Optionally, this can include “compatibility” SWB information. The SWB merge function obtains storage to contain the returned SWBTU data and SJFREQ token (if requested). This function frees that storage.

  Note: JES3 accepts only a data set level token. In other words, SSSFFDTK must point to a valid token. Group level tokens are not checked.

- Cleanup

  This is a housekeeping function. The SWB Merge function returns a pointer to storage containing the SWBTU data. This function frees that storage.

- SWB modify

  This allows the caller to add, delete, or change SWBTU values associated with a SYSOUT data set. For example, the caller can change the OUTBIN value for a particular data set.

  This function can be used to add, delete, or change a SWBTU in the SWBTUs associated with a SYSOUT data set. For JES3, this data set is identified by a client token (mapped by IATYCTKN). There are three sources for this token:

  – SSI 80 (extended status) in which a data set level token is returned in field STVSCTKN on a verbose output request.

  – SSI 79 (SAPI) in which a data set level token is returned in field SSS2DSTR for each data set returned.

– The DYNALLOC macro in which an 80-byte JES client token is returned in text unit DALRTCTK (number 0071) at offset 6.

Note: To change an existing SWBTU's value or add a new SWBTU, the caller must supply a set of SWBTUs for the values to be changed or added. This set must consist of a standard SWB prefix (IEFSJPFX) followed by zero (0) or more SWB text units (IEFDOTUM). The set of possible keys are in IEFDOKEY.

JES3 supports modify by data set token. It does *not* support modify by job name, job ID, or group name.

JES3 requires the caller to have SAF alter authority to the target SYSOUT data set. It does not support (or check) the destination check indicator flag (SSSFDEST) or the SECLABEL dominance check flag (SSSFSECL).

JES3 has a synchronous interface to SWB modify. Operation results are communicated by return and reason code. It does not require a CART ID (SSSFCART) or console ID (SSSFCNID). The caller does not need to monitor for reply messages.

To delete an existing SWBTU for the current set associated with a SYSOUT data set, the caller must provide an erase list that contains a list of keys. The list is an array of full words, one word per entry. Each entry consists of two half words, as follows:

– The first contains the key for the SWBTU that will be deleted.
– The second is set to zero (0). See IEFDOTUM for a mapping (first two fields in the DOCNUNIT DSECT).

Also, the caller must be SAF authorized to alter the target SYSOUT data set.

14.12.4 SSI 82 new function code

This new service to the subsystem interface (SSI), SSI 82, provides a common interface for both JES2 and JES3 to return information about multiple JES-managed structures. Instead of coding programs that traverse internal structures to retrieve this information, the SSI contains packaged information into a single call.

User calls to SSI 82

The full set of returned information and the full set of available filters are defined by the MVS macro, IAZJPNJN. To call SSI 82, set up the SSOB so SSOBFUNC is set to 82, and so SSOBINDV addresses the extension area mapped by the IAZSSJP macro.

To request the NJE Node subfunctions, set SSJPFREQ in the extension area mapped by IAZSSJP to either obtain information about the NJE Nodes (SSJPNJOD) or to release storage accumulated from previous requests (SSJPNJRS). Also within the storage described by IAZSSJP, set the variable SSJPUSER to the address of the User Parameter area defined by the macro IAZJPNJN.

The caller can specify filters within the user parameter area to reduce the amount of data returned by the SSI. If the caller has requested to obtain NJE node information, then after the SSI 82 request returns to the caller, the information about the NJE nodes will be defined by the macro IAZJPNJN.

Returned information from SSI 82

A user-supplied program can call SSI 82 to obtain the following information:

- NJE (network job entry) nodes

  JES3 network job entry (NJE) allows JES3 users at one location to send jobs to another JES3 location for execution, to send output (SYSOUT) data to another JES3 location for processing, and to send jobs, commands, and messages to another JES3 location or to a non-JES3 location.

  Therefore, JES3 can become a part of a network comprised of multiple systems in multiple configurations. For example, a network might include the following:

  - Job entry subsystem to a network job entry (JES2 NJE) configuration
  - A virtual machine/remote spooling communications subsystem (VM/RSCS) configuration
  - A VSE/POWER system
  - JES3 complexes

  Each complex in the network is called a node and is identified by a unique node name. System programmers define nodes by name during JES3 initialization. The name of a JES3 node will always represent a global processor along with *all* of its local processors within the JES3 complex. (The only time a node name can represent a single JES3 processor is when JES3 at that node has no local processors.)

  Networking is not to be confused with remote job processing (RJP), because the concept of nodes does not really apply in an RJP environment. In an RJP environment, users are connected to the computer from a remote distance, but are connected directly and not through another computer serving as a node.

- Spool volumes

  The spool Information subfunction of SSI 82 provides information on the JES3-managed spool partitions and data sets. Information can be obtained on all the spool partitions and data sets, or filters can be specified to limit which ones are returned. For JES3, a spool data set is synonymous with spool extent and can be allocated on any direct access storage device (DASD). The volume on which a spool data set is allocated must be accessible to the global processor and to all local processors. Each spool data set must be contained in a single extent. (A single extent is one adjoining group of tracks or cylinders.) You cannot allocate any secondary extents.

  A spool partition is a logical group of spool data sets. You can specify the spool partitions that JES3 is to use for each processor, for each job class, and for each SYSOUT class. There are several benefits to defining spool partitions. Distributing job classes and SYSOUT classes across multiple partitions can improve performance by limiting contention for a particular partition. Each partition can be defined with a particular track group size, thereby making for more efficient use of spool space. In addition, you can isolate work to specific partitions to ensure the required space is available, route performance critical jobs to higher performance spool extents, and avoid failures in one partition from affecting work in another partition.

  The full set of returned information and the full set of available filters are defined by the MVS macro, IAZJPSPL. To call SSI 82, set up the SSOB so SSOBFUNC is set to 82, and so SSOBINDV addresses the extension area mapped by the IAZSSJP macro. To request the spool Information subfunctions, set SSJPFREQ in the extension area mapped by IAZSSJP to either obtain SPOOL information (SSJPSPOD) or to release storage accumulated from previous requests (SSJPSPRS). Also within the storage described by IAZSSJP, set the variable SSJPUSER to the address of the User Parameter area defined by the macro IAZJPSPL. The caller can specify filters within the user parameter area to reduce the amount of data returned by the SSI.

If the caller requested to obtain spool information, then after the SSI 82 request returns to the caller, the information about the spool partitions and extents is defined by the macro IAZJPSPL.

- Initiators

  For JES3, the initiator subfunction reports information about the resources associated with job execution. These resources include groups, the systems on which these groups are enabled, the classes assigned to the groups, and the initiators that are allocated for managing jobs. The initiator is an integral part of z/OS that reads, interprets, and executes the JCL. An initiator manages the running of batch jobs, one at a time, in the same address space. If ten initiators are active (in ten address spaces), then ten batch jobs can run at the same time. JES performs various JCL processing, but the initiator performs the key JCL work. In fact, the primary purpose of JCL is to tell an initiator what is needed for a job to avoid resource conflicts with other jobs.

  Initiators are defined through the group. In a group, initiators are assigned separately to each system for which the group can be scheduled. For example, GRPA can have three initiators assigned for SYS1 and five initiators assigned for SYS2. The number of dedicated initiators for a group defines the maximum number of jobs of this group that can be scheduled at the same time on a particular system.

  If the initiators are dynamically allocated for a group, then when the first job is scheduled on a system, all of the initiators are allocated at the same time for that system. In contrast, if the initiators are demand allocated, then they are allocated one at a time to handle jobs as they are scheduled. In both cases, the maximum number of initiators for a group that can be allocated for a system is defined by the dedicated initiator counts. The count of in-use initiators for a group is equal to the number of jobs of that group that are currently scheduled on a system.

  The number of allocated initiators for a group on a particular system is the count of initiators that are either in use or waiting for a job to be scheduled.

  Note: The initiator counts given by this subfunction are from the perspective of the group. The count shows the initiator counts for a particular group on a particular system.

  The full set of returned information and the full set of available filters are defined by the MVS macro, IAZJPITD. Because of technical restrictions, JES3 does not report any of the initiator specific information found in the ITIHDIHD sections of this macro. To call SSI 82, set up the SSOB so SSOBFUNC is set to 82, and so SSOBINDV addresses the extension area mapped by the IAZSSJP macro.

  To request the initiator information subfunctions, set SSJPFREQ in the extension area mapped by IAZSSJP to either obtain initiator information (SSJPITOD) or to release storage accumulated from previous requests (SSJPITRS). Also within the storage described by IAZSSJP, set the variable SSJPUSER to the address of the user parameter area defined by the macro IAZJPITD. The caller can specify filters within the user parameter area to reduce the amount of data returned by the SSI.

  If the caller requested to obtain initiator information, then after the SSI 82 request returns to the caller, the information about the corresponding group, systems, and classes are defined by the macro IAZJPITD.

- JESPlex characteristics

  JES3 must run in a sysplex environment. A sysplex (system complex) is defined as a single MVS system or multiple MVS systems that use the MVS cross-system coupling facility (XCF). A JESPlex then is considered to be a group of JES3 systems with one

global and multiple local processors, sharing a common SPOOL, checkpoint, and XCF group.

Note the following considerations for running a JESPlex:

– The systems in a JESPlex may be at other levels.

– A JESPlex *must* be contained entirely within one MVS sysplex.

Although it is possible to have multiple JESPlexes within a sysplex, a good practice is to use a one-to-one relationship between the JESPlex and the sysplex. All systems in a JESPlex must be within the same XCF group. In other words, one JESPlex equates to one XCF group. A maximum of 32 mains can exist in one JESPlex. The JES3 system names, as specified on the NAME parameter on the MAINPROC statement, must match the MVS system name, as specified on the SYSNAME parameter in the IEASYSxx parmlib member.

If both the system and member names are specified, only those systems with names that match *both* filters are reported. For example, if the system name filter is S*, and the member name filter is SYS1, then only the system that is named SYS1 is reported. The full set of returned information and filters are defined by the MVS macro, IAZJPLEX.

To call SSI 82, set up the SSOB so SSOBFUNC is set to 82, and so SSOBINDV addresses the extension area mapped by the IAZSSJP macro. To request the JESPlex information subfunctions, set SSJPFREQ in the extension area mapped by IAZSSJP to either obtain JESPlex information (SSJPJXOD) or to release storage accumulated from previous requests (SSJPJXRS). Within the storage described by IAZSSJP, set the variable SSJPUSER to the address of the user parameter area defined by the macro IAZJPLEX.

The caller can specify filters within the user parameter area to reduce the amount of data returned by the SSI. If the caller requested to obtain JESPlex information, then after the SSI 82 request returns to the caller, the information about the classes are defined by the macro IAZJPLEX.

► Job classes

The CLASS initialization statement allows the system programmer to control the key variables in the job scheduling and execution process. Up to 255 job classes can be defined. A CLASS statement must define each job class that appears on the JOB or //*MAIN control statement. If a CLASS statement is not provided, a default class is used.

TDEPTH is the maximum number of jobs from this class that can be running in the JESPlex. TLIMIT is an array that spells out the maximum number of jobs from another class that can be running in the JESPlex and still allow jobs from this class to be scheduled. There is one array entry for each class for which a TLIMIT value was specified on the CLASS statement. MDEPTH is the maximum number of jobs from this class that can be running on a specified main (system) in the JESPlex. MLIMIT is an array that spells out the maximum number of jobs from another class that can be running on a specified main and still allow jobs from this class to be scheduled.

The full set of returned information and the full set of available filters are defined by the MVS macro, IAZJPCLS. To call SSI 82, set up the SSOB so SSOBFUNC is set to 82, and so SSOBINDV addresses the extension area mapped by the IAZSSJP macro.

To request the job class Information subfunctions, set SSJPFREQ in the extension area mapped by IAZSSJP to either obtain job class information (SSJPJCOD) or to release storage accumulated from previous requests (SSJPJCRS). Within the storage described by IAZSSJP, set the variable SSJPUSER to the address of the user parameter area defined by the macro IAZJPCLS.

Callers can specify filters within the user parameter area to reduce the amount of data returned by the SSI. If the callers request to obtain job class information, then after the SSI

82 request returns to the callers, the information about the classes is defined by the macro IAZJPCLS.

Callers can specify another subfunction value to choose which of the five kinds of information they want the SSI to return.

14.13 SAPI and extended status SSIs

Transactions are typically entered because a task to be performed by a user, typically on a workstation, is to run under an APPC or BPXAS initiator. The transaction is given the user's own job identifier so that output it generates can be distinguished from other similar tasks performed by other users. There is a need to obtain information about jobs and SYSOUT associated with the submitting user who originates the transaction, rather than the job processing the transaction.

In JES3 V1R11, additional filtering has been added to the SAPI SSI 79 request to select data sets using the job name or job ID associated with a transaction. Additional filtering has been added to the extended status request, SSI 80, to obtain data for terse requests using the job name, job ID, or SYSOUT owner associated with a transaction.

14.13.1 SAPI data set selection

SAPI data set selection can be done based upon the job ID or job name for the job creating the data set. The new enhancements to SAPI now allow for data set selection based upon the transaction job ID or transaction job name which may differ from the job processing the transaction. Job ID filtering has been updated to allow the generic characters question mark (?) and asterisk (*).

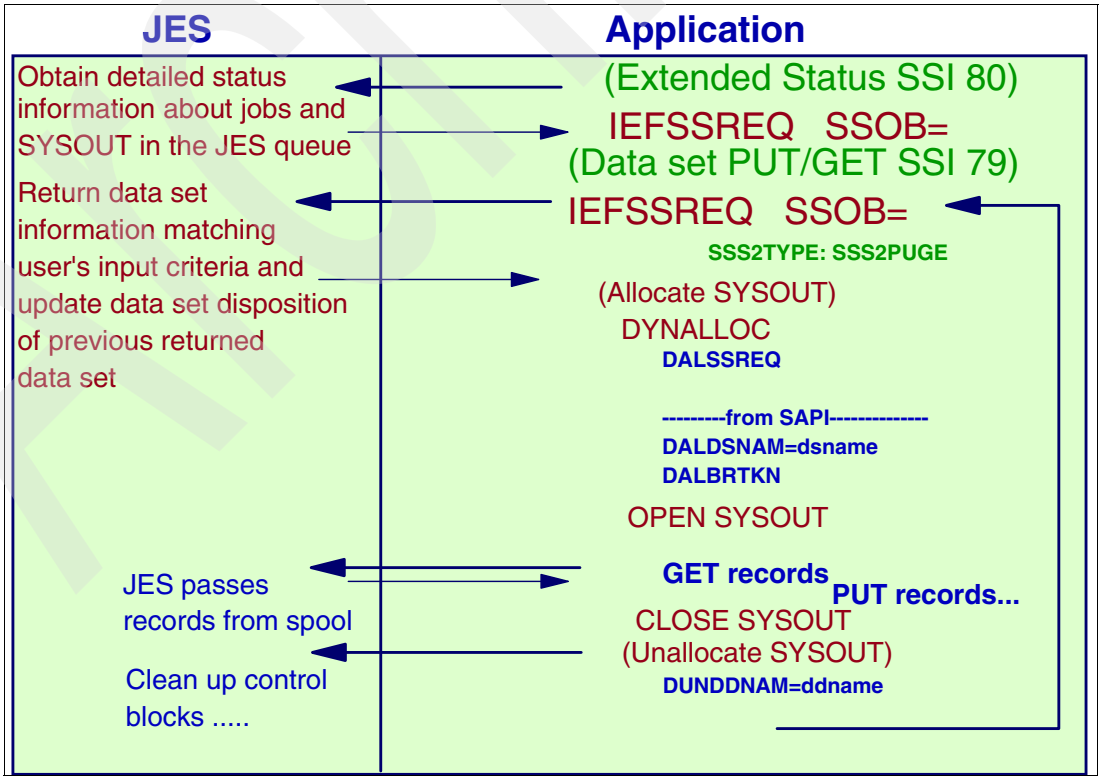


Figure 14-11 Extended status SSI 79 and SAPI SSI 79

Two new SAPI filter options

Two new options are added to SAPI interface in z/OS V1R11 to support the selection of new data sets based on transaction job name or transaction job ID.

These filters are:

STATSTPN Transaction job name filtering. If this bit is on, SYSOUT associated with a transaction job name that matches SSS2JOBN is selected. In addition, SYSOUT that is owned by a job that matches SSS2JOBN is also returned.

The SSS2STPN bit is ignored if one of the following situations occurs:

- SSS2SJBN or SSS2SDUP is not set.

SSS2STPI When SSS2STPI is set and SSS2JBIL is 2 to 8 characters starting with the prefix *, then the suffix is converted to a binary value. Transaction job IDs with a suffix matching the SSS2JBIL suffix are also returned.

When SSS2STPI is set and SSS2JBIL contains 1 to 8 EBCDIC characters A to Z; 0 to 9; national characters @, #, $; or generic characters * and ?, then transaction job IDs that match a 1- to 8-character EBCDIC comparison with SSS2JBIL are also returned. A single character SSS2JBIL with * or ? is not allowed. When generic characters are used, SSS2JBIH must be blank.

Changed filter options

The SAPI job ID filter, SSS2JBIL, can now contain generic characters ? and * for use with job ID and transaction job ID filtering.

Note: Details and a table of example results are provided in *z/OS MVS Using the Subsystem Interface,* SA22-7642.

The changed filters are described here:

SSS2JBIL This is used for the selection of new data sets. The low job ID is used for selection (if SSS2SJBI is on). When SSS2JBIL is 2 to 8 characters and starts with one of the prefixes J, JO, JOB, or *, then the suffix is converted to a binary value. Job IDs with a suffix matching the SSS2JBIL suffix are returned. Embedded and trailing blanks are acceptable. The maximum length of the job ID is eight characters.

When SSS2JBIL contains 1 to 8 characters with one or more generic characters * and ?, and EBCDIC characters A to Z; 0 to 9; or national characters @, #, $, then job IDs, as returned in SSS2JBIR, that match a 1 to 8 character EBCDIC comparison with SSS2JBIL are returned. A single character SSS2JBIL with * or ? is *not* allowed. SSS2JBIH must be blank.

When SSS2STPI is set and SSS2JBIL is 2 to 8 characters starting with the prefix *, then the suffix is converted to a binary value. Transaction job IDs with a suffix matching the SSS2JBIL suffix are also returned.

When SSS2STPI is set and SSS2JBIL contains 1 to 8 EBCDIC characters A to Z; 0 to 9; national characters @, #, $; or generic characters * and ?, then transaction job IDs that match a 1 to 8 character EBCDIC comparison with SSS2JBIL are also returned. A single character SSS2JBIL with * or ? is not allowed. When generic characters are used, SSS2JBIH must be blank.

SSS2JBIH This is the input field used for the selection of new data sets. The high job ID used for selection (if SSS2SJBI on). This value must be null or at

least as high as SSS2JBIL. When SSS2JBIH is 2 to 8 characters and starts with one of the prefixes J, JO, or JOB, then the suffix is converted to a binary value. Job IDs with a suffix within the range from the SSS2JBIL suffix through the SSS2JBIH suffix are returned. Generics characters * or ? are *not* allowed. Embedded and trailing blanks are acceptable. The maximum length of the job ID is eight characters.

When SSS2STPI is set, then EBCDIC characters A to Z; 0 to 9; and national characters @, #, $ are allowed. Job IDs, as returned in STTRJID, and transaction job IDs within the 1 to 8 character EBCDIC range from SSS2JBIL through SSS2JBIH, are returned. Generics characters * or ? are *not* allowed.

14.13.2 Extended status calls

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.

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 information.

Callers can request either terse or verbose information. Terse requests return less data but have lower overhead because no I/O is required. Verbose requests return more detailed data, but involve multiple I/O requests. For this reason, verbose requests are limited in how much data can be obtained in a single SSI invocation.

Extended status filtering for terse requests has also been enhanced to allow for filtering job and SYSOUT data based upon the transaction job ID, transaction name, or SYSOUT owner. The transaction job ID and name have been added to the SYSOUT terse data.

New extended status flags

There are three extended status options for terse requests that are new with z/OS V1R11in flag STATSSL4. STATSSL4 is used to support filtering of information that is returned based on transaction name, transaction job ID, or transaction owner.

The flag STATSSL4 bits are as follows:

STATSTPN Transaction job name filtering. If this bit is on, information about jobs and SYSOUT associated with a transaction job name that matches STATJOBN or STATJBNP is returned. The STATSTPN bit is ignored if one of the following situations occurs:

- STATSJBN is not set.
- Verbose information is requested.

STATSTPI Transaction job ID filtering. If STATSTPI is *not* set, then only jobs and SYSOUT that has a job ID in the range specified by STATJBIL and STATJBIH are returned. If STATSTPI *is* set, then jobs and SYSOUT associated with a SYSOUT data set with a transaction job ID are also selected. The job ID is in the range specified by STATJBIL and STATJBIH.

The STATSTPI bit is ignored if one of the following situations occurs:

- STATSJBI is not set.
- Verbose information is requested.

STATSTPU SYSOUT owner filtering. If this bit is on, then jobs and SYSOUT that are associated with a SYSOUT data set whose transaction owner matches STATOWNR are returned.

The STATSTPU bit is ignored if one of the following situations occurs:

- STATSOWN is not set.
- Verbose information is requested.

15

GRS enhancements

z/OS provides multiple ways of providing serialized access to data on a single or multiple systems, but global resource serialization is a fundamental way for programs to get the control they need and ensure the integrity of resources in a multisystem environment. Because global resource serialization is automatically part of a z/OS system and is present during z/OS initialization, it provides the application programming interfaces that are used by the applications on the system.

Several enhancements were implemented on each release to improve performance and to assist client in the problem determination.

z/OS V1R10 introduced the following enhancements:

- Performance monitoring enhancements
- GRSRNL=EXCLUDE migration to full RNLs
- GRS IPCS enhancements

z/OS V1R11 introduced the following enhancements:

- Provide latch identity services
- Enhance latch creation deadlock detection
- Provide ECA (enhanced contention analysis) for latches
- GRS Virtual Storage Constraint Relief extended to GQScan and ISGQUERY services
- Enhanced filtering support for ENF 51
- Enhanced dynamic GRS exits

15.1 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 ENQs to take various paths through non-monitor ENQ processing. This exposed errors in user code because various ENQ paths enforced or expected the exploiter to abide by certain rules whereas other paths were more forgiving. In z/OS V1R9 and later, 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.

15.1.1 New monitor performance enhancements

z/OS V1R10 reduces 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 make running the monitor more practical from the perspective of performance impact. However, note the following points:

- Although performance was dramatically improved in R10, monitor usage still increases the path length. The health check remains simply to make the monitor aware that it is on. Use the monitor when needed, but do not have it on unless an installation feels it is justified. According to *z/OS MVS Planning: Global Resource Serialization*, SA22-7600, you can measure the performance overhead by using the ISGEQRSP sample program.
- In previous releases, not having the monitor set to the correct dispatch priority (it needs to be SYSSTC) can cause system problems such that high priority ENQs are 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. Its CPU time, however, 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:

```
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.

You define the GRS monitor filter table 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 can remove the old data first by restarting the monitor with the new filter table, rather than simply 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, and that data can be misleading. If you do not want the old data in the reports, restart the monitor with the new filter table.

With this new option, REQTYPE=NCRESERVE specifies that only non-converted reserves (that is, 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 is to 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 benefits both RING and STAR mode, but not GRS=NONE images.

15.2 GRSRNL=EXCLUDE migration to full RNLs

GRS provided the ability to change RNLs dynamically. If a system was IPLed in GRSRNL=EXCLUDE mode, however, then a dynamic RNL change cannot be performed. Migrating from GRSRNL=EXCLUDE to full RNLs required a complex-wide sysplex outage reIPL.

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 though the RNLs were in place when the original ENQ requests were issued. This feature eliminates need for an 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 by using 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 points:

- 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 the user to confirm the use of the FORCE option:

```
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:

- Migration to full RNLs can only be issued in a single system GRS STAR sysplex with GRSRNL=EXCLUDE.
- No ISGNQXIT or ISGNQXITFAST exit routines, or any requests affected by one, can be active.
- 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 can result if requests were issued with RNL=NO.)
- No resources with a MASID request can exist where only several of the requesters will become global. (This can result if requests were issued with RNL=NO or with other 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 can 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 migrating to the specified RNLs, the only way to move back to GRSRNL=EXCLUDE is to perform 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.

15.3 GRS IPCS enhancements

With z/OS V1R10, coding changes have been introduced to increase performance.

- 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:
  - Event TODs show ENQ history for requests, contention, and ownership.
  - Altered by RNLs and exits, and third 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.

    QCBTRACE knows very little or nothing about the dumped system's ENQs that are related to waiters or blockers on other systems. It also 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).

15.3.1 New IPCS GRS panel

You reach the GRSDATA panel by using the following IPCS options:

1. Select **Option 2. Analysis** on the primary IPCS panel.
2. Then select **Option 6. Component**.
3. Type `S` to select **GRSDATA**, as shown in Figure 15-1.

   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 15-1 IPCS panel to select the new GRSDATA panel

New GRSDATA IPCS panel

New keywords can also be used by using keywords directly specified on the **IP GRSDATA** or **IP VERBX GRSTRACE** commands.

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 significantly more detail for ENQs from the locally dumped system.

Figure 15-2 on page 317 shows the new IPCS GRSDATA subcommand panel. (Note that invoking the **GRSDATA** or **VERBX GRSTRACE** commands directly will not present this panel.)

GRSDATA panel filter explanations

At least one requestor in a resource chain must match all of the filtering options for a resource to be returned.

SYSNAME This is the name of the system requesting the resource.

JOBNAME This is the name of the job requesting the resource.

ASID This is the address space number (in hex) requesting the resource.

TCB This is the TCB address of the requesting task.

QNAME This is the QNAME (Major Name) of the resource.

RNAME This is the 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 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.

Time filters

These time filters 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
```

Wildcard support

The rules for wildcard support are listed here:

***** Use an asterisk (*) for zero or more characters.

? Use a question (?) mark 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 15-2 New IPCS GRSDATA subcommand panel

15.4 Query outstanding related ENQs

With z/OS V1R10, to allow applications to query outstanding related ENQs by using 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 differing 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 the 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

The ISGQUERY macro supports the UERSDATA as an ISGQUERY specific or wildcard filter. This support for the ISGENQ and ISGQUERY macros is primarily for Independent Software Vendors (ISVs).

15.5 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. By using the HW reserve function, DASD volumes can be shared between 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.

The GRS latch services provide a high-speed serialization service for authorized callers only by using 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 include UNIX System Services, System Logger, RRS, MVS, and many others.

The GRS latch non-contention instruction path is on the order of a moderate number of instructions. 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 resolves 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 earlier 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 15-3 on page 319.

```
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 15-3 D GRS,LATCH,CONTENTION command output

Note: See *z/OS MVS Diagnosis: Reference*, GA22-7588 and component-specific documentation for additional information about what specific latch contention can mean and what steps to take for various circumstances.

15.6 GRS latch identity services

Up to z/OS V1R10, the latch services simply provided a number to identify each latch request. This made it difficult to identify the real resource used.

In z/OS V1R11, a new latch callable services (ISGLID) provides the facility to give an identity to each request. Latch exploiters must include this service in their program source to associate an identity to each latch request. This service, shown in Figure 15-4, uses 64-bit addressing.

```
Format:     CALL ISGLID(lsetToken,LIDArray, LIDEntryVersion,retcode)
Parameters: LSetToken input char(8) -  latch set token from ISGLCRT
            LIDPtrArray input char - Latch Identity Pointer Array
            LIDEntryVersion input fixed(8) – version of LID entries
            RetCode output fixed(32) - return code from the service
```

Figure 15-4 ISGLID callable service

RRS and LOGGER exploit this facility with z/OS V1R11.

The **D GRS,C** command, for compatibility reasons, continues to display the same information as in previous releases. To explore the latch identity, use the new analyze latch command with BLOCKER, or WAITER, or DEPENDENCY options as shown on Figure 15-5 on page 321.

15.7 GRS latch enhanced contention analysis

The **D GRS** command now has new options for latch contention analysis, the same ones as for ENQ (waiters/blocker/dependency). Because the command applies for both (ENQ/LATCH), you must specify which one you want to display, although ENQ is the default. Options of the **D GRS** command, shown in Figure 15-5 on page 321 for latch analysis, are listed here.

The new keywords included for latch contention analysis command have the following meanings:

cr-asid This is the ASID of the latch set creator space to analyze.

cr-jobname This is the JOBNAME of the latch set creator space to analyze.

Xlsetnamelist This is a list of latch sets to be excluded from the display, as follows:

- ? matches any single character and * matches any string of zero or more characters.
- 1 to 5 Lset names can be entered.

Xjobnamelist This is a list of jobnames to be excluded from the display.

- Generic jobnames can be entered by placing an asterisk (*) at the end.
- 1 to 25 jobnames can be entered.

Workunitaddr This is the address of the workunit of the latch requestor.

- TCB (or WEB for SRB).

Lsetname This is the name of the latch set to analyze.

Latchnum This is the latch number to analyze, specified by 1-8 decimal digits.

Jobname This is the JOBNAME of the latch requestor to analyze.

Asid This is the ASID of the latch requestor to analyze.

This command supports wildcard, which is useful.

Latch identity further improves the display output. Up to 255 characters of the printable string are displayed. Contention thresholds can help filter out “noise.” Latch analysis applies across all latch sets on a system.

```
D GRS
   {,{CONTENTION|C}[,ENQ|,E][,{LATCH|L}[,{JOBNAME|JOB}=jobname]][,HEX]}

   {,{LATCH|L}[,{JOBNAME|JOB}=jobname][,CONTENTION|,C]}[,HEX]
      {CONTENTION|C}}

   {{,ANALYZE|,ANALYSE|,AN}, LATCH, WAITER}
      {[,CASID=cr-asid]|[,CJOBNAME=cr-jobname]
      [,ASID=asid] | [,JOBNAME=jobname] | [,XJOBNAME=(xjobnamelist)]
      [,XLSETNM=(xlsetnamelist)]
      [,COUNT=nn]
      [,DETAIL]
      }

   {{,ANALYSE, LATCH, BLOCKER}
      {[,CASID=cr-asid]|[,CJOBNAME=cr-jobname]
      [,ASID=asid] | [,JOBNAME=jobname] | [,XJOBNAME=(xjobnamelist)]
      [,XLSETNM=(xlsetnamelist)]
      [,COUNT=nn]
      [,DETAIL]
      }

   {{,ANALYSE, LATCH, DEPENDENCY}
      [,ASID=asid] | [,JOBNAME=jobname] | [,XJOBNAME=(xjobnamelist)]
         [,TCB|WEB=workunitaddr]
      [,XLSETNM=(xlsetnamelist)]
      {[,CASID=cr-asid] | [,CJOBNAME=cr-jobname]
         [,LAT=(lsetname,latchnum)]
      [,COUNT=nn]
      [,DETAIL]
      }
```

Figure 15-5 D GRS command options for LATCH analysis

15.8 GRS latch create's deadlock detection

In releases prior to z/OS V1R11, if the system entered latch deadlock contention, then human intervention was required to cancel the unit of work that was causing the contention. Because each latch was identified by a number, determining which unit of work to cancel was difficult.

In z/OS V1R11, an interface defined in ISGLMASM and ISGLMC allows authorized applications to set thresholds in the LIDHOLDTHRESHOLD and LIDCONTTHRESHOLD keywords.

The LIDHOLDTHRESHOLD keyword defines the limit of time (in seconds) that this request will be considered normal.

The LIDCONTTHRESHOLD keyword defines the limit of time (in seconds) that this request can wait until it will be considered an exception.

The callable service ISGLCRT now has several create_option values:

- ISGLCRT_PRIVATE (OR A VALUE OF 0).

- ISGLCRT+PRIVATE + ISGLCRT_LOWSTGUSAGE (or a value of 2).
- ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET1 (or a value of 64).
- ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET2 (or a value of 128).
- ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET1 + ISGLCRT_LOWSTGUSAGE (or a value of 66).
- ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET2 + ISGLCRT_LOWSTGUSAGE (or a value of 130).

Note: ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET2 is the most useful create_option value.

Usage considerations

Use the ISGLCRT_LOWSTGUSAGE value only when virtual storage usage is already known to be a constraint.

The ISGLCRT_DEADLOCKDET2 value can result in the new ABEND9C6 reason:

- xxxx003A - A requestor called the Latch_Obtain service to obtain a latch, specifying an access_option of ISGLOBT_SHARED (value of 1). The latch has already been obtained by the same unit of work earlier with the ISGLOBT_SHARED access_option (value of 1) and another unit of work is waiting for the exclusive ownership on the same latch.

  Note: This reason code will only be returned if the ISGLCRT_DEADLOCKDET2 (value of 128) create_option is specified in ISGLCRT.

When ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET1 is specified, it can detect the following deadlock situations:

- The work unit requests exclusive ownership of a latch that the work unit already owns exclusively.
- The work unit requests shared ownership of a latch that the work unit already owns exclusively.

When ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET2 is specified, it can detect all the deadlock situations listed under ISGLCRT_PRIVATE + ISGLCRT_DEADLOCKDET1, and it can also detect:

- If the work unit holding a SHARED latch requests exclusive use of the same latch.
- If the work unit holding a SHARED latch requests it SHARED and another unit of work is waiting to obtain the latch EXCLUSIVE.

Because ISGLCRT_DEADLOCKDET2 provides the best deadlock detection, use ISGLCRT_DEADLOCKDET1 in cases where it can be used and use ISGLCRT_DEADLOCKDET2 in all cases where there are not many SHARED latch holders.

Note: The unit of work context of the requester is captured at latch obtain time. The system does not know if the application passes the responsibility for releasing the latch to another unit of work. To prevent false detection, dead lock detection cannot be used if latches are used in such a way that responsibility for releasing the latch is passed between the obtainer and the releaser.

Deadlock detection can be safely used by SRBs if all the obtained latches are released by the SRB work unit before the unit of work completes. There is a possibility of false deadlock hits otherwise.

Deadlock detection is not performed if the latches are obtained conditionally using the ISGLOBT_ASYNC_ECB option in ISGLOBT.

15.9 GRS enhanced filtering support for ENF 51

In addition to the functions that GRS provides to guarantee resource serialization as requested, it provides a function to advise other products that the contention serialization environment from a resource has changed using the ENFREQ services. There were requests to improve this function's performance for contention resolution.

In z/OS V1R11, GRS and ENF services created a new infrastructure where exploiters can pass filters of their requests. Using IEFREQ ACTION=LISTEN,**CODE=FLTRBLK**, the application can pass its filters and GRS can analyze its contention more effectively. The filters are always used for inclusion. Multiple filters can be used for a given listener registration, where the event will be presented when all the filters are satisfied. This new option keyword is currently only available for ENF 51 (the GRS event code). There are new ENFREQ return codes associated with this support, all for ACTION=LISTEN. They include:

- 60x Meaning: Program error. An ENF request specified FLTRBLK for an event code that does not support listener filter blocks.

  Action: Verify that the ENF request is for the correct event code. If so, do not specify FLTRBLK.
- 64x Meaning: Program error. An ENF request specified FLTRBLK. It was specified for an event code that does support listener filter blocks, but the block was not accessible by the owner of that particular event code.

  Action: Ensure that the event-specific listener filter block occupies accessible storage of sufficient length.
- 68x Meaning: Program error. An ENF request specified FLTRBLK. It was specified for an event code that supports listener filter blocks, and the block was accessible by the owner of that particular event code, but the filter parameters are incorrect.

  Action: Check the parameters specified in the FLTRBLK. If the event-specific mapping includes a reason code, use its value to assist with the problem determination.

The FLTRBLK filters includes the following:

- Owning sysname
- Owning ASID
- QName
- RName
- RName length

For additional information about this topic, see the following documentation:

- *z/OS MVS Authorized Assembler Service Reference EDT-IXG*, SA22-7610
- *z/OS MVS Data Areas, Volume 2 (DCCB-ITZYRETC)*, GA22-7582
- *z/OS MVS Planning: Global Resource Serialization*, SA22-7600

15.10 GRS enhancements for dynamic exits

In z/OS V1R9, all major internal GRS control blocks were moved above the bar. This created issues for other serialization products when extending their monitoring software through the available external interface.

In z/OS V1R11, the dynamic GRS exits are enhanced so that alternate serialization products can fully support the (ENQ, DEQ, RESERVE for both PCs and SVCs) interface in their resource monitoring.

Alternate serialization products can use the following GRS dynamic relevant exits (ISGCNFXITSYSTEM and ISGCNFXITSYSPLEX were updated and ISGNQXITQUEUED2 is completely new):

- The ISGNXITPREBATCH performance-oriented exit is used to determine if the ISGNQXITBATCHCND is to be called for a particular ENQ request.
- ISGNQXITBATCHCND for details of a particular ENQ request prior to GRS queuing it up.
- ISGCNFXITSYSTEM is a filtering exit for contention seen for scope=SYSTEM ENQs.
- ISGENDOFLQCB is for notification of the last requester of a resource DEQing.
- ISGCNFXITSYSPLEX is a filtering exit for contention seen for scope=SYSTEMS ENQs.
- ISGNQXITQUEUED1 is a follow-up from the batch exit where local resources have been queued.
- ISGNQXITQUEUED2 is a follow-up from the batch and queued2 exits where global resources have been queued.

ISV considerations

For additional information about this facility, ISVs can see *z/OS MVS Data Areas, Volume 2 (DCCB-ITZYRETC)*, GA22-7582, and *z/OS MVS Installation Exits*, SA22-7593.

16

z/OS UNIX System Services

The UNIX System Services element of z/OS is a UNIX operating environment that is 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:

- zFS file system sharing enhancements in z/OS V1R11
- UNIX System Services remount samemode support
- Using the ISPF editor for OEDIT and OBROWSE
- Alternate sysplex root support
- CINET Pre-router changes
- Automount enhancements
- Automatic UID and GID assignment
- dbx demand load

16.1 zFS file system sharing enhancements in z/OS V1R11

Before moving on to a description of the zFS file system sharing enhancements provided in z/OS V1R1, note that the following new terminology has been introduced:

Catch-up mount When a file system mount is successful on a system in a shared file system environment, z/OS UNIX automatically issues a corresponding local mount known as the catch-up mount to every other system's PFS that is running sysplex-aware for that mode (read-write or read-only).

If the corresponding local mount is successful, z/OS UNIX does not function ship from that system to the z/OS UNIX owning system when that file system is accessed. Rather, the file request is sent directly to the local PFS. This is sometimes referred to as Client=N.

If the corresponding local mount is unsuccessful (for instance, DASD is not accessible from that system), z/OS UNIX function ships requests to the z/OS UNIX owning system when that file system is accessed (message BPXF221I can be issued). This is sometimes referred to as Client=Y.

Local mount A local mount means that z/OS UNIX issues a successful mount to the local PFS. This is done when either the PFS is running sysplex-aware for that mode (read-write or read-only) or the system is the z/OS UNIX owner.

Sysplex-unaware A file system is sysplex-unaware if the PFS supporting that file system requires it to be accessed through the remote owning system from all other systems in a sysplex (allowing only one connection for update at a time) for a particular mode (read-only or read-write).

Access from a system not being the owner (sometimes referred to as being a client) is provided through XCF communication to the owning system (sometimes called "function shipping"). This is controlled by z/OS UNIX and the interface is also known as a PFS named XPFS (for Cross System PFS).

Sysplex-aware A PFS that is sysplex-aware for a particular mode (read-only or read-write) allows file requests for file systems that are mounted in that mode to be handled by the local PFS. Furthermore, the file systems are locally mounted on that system. z/OS UNIX does not function ship those file requests to the z/OS UNIX owning system.

zFS namespace Starting with z/OS V1R7, zFS has supported administration commands and APIs that are sysplex-wide in a shared file system environment. zFS communicates between sysplex members using XCF protocols.

The zFS XCF protocol exchanges information among members about zFS ownership and other attributes of zFS mounted file systems. This information, which is kept in the memory of each zFS member, is called the zFS namespace.

See "zSeries File System (zFS) overview" in *z/OS Distributed File Service zSeries File System Administration*, SC24-5989, for more details about terminology.

16.1.1 Main file system sharing concept prior to z/OS V1R11

Prior to z/OS V1R11, in a UNIX System Services file system sharing environment there is a concept of file system ownership, as illustrated in Figure 16-1. SC65 is the owning system of a read/write (R/W) and read/only (R/O) file system; that is, the sysplex member system that owns the file system is the first system that processes the mount. This system always accesses the read-write (R/W) file system locally; the other systems do not access the R/W file system directly. Each system can access the R/O file system directly.

However, when accessing a file system mounted as R/W, there is a disadvantage in having the access performed on a system that is not the file system owner. In this situation the request has to be addressed forward to the owning system.

Other non-owning systems in the sysplex, shown as SC64 and SC63 in the figure, access the R/W file system by issuing a function ship request to the owning system using XCF. Prior to z/OS V1R11, this support is sysplex-unaware.

Note: For file systems that are not sysplex-aware, the file system owner is the system to which z/OS UNIX forwards file requests. zFS must be running on all systems in the sysplex and all zFS file systems must be compatibility mode file systems (that is, they cannot be file systems in multi-file system aggregates beginning with z/OS V1R8).

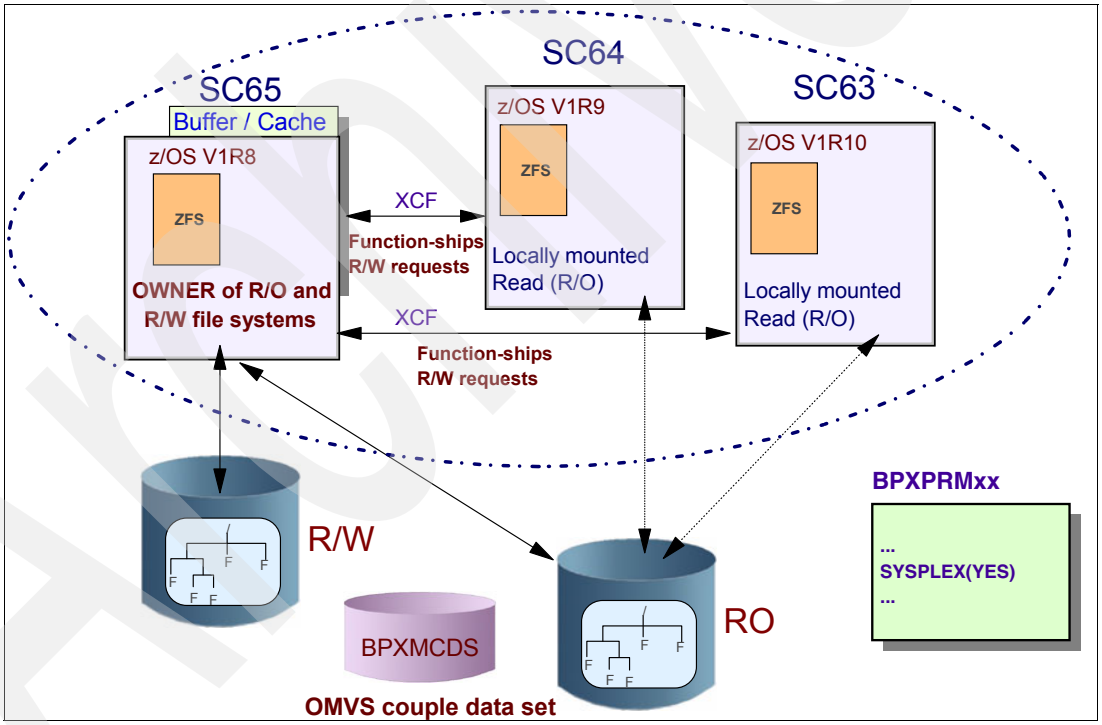


Figure 16-1 UNIX System Services file system sharing prior to z/OS V1R11

16.1.2 UNIX System Services file system sharing limitations for R/W file systems

UNIX System Services sysplex file system sharing provides many benefits as compared to single system mode:

- You can write to file systems from all systems in the sysplex.

- File systems for applications in the UNIX System Services sysplex environment are highly available.
- You can perform small changes to file systems used in read-only mode by switching to read-write mode temporarily, performing the change, and switching back to read-only mode.

However, it is not only the request that has to be routed using XCF; rather, all data to be written or read must use the function shipping path. This causes overhead, as compared to being on the owing system and having all buffer and cache management available locally and making all I/O requests directly.

File systems mounted as R/O have been sysplex-aware from the beginning, and HFS and zFS file systems mounted in R/W mode were sysplex-unaware in all releases previous to z/OS V1R11.

Note: The performance disadvantage on a client system was the same, independent of the request being a write or just a read processing.

16.1.3 zFS support in UNIX System Services sysplex sharing prior to z/OS V1R11

In z/OS V1R6 and older systems, zFS knows only about local zFS file systems. Figure 16-2 on page 329 illustrates this case, and the support is as follows:

- zFS is not sysplex-aware, except for read-only mounted file systems.
- UNIX System Services forwards file requests from a non-owning system (remote) to the owning system.
- zFS administration requests are handled locally.
- zFS is unaware of other remote zFS file systems and aggregate.

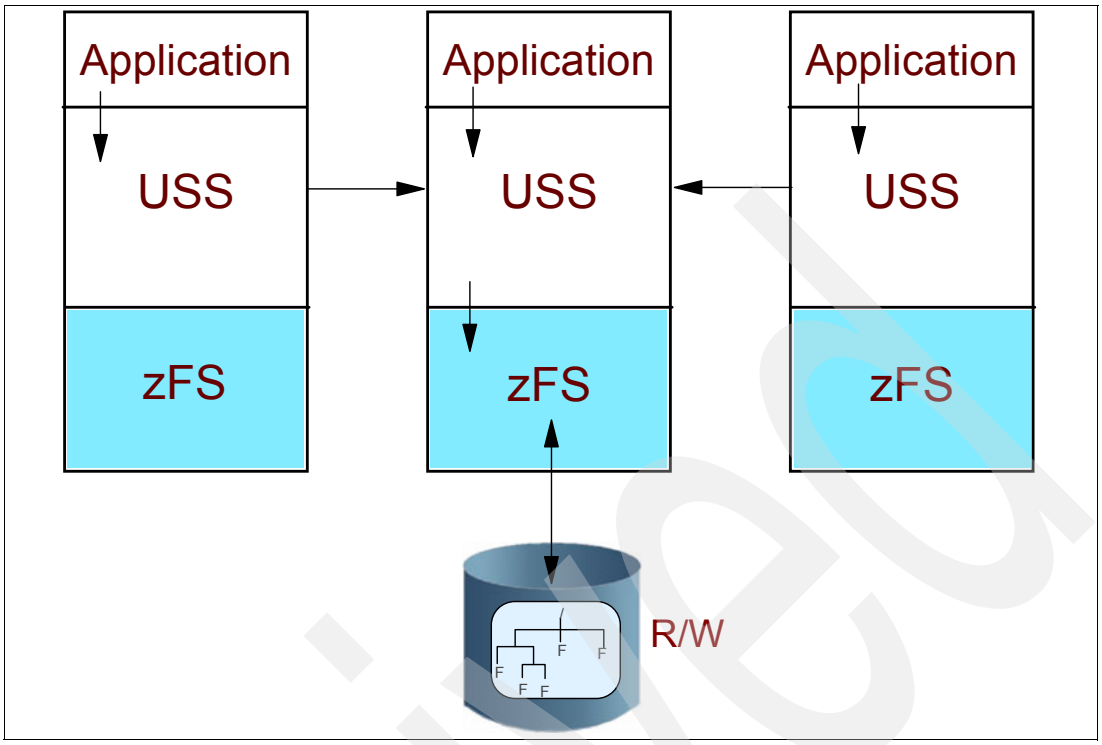


Figure 16-2 zFS support in z/OS V1R6 and earlier

z/OS V1R7 zFS support

In z/OS V1R7, the zFS administration interface works across the sysplex as shown in Figure 16-3 on page 330.

- zFS is sysplex-unaware (for file systems mounted read-write).
- UNIX System Services forwards file requests still to the z/OS UNIX owning system.
- zFS administration requests are handled across the sysplex.
- zFS knows about other remote zFS file systems and aggregates.

Requests are forwarded by UNIX System Services to the z/OS UNIX owning system, and then given to zFS. Responses are returned from zFS to UNIX System Services and then returned to the issuing system. As mentioned previously, this can cause a significant increase in the instruction pathlength as opposed to local access to a file system.

IOEZFS group specification

In addition, zFS support in z/OS V1R7 provided support for **`zfsadm`** commands that apply to zFS aggregates or file systems to work against all aggregates and file systems across the sysplex. The following **`zfsadm`** subcommands can optionally direct their operation to a particular member of the sysplex: aggrinfo, attach, clonesys, config, configquery, define, detach, format, lsaggr, lsfs, and query. This support is represented by the Admin in Figure 16-3 on page 330.

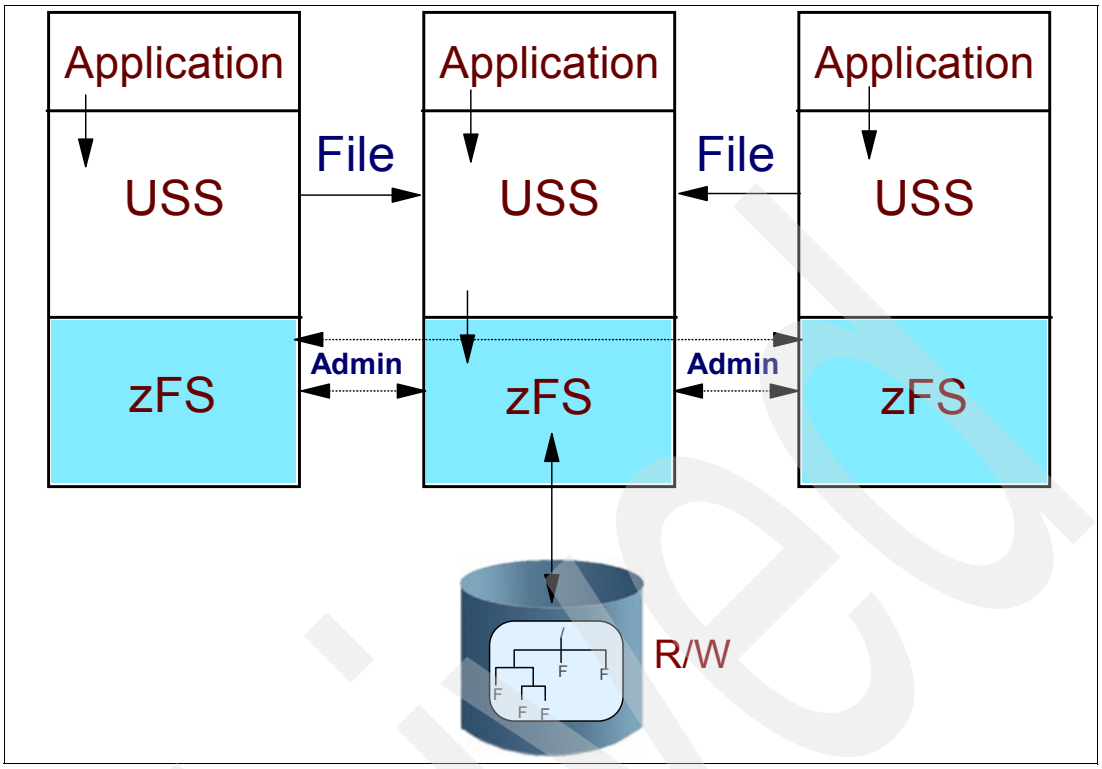


Figure 16-3 zFS support in z/OS V1R7 to z/OS V1R10

Note: This is exactly the same technique that is used for HFS file system sharing in a sysplex.

16.1.4 New and earlier zFS configuration options in z/OS V1R11

The following new configuration options can be defined either in the IOEPRMxx parmlib member or in the IOEFSPRM DD member in the ZFS procedure.

sysplex={ on | off }
The sysplex configuration option controls whether or not zFS runs sysplex-aware. The BPXPRMxx parmlib member option, SYSPLEX(YES), must be set to allow zFS to run sysplex-aware. See “zFS sysplex-aware in z/OS V1R11 for R/W mounts” on page 331.

group = ioezfs
The group configuration option specifies the group name that zFS uses for XCF communications. This option was implemented in z/OS V1R7. See “IOEZFS group specification” on page 329.

token_cache_size
This option specifies the maximum number of tokens in the server token manager cache to use for cache consistency between zFS members. The default value is double the number of vnode_cache_size.

file_threads = 40
The file_threads configuration option determines how many threads are in the pool to handle requests from remote clients.

client_cache_size=128M	The client_cache_size configuration option specifies the size of the user data cache for client access. The user_cache_size is used for the user data cache for server access.
client_reply_storage = 40M	The client_reply_storage configuration option specifies how much storage is used for sysplex server replies.
recovery_max_storage=256M	This option indicates the maximum amount of zFS address space storage to use for concurrent log recovery during multiple concurrent aggregate mounts.

16.1.5 zFS sysplex-aware in z/OS V1R11 for R/W mounts

In z/OS V1R11, zFS is sysplex-aware for file systems mounted read-write. This means that UNIX System Services sends all file requests directly down to the local zFS physical file system (PFS). It does not function ship the requests. The local zFS either sends the request to the owning zFS system, or it satisfies the request from the local zFS cache. In many cases, this improves the pathlength over the function shipping model. The request and data flow is shown in Figure 16-4 on page 332.

With z/OS V1R11, the zFS PFS allows a file system to be locally accessed on all systems in a sysplex for a particular mode, and then the PFS is sysplex-aware for that mode. Therefore, performance can be improved for zFS in the UNIX System Services sysplex shared file system mode by zFS becoming sysplex-aware for R/W mounted file systems. This is a new specification that is specified with a new option as shown here:

```
sysplex=on
```

This zFS sysplex file system support improves the response time for zFS file system requests, as explained here:

- zFS becomes sysplex-aware for zFS read-write file systems.
- zFS uses less XCF communications, as a new client caching is introduced.
- This answers the client requirement MR1211023911 asking for improved performance in a UNIX System Services sysplex file system sharing environment.

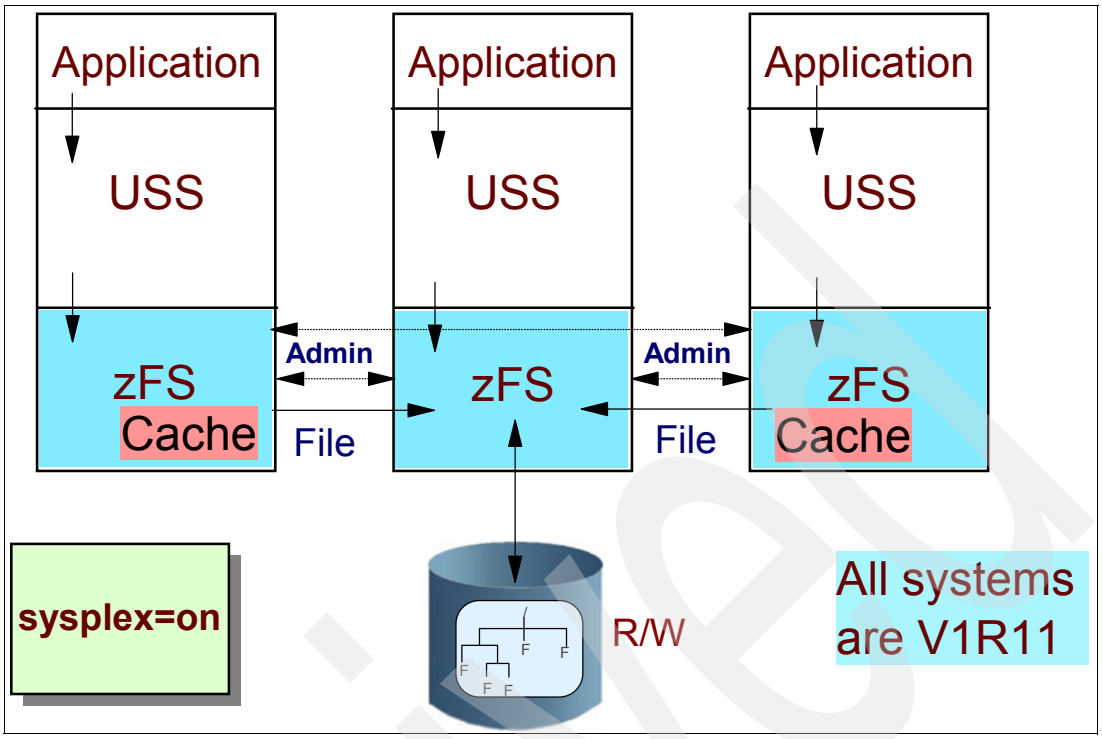


Figure 16-4 zFS sysplex support in z/OS V1R11

Defining zFS to run sysplex-aware

When the BPXPRMxx parmlib member specifies **SYSPLEX(YES)**, you define sysplex-aware beginning for z/OS V1R11 systems by using *one* of the following ways to specify zFS options.

- Use the IOEPRMxx parmlib member to contain the configuration options that control the zFS environment and the zFS aggregates to be attached at the startup of zFS. The IOEPRMxx files are contained in the logical parmlib concatenation. The parameter to use is **sysplex=on**.

  Using this option means that zFS runs as sysplex-aware and that z/OS UNIX does not forward file requests to the z/OS UNIX owning system.

- If you still use the IOEZPRM DD statement in the ZFS PROC, then specify the zFS configuration option in the IOEFSPRM file, which is pointed to by the DD name. (Note, however, that this way of setting the zFS parameters is not considered a good practice.)

zFS cache management

When the zFS PFS is sysplex-aware, z/OS UNIX now sends requests to the local zFS and zFS takes on the responsibility of forwarding the request, if necessary. For many requests, it is not necessary because zFS caches data locally and can satisfy the request from its cache. Cache consistency is maintained through a token management mechanism. This can improve performance in a shared file system environment.

Note: With z/OS V1R11, zFS mounted file systems can be sysplex-aware because zFS has it own concept of file system ownership.

The local zFS PFS that receives a request forwards the request to the zFS file system owner, if necessary. zFS cache consistency is maintained by using a token management mechanism. A zFS client caches data as long as it holds a token for the data. Clients request tokens from the token manager.

Token management for an aggregate runs on the zFS owning system. This administration task is handled by the token manager for an aggregate, and it handles token requests and manages token conflicts. It revokes conflicting tokens from clients.

Important: The new support in zFS has implemented a control mechanism for all files and directories in a file system (with a possible granularity of 64 KB blocks of data, if needed). This so-called “token management” is controlled by zFS on the system that is the zFS owner. It allows that zFS on a specific system has full control according to the type of token that it owns, either for reads or writes or even for both. This is true as long as the system owns the token or both tokens.

In addition to using caching to improve performance, the zFS owning system also keeps track of the number of requests it receives from each client (and local) system. Based on a specific calculation and at internally set intervals, zFS can move zFS ownership to attempt to optimize zFS aggregate ownership to the system making the most requests.

On a system failure, zFS will move zFS ownership to another system.

zFS cache processing

When a request is received by the local zFS PFS, zFS determines if the request can be satisfied from its cache. If it is a read, lookup, or readdir and the data is contained in the cache, the request can be satisfied without communication with the zFS owning system. Write requests are written forwarded to the owner, unless you know that the space is already allocated on disk.

With this new support in z/OS V1R11, the following conditions exist:

- zFS is sysplex-aware.
- UNIX System Services sends requests directly to local zFS PFS, regardless of the UNIX System Services owner.
- zFS decides whether it needs to forward the request to the zFS owning system.
- zFS caching and tokens allow it to sometimes avoid XCF communications.
- zFS moves zFS aggregates on system failure and to minimize XCF processing.
- Only compatibility mode zFS aggregates are supported.

New support details

When all systems are running at the z/OS V1R11 level, only compatibility mode aggregates are supported in a sysplex. Starting with z/OS V1R8, a zFS file system that resides in a multi-file system aggregate cannot be mounted in a sysplex sharing environment.

The new options can be set or queried using the **zfsadm** command interface:

- The **zfsadm config** command allows setting all the new configuration options except for **sysplex=on|off**.
- The **zfsadm configquery** command can show all the new configuration options and also the new additional value for sysplex_state (option 2):

  0 This indicates that zFS is not in a shared file system environment (this is normal for V1R6 and prior releases and for single system configurations including monoplex and xcflocal).

  1 This indicates that zFS is in a shared file system environment (this is normal for z/OS V1R7 to z/OS V1R10).

2 This indicates that zFS is running in a sysplex-aware environment (this is normal for V1R11 in a shared file system environment where zFS is running sysplex-aware for read-write file systems). Note the following example:

```
ROGERS @ SC74:/u/rogers>zfsadm configquery -sysplex_state
IOEZ00317I The value for configuration option -sysplex_state is 2.
```

Defining zFS to run sysplex-unaware in z/OS V1R11

When all systems are running at z/OS V1R11, you can define the zFS sysplex setting as **sysplex=off** and perform sysplex file sharing as in all previous releases; see Figure 16-5.

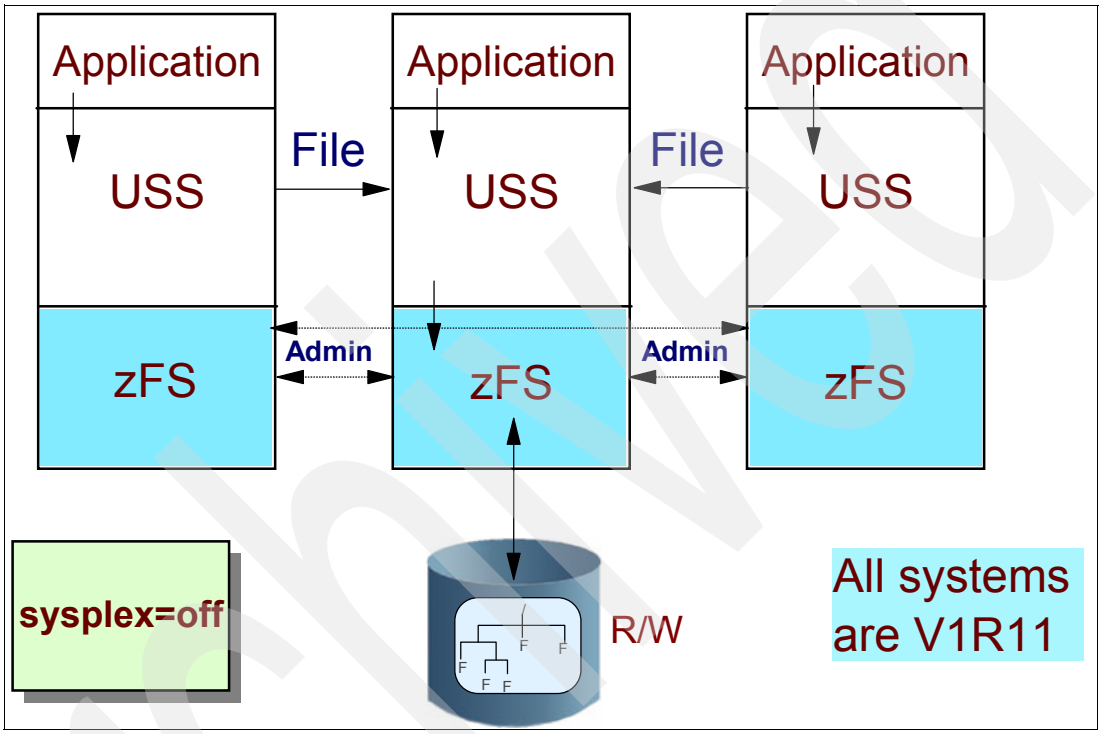


Figure 16-5 z/OS V1R11 sysplex running zFS with sysplex=off

Normal migration path to zFS being sysplex-aware

The normal migration path consists of the following steps:

1. zFS in z/OS V1R11 is running with option sysplex=off as sysplex-unaware for read-write mounted file systems. This is the default.
2. Then zFS in R11 is started with sysplex=on as sysplex-aware through a rolling IPL or by carefully stopping and restarting zFS.

zFS sysplex-aware restrictions

The following restrictions apply to zFS running sysplex-aware for read-write mounted file systems:

- The z/OS SMB server cannot export zFS sysplex-aware read-write file systems.
- The fast response cache accelerator support of the IBM HTTP Server V5R3 for z/OS cannot cache files contained in zFS sysplex-aware file systems.
- The z/OS UNIX ownership of a zFS aggregate mounted sysplex-aware cannot be moved to a prior release or z/OS V1R11 zFS being sysplex-unaware; see Figure 16-28 on page 350.

16.1.6 Read-only mounted file systems (sysplex-aware)

When a file system is mounted read-only, as shown in Figure 16-6, the mount request is sent to the local physical file system (in this case, zFS). zFS opens the file system data set (for read).

If the mount is successful on that system, z/OS UNIX records the mount and sends a signal to the other sysplex member systems to issue a catch-up mount on each system. Each z/OS UNIX on each other system then reads the couple data set (CDS) and determines that it needs to send a mount request to the local zFS for that file system. Each “local mount” causes zFS to open the data set (for read). In this way, the mount on SC65 causes the file system to be mounted on every member of the sysplex.

For read-only mounted file systems, file requests are sent directly to the local physical file system, which directly reads the file system data on DASD (as shown in Figure 16-6). That means each zFS on each system has the zFS file system opened (for read) and directly accesses the data.

Note: Read-only mounted file systems are referred to as being sysplex-aware.

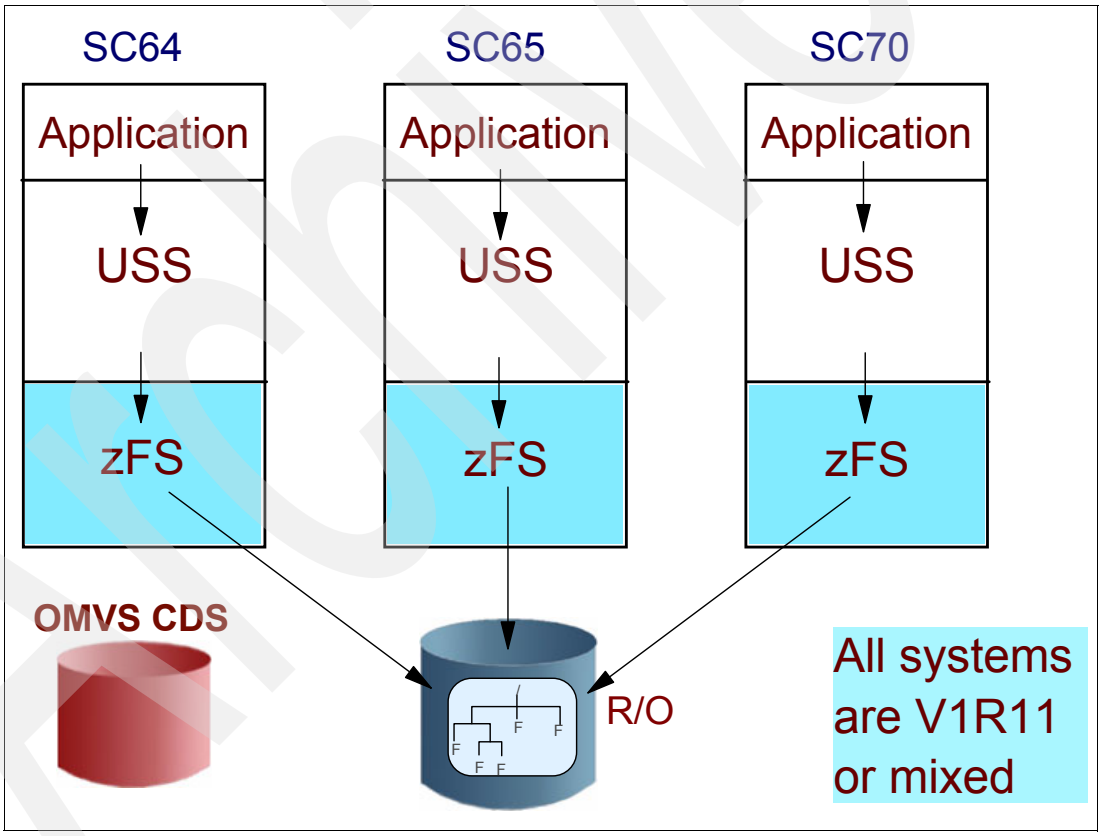


Figure 16-6 Sysplex-aware file system (read-only)

16.1.7 z/OS UNIX file system ownership versus zFS aggregate ownership

With z/OS V1R11 there are now two types of file system owners for zFS aggregates:

- z/OS UNIX file system ownership
- zFS aggregate ownership

We discuss these types in more detail in the following sections.

z/OS UNIX file system ownership

z/OS UNIX ownership of file systems determines where z/OS UNIX will forward requests for sysplex-unaware file systems.

However, when zFS runs sysplex-aware, z/OS UNIX ownership does not have the same significance for zFS file systems as it does for HFS file systems because z/OS UNIX does not normally function ship requests for zFS file systems in this situation. UNIX System Services still chooses a z/OS UNIX owning system for each zFS file system, but that does not control where local file requests are forwarded to by z/OS UNIX. File requests to sysplex-aware file systems are sent directly to the local PFS.

z/OS UNIX ownership does, however, control to which distributed Byte Range Lock Manager that byte range lock requests are forwarded.

The z/OS UNIX file system owner can be other than the zFS file system owner. z/OS UNIX file system ownership can be determined by using the **df -v** command, as shown in Figure 16-7.

```
/pp/db2v9/batch17/db2910_base: >df -v .
Mounted on    Filesystem          Avail/Total   Files    Status
/pp/db2v9/batch17/db2910_base (OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS) 6884/21600
4294967272 Available
ZFS, Read/Write, Device:60, ACLS=Y
File System Owner : SC63      Automove=Y    Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS
```

Figure 16-7 df -v command for a single file system

zFS aggregate ownership

zFS aggregate ownership determines where zFS forwards requests to where zFS is running sysplex-aware.

zFS can move the zFS ownership of an aggregate in the following cases:

- Based on usage of the zFS aggregate in the systems of the sharing environment
- When the owning system fails or is being shut down
- When a stop or failure of zFS occurs

As a result, the zFS owner of the aggregate can be other than the z/OS UNIX owner of the file system. Both ownerships can change independently. The zFS aggregate owner coordinates I/O requests as well as actions such as quiesce, unquiesce, and unmount.

Note that even zFS aggregate ownership is not as critical as z/OS UNIX file system ownership was in previous releases due to zFS client caching and the possibility of dynamic aggregate movement.

zFS aggregate ownership can be determined by using the **zfsadm lsaggr** command., as shown in Figure 16-8.

```
zfsadm lsaggr
OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS                SC63      R/W
OMVS.DB2V9.SDSNWORF.BATCH17.ZFS                SC63      R/W
OMVS.DARIO.HFS                                 SC70      R/W
OMVS.HERING.TESTBAT                            SC63      R/W
OMVS.HERING.TEST.HFS.TAV                       SC70      R/W
OMVS.TEST.MULTIFS.ZFS                          SC63      R/O
ZFSFR.ZFSE.ZFS                                 SC64      R/W
```

Figure 16-8 zfsadm lsaggr command that displays the zFS file system owner

Note: In the examples shown in Figure 16-7 on page 336 and Figure 16-8 on page 337, the z/OS UNIX and zFS ownership for the file system is the same.

Displaying file owner information

Both z/OS UNIX file system ownership and zFS aggregate ownership are initially determined during mount processing. When a file system is mounted, a z/OS UNIX file system owner is assigned by z/OS UNIX. Initially, for a compatibility mode aggregate, the zFS owner of the aggregate and the z/OS UNIX owner of the file system in the aggregate are the same.

Thereafter, they can move independently. Figure 16-9 shows an example listing both ownerships for a specific zFS aggregate.

awk information: An instruction of the form: pattern {actions} indicates that awk is to perform the given set of actions on every record that meets a certain set of conditions. The conditions are given by the pattern part of the instruction.

In Figure 16-9 on page 337, the pattern of an instruction often looks for records that have a particular value in a field. The notation $5 stands for the fifth field of a record, $2 stands for the second field, and so on.

```
$> /usr/sbin/mount -qv test
----A- OMVS.HERING.TEST.ZFS /u/hering/test
$> df -v test | grep Owner | awk '{print "USS owner: "$5}'
USS owner: SC70
$> zfsadm lsaggr | grep OMVS.HERING.TEST.ZFS | awk '{print "zFS owner: "$2}'
zFS owner: SC65
$>
```

Figure 16-9 Listing the z/OS UNIX file system and the zFS aggregate owner

Note: You can be less concerned about z/OS UNIX ownership when zFS runs sysplex-aware because it does not control request forwarding. You can also be less concerned about zFS ownership because of caching and because zFS will try to be intelligent about automatically moving zFS ownership.

The new zFS functions can make it easier for users working in separate systems because the user file system follows the user automatically in case of sysplex-aware zFS.

Utilities for displaying ownership information

In addition to the standard commands and functions, in our case we created two REXX routines to display owner and owner-related information about UNIX System Services file systems and zFS aggregates:

RXDOWNER Display OWNER information

ZFSOWNER zFS aggregate OWNER information

Note: See *z/OS Distributed File Service zSeries File System Implementation z/OS V1R11,* SG24-6580, for a detailed description of the procedures and the contents of the common procedure.

Figure 16-10 lists corresponding information for the aggregate referenced in Figure 16-9 on page 337.

```
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

$> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
$>
```

Figure 16-10 Listing owner information with rxdowner and zfsowner

In addition to zFS aggregates, the rxdowner routine also lists information for other types of file systems; see Figure 16-11 on page 338.

```
$> rxdowner -d /etc

MP Directory : /SC70/etc
File System  : HFS.SC70.ETC
PFS Type     : HFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
$> rxdowner -f *AMD/u

MP Directory : /u
File System  : *AMD/u
PFS Type     : AUTOMNT
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC64     - File System read-only=N
$>
```

Figure 16-11 Listing owner information with rxdowner for other PFS types

16.1.8 zFS Admin support in z/OS V1R11

To avoid a possible multi-system outage due to a zFS namespace inconsistency that had been seen with an earlier implementation of the enhanced zFS sysplex file system support, the following enhancements have been added:

- The zFS XCF Admin protocol (for mount and unmount processing and zfsadm commands) has been simplified. Only information about locally mounted file systems is exchanged between sysplex members.
- The zFS file system ownership is kept externally in GRS enqueues.
- An automatic validation and if needed, a correction of the zFS namespace, is provided.

Toleration support for zFS in z/OS V1R11

Table 16-1 shows the PTF information for the necessary toleration support of zFS in z/OS V1R11 for systems z/OS V1R9 and z/OS V1R11.

Table 16-1 Toleration support for zFS Admin protocol in z/OS V1R9 and V1R10

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA45616	OA25026
z/OS V1R10 (3A0)	UA45614	OA25026

z/OS V1R9 and z/OS V1R10 with this toleration support added are the only z/OS releases that are supported together with zFS of z/OS V1R11.

zFS sysplex Admin levels

The following zFS Admin levels are available.

0 This is the zFS Admin interface protocol that is used for zFS running on systems z/OS V1R9 or V1R10 without APAR OA25026. This is the same as for z/OS levels z/OS V1R7 and V1R8.

1 This is the default zFS Admin level used in z/OS V1R9 and V1R10 when APAR OA25026 is applied, also called the conditioning level.

2 This the zFS Admin level for zFS of z/OS V1R9 and z/OS V1R10 that allows it to run together with zFS of z/OS V1R11. Therefore, it is called the toleration mode. It also exploits the benefits of the new interface.

3 This is the zFS Admin level of z/OS V1R11. Any specification of parameter sysplex_admin_level in z/OS V1R11 is ignored because zFS in z/OS V1R11 always runs at level 3.

Note: The default value is 3 in z/OS V1R11. The default value is 1 in z/OS V1R9 and V1R10. In z/OS V1R11, this option is ignored. In z/OS V1R9 and V1R10, the options are either 1 or 2. When the sysplex contains a z/OS V1R11 system, any other coexistence system must be at level 2.

Example: sysplex_admin_level=2

With the three z/OS levels at z/OS V1R9 plus PTF UA45616, z/OS V1R10 plus PTF UA45614, and V1R11, you need to understand the following situations.

- zFS in z/OS V1R9 or z/OS V1R10 with **`sysplex_admin_level=1`** set uses and follows the XCF Admin protocol among systems with level 0 and 1. It and does not initialize if a system with XCF Admin level 3 is active in the sysplex.

- zFS in z/OS V1R9 or z/OS V1R10 with **sysplex_admin_level=2** set uses and follows the new XCF Admin protocol among systems with levels 2 or 3. It does not initialize if a system with XCF Admin level 0 is active in the sysplex.
- zFS in z/OS V1R11 running with **sysplex_admin_level=3** by default uses and follows the new XCF Admin protocol among systems with levels 2 or 3. It does not initialize if a system with XCF Admin level 0 or 1 is active in the sysplex.

  It supports zFS namespace validation and correction among level 3 systems.

zFS namespace validation

In a shared file system environment, zFS in z/OS V1R11 verifies the zFS namespace on the following occasions:

- When an administration command experiences an XCF message timeout
- At zFS initialization
- When an inconsistency is detected
- When the operator **MODIFY ZFS,NSVALIDATE** command is issued, for example:

```
F ZFS,NSVALIDATE
IOEZ00025I zFS kernel: MODIFY command - NSVALIDATE completed successfully.
```

If a zFS namespace problem is found, zFS runs zFS namespace correction to resolve the problem. In a worst case this can cause a zFS restart, which is disruptive. However, this does not normally occur.

Because zFS file system ownership is kept externally in GRS enqueues, you can use the MVS system command **D GRS,RES=(SYSZIOEZ,*)** to display all zFS owners for all the zFS aggregates in your sysplex, if the owning system uses zFS Admin level 3. Figure 16-29 on page 350 shows an example for a specific aggregate.

Activating toleration of R11 zFS in z/OS V1R9 or z/OS V1R10

Activating toleration in this case is a two-step process.

1. Install APAR OA25026 on all z/OS V1R9 and V1R10 systems.
2. After OA25026 is active on all systems, specify **sysplex_admin_level=2** in the zFS parameter file for all systems and activate sysplex Admin level 2 on all systems. zFS of z/OS V1R11 can enter the sysplex at this point.

16.1.9 zfsadm commands in z/OS V1R11

This section explains the changes and new functions of the **zfsadm** command interface.

- The **zfsadm attach** command can no longer attach multi-file system aggregates in a UNIX System Services file system sharing environment.

  Important: In a z/OS V1R11 UNIX System Services file system sharing environment, multi-file system aggregates can no longer be attached.

  This is a further step in the removal of multi-file system aggregates.
- The **zfsadm config** command supports all new configuration options except setting the parameters **sysplex** and **sysplex_admin_level**. These options cannot be set or changed dynamically.
- The **zfsadm configquery** command displays the sysplex state with the already existing option **-sysplex_state**.

- Using the **zfsadm configquery -syslevel** command displays information about the PFS level similar to the MVS system command **MODIFY ZFS,QUERY,LEVEL**; see Figure 16-12.

UNIX zfsadm command:

```
$> zfsadm configquery -syslevel
IOEZ00644I The value for configuration option -syslevel is:
zFS kernel: z/OS    zSeries File System
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
sysplex(file) interface(3)
$>
```

MVS system command:

```
F ZFS,QUERY,LEVEL
IOEZ00639I zFS kernel: z/OS    zSeries File System 836
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
sysplex(file) interface(3)
IOEZ00025I zFS kernel: MODIFY command - QUERY,LEVEL completed successfully.
```

Figure 16-12 Showing information about PFS and sysplex interface level in z/OS V1R11

Figure 16-13 on page 341 shows corresponding information for zFS in z/OS V1R9 or z/OS V1R10 with toleration support APAR OA25026 applied.

```
F ZFS,QUERY,LEVEL
IOEZ00639I zFS kernel: z/OS    zSeries File System 831
Version 01.10.00 Service Level OA27728 - HZFS3A0.
Created on Mon Feb  2 12:09:18 EST 2009.
sysplex(admin-only) interface(2)
IOEZ00025I zFS kernel: MODIFY command - QUERY,LEVEL completed successfully.
```

Figure 16-13 Showing information about PFS and sysplex interface level in z/OS V1R10

zFS kernel messages during initialization

Similar information is provided when zFS is starting; see Figure 16-14.

zFS in z/OS V1R10 with sysplex_admin_level=2

```
IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
Version 01.10.00 Service Level OA27728 - HZFS3A0.
Created on Mon Feb  2 12:09:18 EST 2009.
Address space asid x7B
IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
IOEZ00617I zFS is running sysplex admin-only with interface level 2
IOEZ00350I Successfully joined group IOEZFS
IOEZ00055I zFS kernel: initialization complete.
```

zFS in z/OS V1R11 with sysplex=on

```
IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
Address space asid x6E
IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
IOEZ00617I zFS is running sysplex file-support with interface level 3
IOEZ00350I Successfully joined group IOEZFS
IOEZ00055I zFS kernel: initialization complete.
```

Figure 16-14 zFS initialization messages

16.2 zFS sysplex migration scenarios

When migrating to z/OS V1R11, there are special considerations for the new support when previous levels of z/OS are in the same sysplex with the V1R11 system. This is an important change with zFS when accessing, opening, and working with aggregates in z/OS V1R11.

Important: When zFS handles a mount request prior to V1R11 (or when V1R11 runs sysplex=off), zFS allocates the aggregate exclusive and then opens the aggregate for read-write.

When zFS handles a mount request for R11 running sysplex-aware (sysplex=on), zFS allocates the aggregate shared and then opens the aggregate for read-write on each V1R11 system in the sysplex.

This is why administrators cannot force a reopen of an aggregate by moving it to another system. However, the UNIX unmount (remount) samemode function can used to perform this task. See 16.2.6, “Exploiting UNIX unmount remount samemode” on page 353 for information about how to exploit this for zFS processing.

16.2.1 zFS migration support for z/OS V1R11

Two zFS migration health check routines, available through APAR OA27198, verify the readiness of your environment to migrate to zFS of z/OS V1R11 and warn about another step in the removal of zFS multi-file system aggregates; see Table 16-2.

Table 16-2 Health checks for migration to zFS in z/OS V1R11

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA49074	OA27198

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R10 (3A0)	UA49072	OA27198

Health check ZOSMIGV1R11_ZFS_INTERFACELEVEL

This health check queries the zFS interface level and verifies that **sysplex_admin_level=2** is specified. Otherwise no z/OS V1R11 zFS can initialize; see Figure 16-15.

```
 IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
...
 Version 01.11.00 Service Level z110117 - HZFS3B0.
 Created on Sun Mar 22 17:07:39 EDT 2009.
 Address space asid x3A
 IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
 IOEZ00617I zFS is running sysplex admin-only with interface level 3
 IOEZ00614A zFS has detected an incompatible interface level 1 for member SC63.
 IOEZ00057I zFS kernel program IOEFSCM is ending
*108 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO
 RESTART. REPLY 'I' TO IGNORE.
```

Figure 16-15 zFS of z/OS V1R11 not starting because of incompatible Admin level

Health check ZOSMIGREC_ZFS_RM_MULTIFS

In a z/OS V1R11 UNIX System Services file system sharing environment, multi-file system aggregates cannot be attached. This health check queries aggregates that are attached (not mounted) and reports them as shown in Figure 16-16 on page 344.

```
CHECK(IBMZFS,ZOSMIGREC_ZFS_RM_MULTIFS)
START TIME: 05/07/2009 15:06:50.136149
CHECK DATE: 20071231  CHECK SEVERITY: LOW

IOEZH0022I Attached multi-file system aggregates:
AGGREGATE                                      # OF FILE SYSTEMS
-----------------------------------------      -----------------
OMVS.TEST.MULTIFS.ZFS                                2

* Low Severity Exception *

IOEZH0021E Multi-file system aggregates are attached.

  Explanation:  Multi-file system aggregates are attached to this system.
    In zFS V1R11, multi-file systems cannot be attached in a sysplex environment.
...
    All multi-file system aggregates should be converted to compatibility mode
    aggregates.
...
  System Programmer Response:  Convert your multi-file system aggregates
    to compatibility mode aggregates.

    Remove all define_aggr configuration options in your IOEFSPRM file
    and run zfsadm detach -all.
...
  Check Reason:  Verifies no multi-file system aggregates attached.

END TIME: 05/07/2009 15:06:50.646808  STATUS: EXCEPTION-LOW
```

Figure 16-16 ZOSMIGREC_ZFS_RM_MULTIFS showing a multi-file system aggregate

If no multi-file system aggregate is remaining, the check runs successfully as shown in Figure 16-17.

```
CHECK(IBMZFS,ZOSMIGREC_ZFS_RM_MULTIFS)
START TIME: 05/07/2009 12:11:15.935653
CHECK DATE: 20071231  CHECK SEVERITY: LOW

IOEZH0020I No multi-file system aggregates are attached.
This check ran successfully and found no exceptions.

END TIME: 05/07/2009 15:11:16.197562  STATUS: SUCCESSFUL
```

Figure 16-17 ZOSMIGREC_ZFS_RM_MULTIFS running successfully

Note: In any environment, it is good practice to migrate off multi-file system aggregates.

16.2.2 zFS aggregate mounted on a system prior to z/OS V1R11

z/OS V1R9 and z/OS V1R10 are the only systems that can be in the same sysplex as z/OS V1R11. The first example shows a situation when an aggregate is mounted on a system prior to z/OS V1R11; see Figure 16-19 on page 345.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC64: z/OS Version 01.10.00
#> /usr/sbin/mount -t zfs -f OMVS.HERING.TEST.ZFS /u/hering/test
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#>
```

Figure 16-18 Mounting zFS aggregate on system SC64 running z/OS V1R10

When a primary zFS file system mount occurs on a system and it becomes the z/OS UNIX owner, UNIX System Services issues catch-up mounts on the other sysplex members (because every system needs to know about all file systems in the sharing environment).

- These catch-up mounts go to the local zFS when that system runs sysplex-aware.
- For PFSs that are not sysplex-aware, the catch-up mounts go to the z/OS UNIX internal Cross System PFS (XPFS) for function shipping.

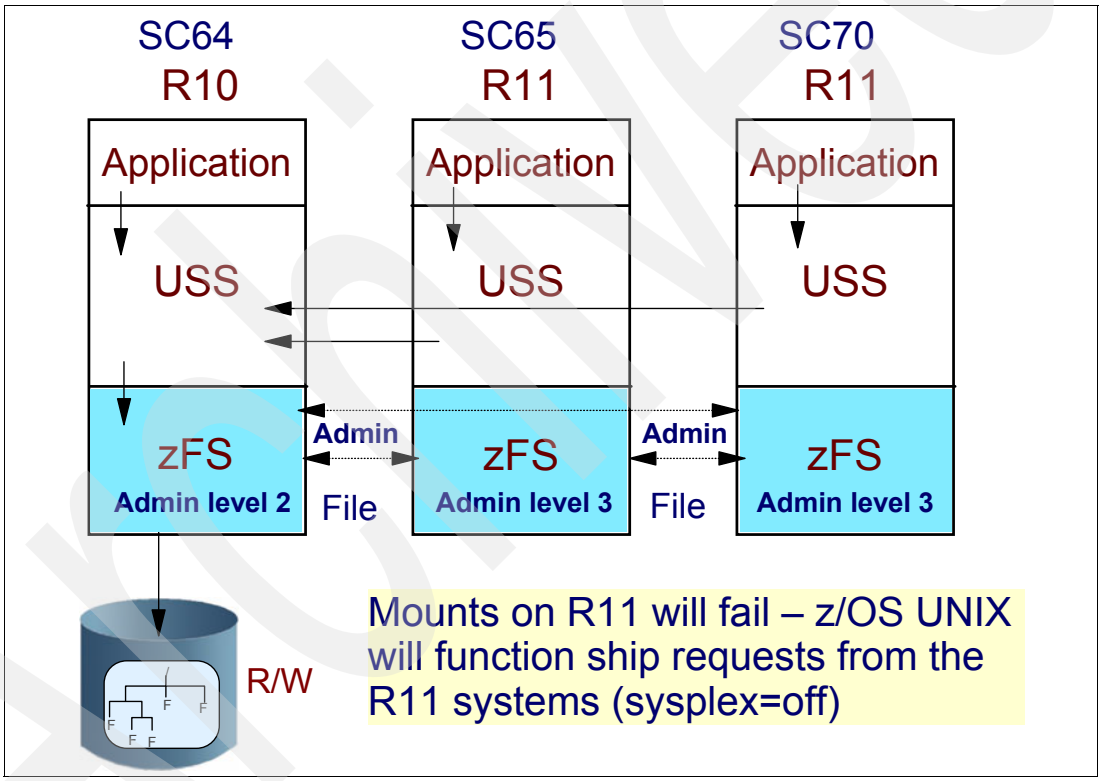


Figure 16-19 zFS aggregate mounted on a system prior to z/OS V1R11

The other systems in this sysplex, SC65 and SC70, are running z/OS V1R11 and get the messages shown in Figure 16-20 on page 346 on these systems.

```
SC65:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 076
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
SC70:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 185
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
```

Figure 16-20 Catch-up zFS mount error messages on z/OS V1R11 systems

zFS cannot allocate the file system shared on the z/OS V1R11 systems because the zFS on the earlier z/OS release has allocated the file system exclusively. This is shown in Figure 16-21.

```
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 17.33.00 GRS STATUS 597
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                  0030       007FD0C0 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                  0030       007EAE88 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                  0030       007EAE88 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                  0030       007EAE88   SHARE      OWN
```

Figure 16-21 zFS aggregate allocated exclusively on SC64 running z/OS V1R8

Therefore, z/OS UNIX will function ship zFS requests from the z/OS V1R11 systems to the owning z/OS V1R10 system in this situation.

16.2.3 zFS aggregate mounted on z/OS V1R11 system (sysplex=off)

Figure 16-24 on page 347 shows a mixed release sysplex. When a mount is issued on an R11 system, and there are other systems running pre-R11 zFS (or a R11 zFS with sysplex=off), the catch-up mounts on the pre-R11 zFS systems go to the z/OS UNIX internal XPFS PFS for function shipping. No error messages are issued.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC65: z/OS Version 01.11.00
#> /usr/sbin/mount -t zfs -f OMVS.HERING.TEST.ZFS /u/hering/test
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#>
```

Figure 16-22 Mounting zFS aggregate on system SC65 running z/OS V1R11

A message as shown in Figure 16-23 is issued on each z/OS V1R11 system that is running zFS sysplex-aware.

```
SC70:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 185
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
```

Figure 16-23 Catch-up zFS mount error messages on z/OS V1R11 systems

Systems SC64 and SC70 function ship requests to the owning system SC65.

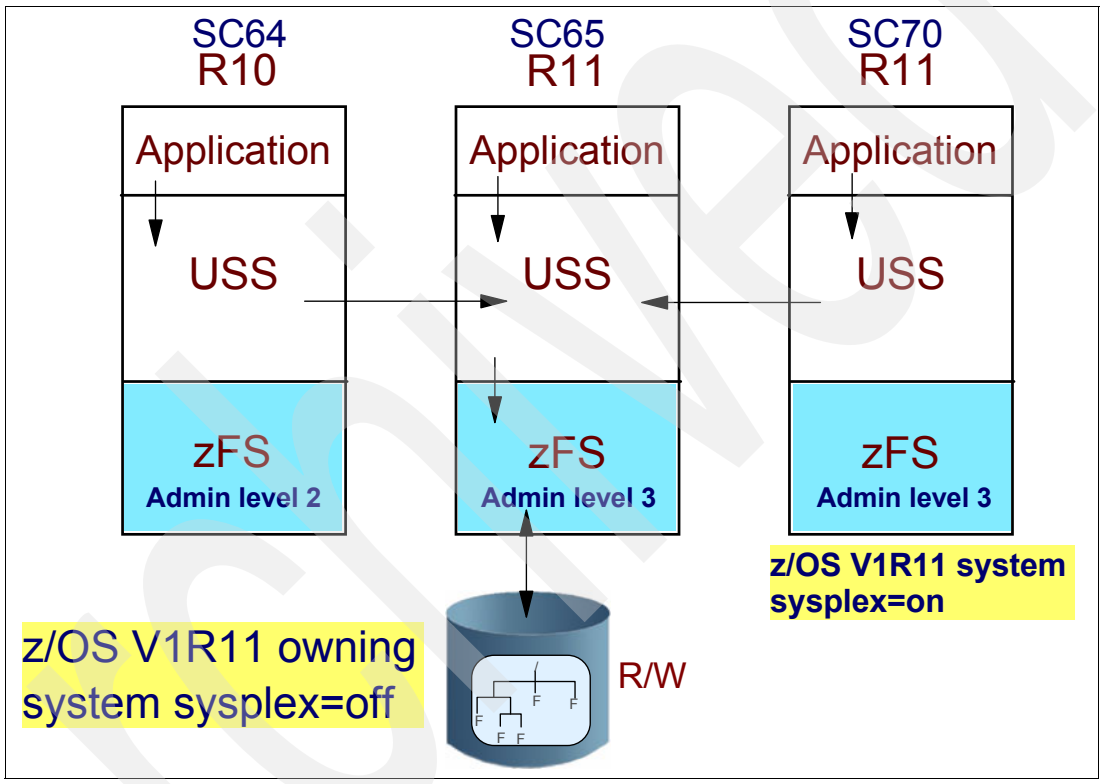


Figure 16-24 z/OS V1R11 systems sysplex=off and sysplex=on

Moving a mounted file system

However, if you try to move the R11 owned zFS file system on system SC65 in Figure 16-24 to the prior release system SC64 (for example, by using a **chmount** command), and there are other R11 zFS sysplex-aware systems (system SC70), the move will fail as shown in Figure 16-28 on page 350.

16.2.4 zFS aggregate mounted on z/OS V1R11 system (sysplex=on)

Figure 16-25 on page 348 shows an aggregate that is mounted on a system running at level z/OS V1R11 with zFS being sysplex-aware. This causes z/OS UNIX to send file requests directly to the local zFS PFS and to not do function shipping from the sysplex=on z/OS V1R11 systems.

When the request is received by the local zFS PFS, zFS determines whether the request can be satisfied from its cache. If it is a read, lookup, or readdir request from the V1R11 sysplex=on system and the data is contained in the cache, the request can be satisfied without communication with the zFS owning system. Write requests are written through, unless you know the space is already allocated on disk.

In this mixed zFS sysplex sharing environment, when a read-write zFS mount is issued on a non-z/OS V1R11 system, a function ship request is made to the owning system.

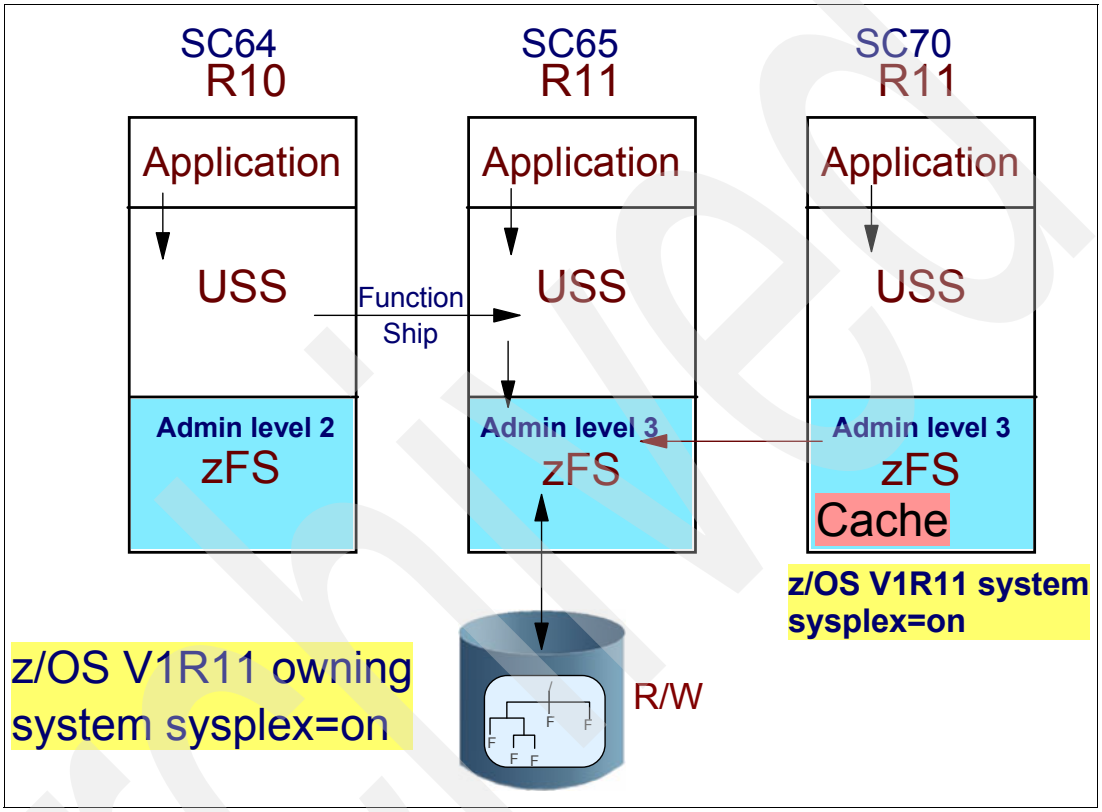


Figure 16-25 zFS aggregate mounted on a system running z/OS V1R11 (sysplex=on)

Figure 16-26 on page 349 shows the file system ownership for UNIX System Services and zFS. The start of the ownership is on system SC64, which is running z/OS V1R10.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC64: z/OS Version 01.10.00
#> rxdowner -a OMVS.HERING.TEST.ZFS
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#> /usr/sbin/chmount -d SC65 /u/hering/test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC64     - File System local-client=Y
USS Owner    : SC65     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 16-26 Moving the zFS aggregate to system SC65 running z/OS V1R11

If you try to move the zFS file system owned on the z/OS V1R11 system to a prior release system, and there are other R11 zFS sysplex-aware systems, the move fails because z/OS UNIX does not unmount the zFS file system from the R11 systems.

The R11 systems have the zFS file system allocated shared, and the zFS of the system running at a lower level of z/OS attempts to allocate the zFS file system exclusively. Figure 16-27 displays the new situation.

```
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 18.50.55 GRS STATUS 832
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FFB00   SHARE      OWN
SC65      ZFS                 001E       007FFB00   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
```

Figure 16-27 zFS aggregate allocated shared on systems running z/OS V1R11

However, the zFS file system can be moved to other R11 zFS sysplex-aware systems. Note that this changes the z/OS UNIX ownership only. Thus, after a zFS file system is owned by an R11 sysplex-aware system, it cannot be moved to a system with zFS being sysplex-unaware. Figure 16-28 on page 350 shows example commands.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC70: z/OS Version 01.11.00
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
#> /usr/sbin/chmount -d SC64 /u/hering/test
FOMF0504I mount error: 9D EF096058
EMVSERR: A MVS environmental or internal error has occurred
Description: HFS-compat mount, - error attaching aggregate or was found not to
be HFS-compat.
#> /usr/sbin/chmount -d SC70 /u/hering/test
#> rxdowner -d /u/hering/test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 16-28 Moving the zFS aggregate UNIX System Services ownership between systems running z/OS V1R11

There is one exception to this rule. If the R11 system is the last (or only) sysplex-aware system in the sysplex, then z/OS UNIX will unmount the sysplex-aware file system before it moves it to the other system. In this case, the move is successful.

Figure 16-29 shows that system SC65 is the only sysplex-aware system and that it owns the aggregate.

```
D GRS,RES=(SYSZIOEZ,IOEZLT.OMVS.HERING.TEST.ZFS*)
ISG343I 16.18.01 GRS STATUS 321
S=SYSTEMS SYSZIOEZ IOEZLT.OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                  003A       007EBE88 EXCLUSIVE    OWN
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 18.24.56 GRS STATUS 259
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                  003A       007FF808   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                  003A       007FF2E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                  003A       007FF2E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                  003A       007FF2E8   SHARE      OWN
```

Figure 16-29 Enqueue resource information for zFS aggregate

Figure 16-30 shows that the move to a sysplex-unaware system is successful in this situation.

```
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
#> /usr/sbin/chmount -d SC64 /u/hering/test
#> rxdowner -d /u/hering/test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=Y
USS Owner    : SC64     - File System read-only=N
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N

#>
```

Figure 16-30 Moving the aggregate ownership from last or only sysplex-aware system

When all systems are at level z/OS V1R11 and zFS is running sysplex-aware, all catch-up mounts are successful as long as the aggregate can be opened at each member. If the aggregate cannot be opened at a member (for example, due to no DASD connectivity), then the catch-up mount fails and z/OS UNIX again will function ship file requests to the z/OS UNIX owner system.

16.2.5 Further migration considerations for zFS in z/OS V1R11

There are several other changes introduced in z/OS V1R11.

- zFS ignores a vnode cache limit that has been specified.
- Unmount processing for a zFS aggregate fails if the aggregate is quiesced, cloning or growing, unless the option force is issued.

Figure 16-31 on page 352 shows a sample of r trying to unmount a quiesced aggregate in z/OS V1R11.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST
Aggregate    : OMVS.HERING.TEST
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> zfsadm quiesce OMVS.HERING.TEST
IOEZ00163I Aggregate OMVS.HERING.TEST successfully quiesced
#> /usr/sbin/unmount test
FOMF0504I unmount error: 72 EF0969B5
EBUSY: The resource is busy
Description: An unmount was requested and the file system was busy with an
administration command.

#>
```

Figure 16-31 Unmounting a quiesced aggregate in z/OS V1R11 fails

In the case of cloning or growing an aggregate, you can wait for the command to complete or you can issue an **unmount** command with the **FORCE** option to interrupt the administration command. If the aggregate is quiesced, then unquiesce it first.

This unmount behavior was quite different in the past, as shown in Figure 16-32.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
MCEVSF: z/OS Version 01.10.00
#> rxdowner -d test
MP Directory : /u/hering/test
File System  : OMVSU.HERING.TEST
Aggregate    : OMVSU.HERING.TEST
PFS Type     : ZFS
Local Sysname: MCEVSF   - File System local-client=N
USS Owner    : MCEVSF   - File System read-only=N
zFS Owner    : MCEVSF   - Aggregate read-only=N, compat-mode=Y, sysplex-aware=-
#> zfsadm quiesce omvsu.hering.test
IOEZ00163I Aggregate OMVSU.HERING.TEST successfully quiesced
#> /usr/sbin/unmount test
#> rxdowner -a OMVSU.HERING.TEST
zFS Owner    : MCEVSF   - Aggregate read-only=N, compat-mode=Y, sysplex-aware=-
#> rxlsaggr -qsd
OMVSU.HERING.TEST
#> zfsadm unquiesce omvsu.hering.test
IOEZ00166I Aggregate OMVSU.HERING.TEST successfully unquiesced
#> rxdowner -a OMVSU.HERING.TEST

Aggregate OMVSU.HERING.TEST cannot be found.
#>
```

Figure 16-32 Unmounting a quiesced aggregate in z/OS V1R10

In this example, system MCEVSF is a z/OS V1R10 single system. Aggregate OMVSU.HERING.TEST is mounted, then quiesced and successfully unmounted.

Nevertheless, the aggregate is still kept attached and quiesced, as **rxlsaggr -qsd** lists all quiesced aggregates. After unquiescing the aggregate, it is automatically detached.

Attaching a compat mode aggregate for growing

In z/OS V1R11 it is no longer possible to attach a multi-file system aggregate. Nevertheless, you still can attach a compat mode aggregate. This is a valid way for growing a zFS aggregate without making it available for use; see Figure 16-33.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC74: z/OS Version 01.11.00
#> zfsadm attach HERING.TEST.GROW
IOEZ00117I Aggregate HERING.TEST.GROW attached successfully
#> zfsowner hering.test.grow
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=N, sysplex-aware=-
#> zfsadm aggrinfo HERING.TEST.GROW
HERING.TEST.GROW (R/W MULT): 566 K free out of total 720
#> zfsadm grow HERING.TEST.GROW -size 0
IOEZ00173I Aggregate HERING.TEST.GROW successfully grown
HERING.TEST.GROW (R/W MULT): 1286 K free out of total 1440
#> zfsadm detach HERING.TEST.GROW
IOEZ00122I Aggregate HERING.TEST.GROW detached successfully
#>
```

Figure 16-33 Growing an attached zFS compat mode aggregate in z/OS V1R11

Using a zFS sysplex-aware /dev structure

If the /dev structure (the file system containing the /dev directory) is accidentally moved to a remote system, then UNIX System Services cannot use the dev devices correctly, if PFS is of type HFS or if zFS is sysplex-unaware. Function shipping to the owning system is not used in this situation. Figure 16-34 shows the error information displayed in this case when you try to open an OMVS shell session.

```
No session was started.  No pseudo-TTYs are available.+
Function = stat(), ending name = '/dev/ptyp0000', return value = -1, errno = 129
(X'00000081'), reason code = 053B006C, description = 'EDC5129I No such file or
directory.'
```

Figure 16-34 Error information about opening a shell session if /dev is owned remotely

However, if the file system containing the /dev directory is a sysplex-aware zFS aggregate, then no problem will occur, regardless of which system is the UNIX System Services and which is the zFS owner.

16.2.6 Exploiting UNIX unmount remount samemode

There are several situations in which zFS can make use of the UNIX unmount remount samenode function, which is introduced in z/OS V1R11. See also 16.4, “UNIX System Services remount samemode support” on page 361 for further information. We discuss the following situations here:

- Dynamically changing zFS attributes for an aggregate that is active
- Making new candidate volumes available while the aggregate is active
- Forcing changing the zFS owner of an aggregate to a specific system

Restriction: Using this UNIX Unmount (Remount) Samemode function is only possible if the minimum required LFS protocol level of z/OS V1R11 is used by *all* systems in the UNIX System Services sysplex file system sharing environment.

Dynamically changing zFS attributes for an aggregate

Figure 16-35 shows a situation with an aggregate that is active with auto-extension disabled; this means aggrow is off. However, at the system level, aggrow is on.

After using the unmount remountsamemode function for this zFS file system, auto-extension is enabled.

```
#> rxlsaggr OMVS.TESTAGGR.ZFS | grep Auto-extend
Auto-extend . . . . . : Disabled
#> zfsadm configquery -aggrgrow
IOEZ00317I The value for configuration option -aggrgrow is ON.
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> /usr/sbin/chmount -s test
#> rxlsaggr OMVS.TESTAGGR.ZFS | grep Auto-extend
Auto-extend . . . . . : Enabled
#>
```

Figure 16-35 Dynamically enable auto-extension for an active zFS aggregate

Adding new candidate volumes

If a sysplex-aware zFS aggregate runs out of space, you cannot move the aggregate to another system to reopen it and make new candidate volumes known to zFS.

However, the samemode option of unmount can be used to force this action, as shown in Figure 16-36 on page 355 and Figure 16-37 on page 355.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TEST.ZFS
Aggregate    : OMVS.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#> tsocmd "ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)"
ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)
ALTERED ALLOCATION STATUS FOR VOLUME *      IS 0
ENTRY OMVS.TEST.ZFS.DATA ALTERED
READY
END
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#>
```

Figure 16-36 Adding new candidate volumes to a zFS aggregate when no space is left

Using the unmount remount samemode function allows you to reopen and therefore make the candidate volume known to zFS; see Figure 16-37.

```
#> /usr/sbin/chmount -s test
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00173I Aggregate OMVS.TEST.ZFS successfully grown
OMVS.TEST.ZFS (R/W COMP): 220950 K free out of total 223200
#>
```

Figure 16-37 Making the new candidate known to the sysplex-aware zFS and growing it

Changing the zFS owner of an aggregate

You can at least temporarily change the zFS owner of an aggregate to a specific system (although this should not be necessary because zFS is “intelligent” enough to choose the right system); see Figure 16-38 on page 356.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> /usr/sbin/chmount -d SC75 test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> /usr/sbin/chmount -s test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC75     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 16-38 Changing the zFS owner of a sysplex-aware aggregate

When adding candidate volumes as shown in Figure 16-36 on page 355, keep this in mind. It is good practice to change the UNIX System Sysplex owner first, because zFS has probably chosen its owing system correctly.

Restriction: If a read-only clone is mounted in parallel, using this method to move the zFS owner does not work.

16.2.7 zFS abnormal termination

If zFS terminates abnormally, automatic ownership movement occurs for both the z/OS UNIX owner and the zFS owner, if possible. zFS aggregate ownership moves unless the file system is unmounted by z/OS UNIX.

Applications with an open file on these file systems receive I/O errors until the file is closed. After zFS is restarted, the operator must remount any file systems that were locally mounted (that is, file systems that were owned by that system and were not moved). This can be done by using the following command:

```
F BPXOINIT,FILESYS=REINIT
```

This causes a remount for each file system that was mounted through a BPXPRMxx parmlib statement.

Restriction: The ALTROOT statement is ignored during processing of the **F BPXOINIT,FILESYS=REINIT** command. You will have to manually issue **SET OMVS=(xx)** command where BXPRMxx is the parmlib member containing the original ALTROOT statement.

16.3 Performance comparisons with sysplex-sharing

In this section we provide a small, simple test scenario that you can easily set up and run in a UNIX System Services sysplex file system sharing environment to learn about the functions provided in zFS to improve performance when accessing a file system. The results illustrate how effective the enhancements added in z/OS V1R11 can be, and that you normally will not see a situation performing worse than before.

16.3.1 Setup and description of the test environments

For these test environments, we used two systems with UNIX System Services sysplex file system sharing, SC74 and SC75. First we created two file systems on the same volume, one of type HFS and the other of type zFS. Next, we created two files of 750 MB in size and filled them with data in both file systems; see Figure 16-39.

```
#> mkdir -m 755 /u/hering/r11test/hfs
#> /usr/sbin/mount -o "sync(60)" -f OMVS.R11TEST.LARGE.HFS -t HFS \
> /u/hering/r11test/hfs
#> chmod 755 /u/hering/r11test/hfs
#> mkdir -m 755 /u/hering/r11test/zfs
#> /usr/sbin/mount -o noreadahead -f OMVS.R11TEST.LARGE.ZFS -t ZFS \
> /u/hering/r11test/zfs
#> /usr/sbin/mount -qv /u/hering/r11test/
----A- OMVS.R11TEST.LARGE.ZFS /u/hering/r11test/zfs
----A- OMVS.R11TEST.LARGE.HFS /u/hering/r11test/hfs
----A- HERING.ZFS         /u/hering
#> # Now creating and filling the files...
#> ls -l /u/hering/r11test/hfs
total 3072000
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 17:43 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 17:48 large_file_2
#> ls -l /u/hering/r11test/zfs
total 3073568
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 22:35 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 22:35 large_file_2
#> rexx 'Say "File size=" 786432000/(1024*1024)||"MB"'
File size= 750MB
```

Figure 16-39 Creating and filling the two HFS and the two zFS files

Note: See *z/OS Distributed File Service zSeries File System z/OS V1R11 Implementation*, SG24-6580, for more information about the setup, JCL, and procedures that we used.

User and client cache with V1R11

The user_cache and client_cache are shown in Figure 16-40 on page 359. In z/OS V1R11, the cache is in two parts, as explained here:

1. For requests at the zFS owner (also called the server), the user_cache_size cache is used.

   The default is 256 MB, the maximum size is 65536 MB (which is 64 GB).

2. For requests at a system that is not the zFS owner (these systems are called clients), the client_cache_size cache is used.

   The default is 128 MB. Again the maximum value is 65536 MB.

   In addition, there is the client_replay_cache, which is used to handle sysplex server replies.

   The default size is 40 MB, the maximum size is 128 MB.

The HFS environment

In the test, the HFS file system was owned by system SC74, and SC75 was the client system. Filling the files was performed on system SC74. HFS was used with the default cache setting.

Sysplex-unaware zFS environment

The first zFS environment was set up to be sysplex-unaware as in the past; the file system owner was SC74 as for HFS. Filling the files was performed on SC74, as well. The user cache size was set to 512 MB.

Sysplex-aware zFS environment

The second zFS environment was set up to be sysplex-aware as it is by default in a z/OS V1R11 UNIX System Services sysplex file system sharing environment.

Attention: In z/OS V1R11, zFS is running sysplex-unaware (for read-write mounted file systems) by default. To change this, specify **`sysplex=on`** in the parameter file.

We used the same zFS files as before and the same user cache size of 512 MB. The client cache size was set to 512 MB, as well. Figure 16-40 on page 359 illustrates this environment.

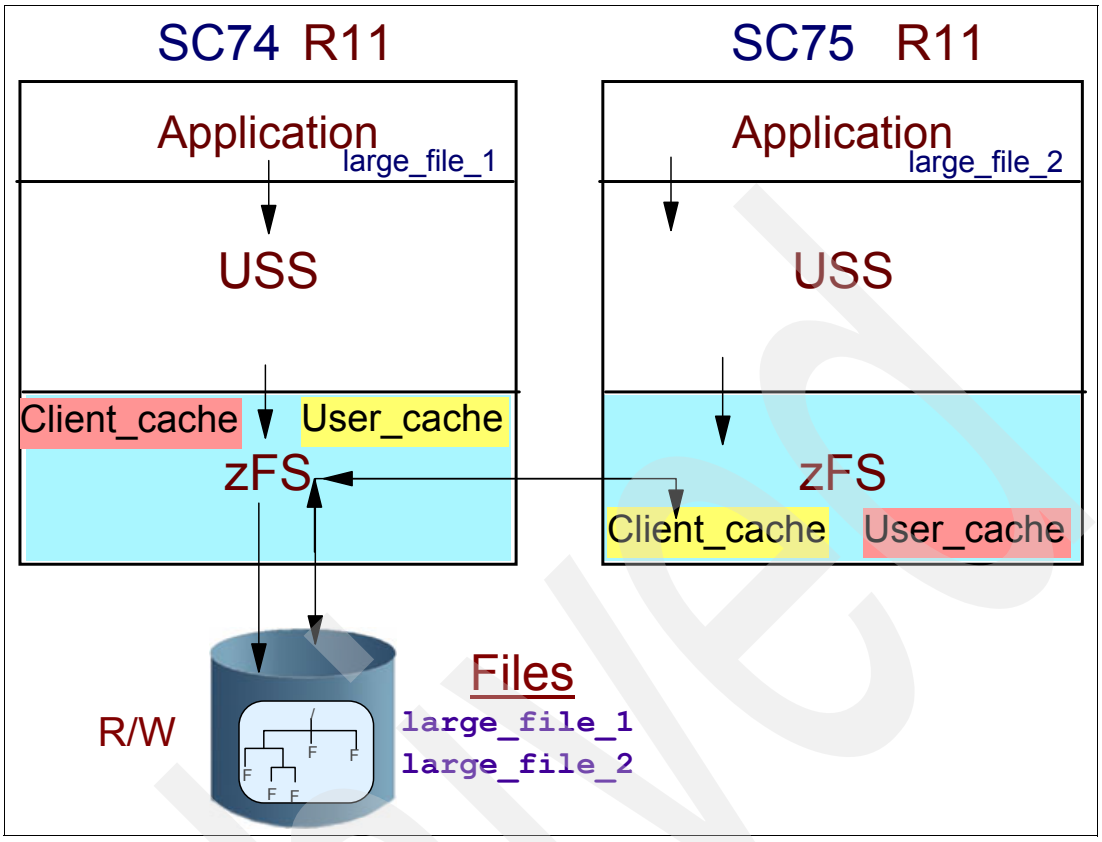


Figure 16-40 Sysplex-aware zFS environment with owner SC74 and client SC75

Accessing the file system files in the tests

The tests done were generally set up as follows.

- We set up two jobs running in parallel on separate systems for every environment tested. One job accessed large_file_1 in system SC74. The other job accessed large_file_2 in system SC75.
- Each job locally started 20 processes running in parallel and performing 200 I/Os to read or write 1 MB blocks.
- 70% of the I/Os were reads and 30% were writes. (This percentage of reads is seen to be realistic for applications.)
- The sequence of I/Os was based on a set of predictable random numbers for offsets into the large files when performing an I/O access. The expression “predictable” means that all tests were using exactly the same set of random numbers and thus make the results directly comparable.

16.3.2 Test results

The following single tests provided the necessary information to clearly identify what was set in the environment.

HFS numbers

Table 16-3 on page 360 lists the results for the HFS tests, locally in SC74 and remotely on SC75. There are no useful numbers listed for the client system because all client requests must be forwarded to the owning system for access to the file system.

Table 16-3 HFS test results

	AVG sec per process	MIN sec per process	MAX sec per process
HFS on owning system	68.44	67.24	68.93
HFS on client system	353.91	351.73	354.81

Results for zFS being sysplex-unaware

Table 16-4 lists the results for the old zFS behavior and the setup as described in "Sysplex-unaware zFS environment" on page 358 is shown. The results on the zFS owning system are good. The numbers for the zFS client are slightly better than for HFS, but still undesirable because of the need to send function shipping requests to the owning system.

Table 16-4 zFS test results with sysplex=no

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	13.19	7.77	15,90
zFS on client system	206,80	205,23	207,62

Results for zFS being sysplex-aware

Table 16-5 lists the results when zFS was run with **`sysplex=on`**. In this test both important caches, the user cache and the client cache, were set to 512 MB. In addition we changed the client reply cache to 128 MB, to ensure it did not develop a bottleneck. The zFS client system then had much better performance because it was able to access data in the cache on the same system and did not always have to go to the owning system for data.

Important: The user cache is used for caching on the zFS owning system. The client cache is used for client access.

In this case, even the client results were quite good. However, depending on the sequence of data access, switching the zFS ownership can provide an advantage. In this case, though, no switch was performed, as planned. Thus, these numbers reflect access from remote.

Table 16-5 zFS test results with sysplex=yes

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	14.14	6.96	16.20
zFS on client system	128.27	124.40	136.54

Tests exploiting automatic move of zFS aggregate ownership

Table 16-6 on page 361 lists the test results on a system that was the zFS client when the test started.

Important: In z/OS V1R11 with zFS being sysplex-aware, a read-write zFS file system's (zFS) ownership is automatically moved to the system most using the file system if it is initially mounted on the wrong system.

These results show that, probably close to the moment when the first 200 I/Os were completed, zFS decided to switch the ownership. Based on the number of I/Os performed, this example highlights the effectiveness of this zFS behavior.

Table 16-6 zFS test results with sysplex=yes on what was initially the client system

Number of 1 MB I/Os	AVG sec per process	MIN sec per process	MAX sec per process
200	120.02	86.25	132.17
2000	373.34	278.57	381.16

From the numbers and behavior observed in this test, we can say that the zFS implementation with automatically changing the zFS owner is the most effective change for using read-write zFS file systems.

Further tests and findings

We performed further testing, and list our comments here regarding the results we found:

- We performed special testing that only read data on the client side. We found no essential difference to a case with a low amount of writes.
- In the tests described, we still saw client cache read fault ratios of about 30%. Therefore, we ran further tests with a client cache size of 1024 M and got response times that were comparable to the server side, because most of the data can be handled or was found in the cache already.
- As long as writes do not enlarge a file, there is no need to move data back right away to the owning system, and all I/O operations can be done with the client cache.

Note: A high client cache size in combination with low or no file increasing on I/Os can provide very desirable performance numbers on client systems.

16.4 UNIX System Services remount samemode support

Remount currently exists but it is used to change the mode of the file system without interruption. So this only allows you to switch from read-write to read-only mode and vice versa.

Remount samemode is introduced to internally unmount and mount the file system back in the same mode. This provides a significant benefit over the earlier possibilities and can be exploited by zFS, especially in several situations.

UNMOUNT command

The **UNMOUNT** command removes a file system from the file system hierarchy. The alias for this command is UMOUNT.

```
UNMOUNT FILESYSTEM(file_system_name)
        DRAIN | FORCE | IMMEDIATE | NORMAL | REMOUNT(RDWR|READ|SAMEMODE)| RESET
```

REMOUNT SAMEMODE

The REMOUNT SAMENODE option on the **UNMOUNT** command specifies that the specified file system be remounted and its mount mode changed, if necessary. REMOUNT takes an optional argument of RDRW, READ, UNMOUNT, or SAMEMODE, as follows:

- If REMOUNT is specified without any arguments, the mount mode is changed from RDWR to READ, or READ to RDWR.
- If RDWR is specified and the current mode is READ, the file system is remounted in RDWR mode.

- If READ is specified and the current mode is RDWR, the file system is remounted in READ mode.
- If SAMEMODE is specified, the file system is remounted (internally unmounted and remounted) without changing the mount mode. You can use this option to attempt to regain use of a file system that had I/O errors.

Note: Using remount samemode is transparent for applications using the file system that is remounted.

16.4.1 Benefits of the remount samemode function

Using this function provides the following benefits:

- A file system that is disabled for writes gets recovered without unmounting it.
  - R/W mounted file systems can have I/O errors and get disabled for write. There is no way to recover from this without unmounting the file system.
- Active write() calls to the file system will not fail during processing.
  - With remount samemode, pending write() calls will be blocked until processing is over.
- zFS attributes for an aggregate that is active can be dynamically changed and pick up the current zFS system settings.
- New candidate volumes can be made available while a zFS aggregate is active. This also makes it easier when adding candidate volumes to an HFS file system.

Important: When zFS is sysplex-aware for read-write mounted file systems, moving the aggregate to another system and then pulling it back no longer works.

16.4.2 Examples of remount samemode

TSO	Command **`UNMOUNT FILESYSTEM(fsname) REMOUNT(SAMEMODE)`**
Shell	Shell command **`/usr/sbin/chmount -s fsname`**
Assembler	Callable service **`CALL BPX1UMT,(fsname,MtmFlags,...)`**; MtmSameMode must be set in MtmFlags.
REXX	Syscall command **`unmount fsname MTM_SAMEMODE`**
C/C++	Function **`unmount(fsname,MTM_SAMEMODE)`**
ISHELL	Using prefix command **`M`** on panel Work with Mounted File Systems as shown in Figure 16-41 on page 363.

Using remount samemode with the ISHELL

To use remount samemode with the ISHELL, follow these steps:

1. Place the cursor under the File_systems pull-down at the top of the panel and select option **1. Mount Table**.
2. Place an `M` next to the file system that you want to do the REMOUNT.
3. When you receive the panel, select option **4. Remount Samemode (R/O)**, as shown in Figure 16-41 on page 363.

The selected file system is about to be remounted without changing the mode. The file system is first unmounted and then mounted in the same mode.

```
BPXWP20                  Work with Mounted File Systems
C
   BPXWP60            Select the attribute to change
S
U  Select the attribute to change:
   4_  1.  Change mount mode to R/W
       2.  Change Owning system from SC63
       3.  Change automove attribute...
       4.  Remount Samemode (R/O)

   New owning system    ________

    F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel
m
_ HFS.ZOSR1A.Z1ARB1.JAVA31V5                    Available
_ HFS.ZOSR1A.Z1ARB1.JAVA64V5                    Available
_ HFS.ZOSR1A.Z1ARB1.JV390                       Available
_ HFS.ZOSR1A.Z1ARB1.ROOT                        Available
```

Figure 16-41 Selecting 4 to invoke remount samemode

A remount samemode confirmation panel is shown in Figure 16-42. However, it does not mention that the unmount and the new mount is done transparently for applications using the file system, provided it is successful.

```
BPXWP20                  Work with Mounted File Systems
C
   _BPXWP80            Remount Samemode Confirmation
S
U  CAUTION:
      The selected file system is about to be remounted without
   changing the mode.  The file system is first unmounted and then
   mounted in the same mode.
   File system name:
   HFS.ZOSR1A.Z1ARA1.XML

   To proceed with the remount without changing the mode, press Enter.
   To cancel the remount and continue, use the Cancel function key.  To
   exit this function, use the Exit function key.

m   F1=Help       F3=Exit       F4=Name       F6=Keyshelp  F12=Cancel
_
_ HFS.ZOSR1A.Z1ARB1.JAVA64V5                    Available
_ HFS.ZOSR1A.Z1ARB1.JV390                       Available
_ HFS.ZOSR1A.Z1ARB1.ROOT                        Available
_ HFS.ZOSR1A.Z1ARB1.XML                         Available
```

Figure 16-42 Displaying the remount samemode confirmation panel

Using remount samemode with SYSCALL command unmount

Figure 16-43 on page 364 shows an example using the interactive REXX routine rexx that works with a zFS aggregate while it is remounted with the same mode in parallel.

```
#> rexx
SH> s "open test.file" o_creat+o_wronly 600
OMVS Return Value (retval) = 4
SH> filedata = "This is a test line." || esc_n
SH> s "write 4 filedata"
OMVS Return Value (retval) = 21
SH> sh "rxdowner -d . | grep 'Aggregate    :'"
Aggregate    : HERING.ZFS
SH> s "unmount HERING.ZFS" mtm_samemode
OMVS Return Value (retval) = 0
SH> s "write 4 filedata"
OMVS Return Value (retval) = 21
SH> s "close 4"
OMVS Return Value (retval) = 0
SH> exit
#> cat test.file
This is a test line.
This is a test line.
#>
```

Figure 16-43 Working with a zFS aggregate while it is remounted with the same mode

Remount samemode is not supported in mixed environments

All systems in the UNIX System Services file system sharing environment must be running at z/OS V1R11 in order for that function to be used. Figure 16-44 shows an example for a mixed environment.

```
#> /usr/sbin/chmount -s test
FOMF0504I remount error: 86 58804F6
ENOSYS: The function is not implemented
Notice: this is an internal error
JrLfsProtocolLev: The function could not be performed because the minimum
required LFS protocol level was not met by all systems in the sysplex group.
#>
```

Figure 16-44 Remount samemode fails if a system previous to z/OS V1R11 is active

Restriction: Using the UNIX remount samemode function is only possible if the minimum required LFS protocol level of z/OS V1R11 is used by all systems in the UNIX System Services sysplex file system sharing environment.

16.5 Asynchronous accept_and_recv() support

The asyncio (BPX1AIO BPX4AIO) callable service has been extended to include the accept_and_recv (BPX1ANR, BPX4ANR) function.

- The asyncio accept() and asyncio recv() calls are combined into a single API.
- The synchronous accept_and_recv() call is also optimized to improve performance.

Only one single socket operation is required between UNIX System Services and Communication Services to achieve the accept_and_recv() function. Previously two socket operations (accept and recv) were required.

Note: This new function can be exploited by WebSphere. It improves the network application performance of network applications such as Web browsers.

16.6 Using the ISPF editor for OEDIT and OBROWSE

This enhancement provides the same edit and browse functionality as with ISPF edit and browse. Previously oedit and obrowse had their own dialogs implemented using ISPF EDIF and BRIF services. Although EDIF and BRIF dialogs are very similar to ISPF edit and browse, they are not identical.

This change makes use of the new capabilities in edit and browse to be able to handle z/OS UNIX files. Specifically, you can edit files tagged as ASCII because this is supported by ISPF edit.

The old dialog code, panels, and messages for oedit and obrowse have not been removed in case the oedit or obrowse REXX programs have been locally copied and modified. Those versions of the code should continue to work. However, consider changing those programs to make direct use of ISPF EDIT and BROWSE services.

16.6.1 Using OEDIT and OBROWSE when using tagging and format settings

Figure 16-45 shows a short scenario of editing files to illustrate how this enhancement works. The example starts with copying a sequential MVS data set with a VB record format before editing the UNIX System Services file.

```
$> cp "//testfile" /dev/fd1
1234567890
aaaaaaaaaa
zzzzzzzzzz
$> cp -F crlf "//testfile" testfile
$> ls -H testfile
-rw-r--r--  crlf   1 HERING   SYS1          36 May 13 11:41 testfile
$> od -X testfile
0000000000          F1F2F3F4        F5F6F7F8        F9F00D25        81818181
0000000020          81818181        81810D25        A9A9A9A9        A9A9A9A9
0000000040          A9A90D25
0000000044
$> oedit testfile
```

Figure 16-45 Copy an MVS data set to UNIX and start editing the UNIX System Services file

Figure 16-46 on page 366 shows that the data is presented as designed in the edit session.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________
EDIT       /u/hering/testfile                          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 1234567890
000002 aaaaaaaaaa
000003 zzzzzzzzzz
****** **************************** Bottom of Data ****************************
```

Figure 16-46 Editing a UNIX System Services file with oedit in z/OS V1R11

Figure 16-47 shows the creation of an ASCII file and starting an oedit session.

```
$> ls testfile.ascii
ls: FSUM6785 File or directory "testfile.ascii" is not found
$> _ICONV_TECHNIQUE=R iconv -f IBM-1047 -t ISO8859-1 testfile > testfile.ascii
$> od -X testfile.ascii
0000000000          31323334        35363738        39300D0A        61616161
0000000020          61616161        61610D0A        7A7A7A7A        7A7A7A7A
0000000040          7A7A0D0A
0000000044
$> chtag -tc ISO8859-1 testfile.ascii
$> extattr -F crlf testfile.ascii
$> ls -TH testfile.ascii
t ISO8859-1   T=on  -rw-r--r--  crlf   1 HERING   SYS1          36 May 13 14:10
testfile.ascii
$> oedit testfile.ascii
```

Figure 16-47 Creating a UNIX ASCII file with line end format CRLF starting oedit

Note: Using the environment variable **_ICONV_TECHNIQUE** with the **iconv** command can be used for z/OS levels at V1R9 or later. At that time it started to use Unicode instead of the old Language Environment interface. However, the default behavior is still compatible with the Language Environment behavior and does not use the correct line end conversion.

Figure 16-48 verifies that the data is presented again as planned.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________
EDIT       /u/hering/testfile.ascii                    Columns 00001 00072
000001 1234567890
000002 aaaaaaaaaa
000003 zzzzzzzzzz
****** **************************** Bottom of Data ****************************
```

Figure 16-48 Editing a UNIX System Services ASCII file with oedit in z/OS V1R11

However, using the command **obrowse** displays the data still in ASCII, as shown in Figure 16-49 on page 367.

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    /u/hering/testfile.ascii                  Line 00000000 Col 001 036
******************************** Top of Data ********************************
...........//////////..:::::::::::..
******************************** Bottom of Data *****************************
```

Figure 16-49 Browsing a UNIX System Services ASCII file with obrowse in z/OS V1R11

You need to enter the command **DISPLAY ASCII** to present the characters correctly, but the data is still presented wrong. The line end format settings are not supported, as shown in Figure 16-50.

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    /u/hering/testfile.ascii                    Converted data shown
******************************** Top of Data ********************************



 ------------------------------------------------------------------------------
1234567890..aaaaaaaaaa..zzzzzzzzzz..
333333333300666666666600777777777700
1234567890DA1111111111DAAAAAAAAAADA
 ------------------------------------------------------------------------------
******************************** Bottom of Data *****************************
```

Figure 16-50 Displaying UNIX System Services ASCII file data in EBCDIC with obrowse in z/OS V1R11

Note: When using **oedit**, ASCII CRLF (“0D0A”x), ASCII LF (“0A”x) and EBCDIC NL (“15”x) are supported without any need to set a line end format.

16.6.2 Using OEDIT and OBROWSE from OMVS shell in superusermode

In this section we demonstrate that the support to edit files in superuser mode from an OMVS shell currently does not work correctly with the changes for **oedit** and **obrowse** made in z/OS V1R11.

Figure 16-51 shows a non-superuser shell session starting to edit a file owned by UID 0.

```
$> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC64: z/OS Version 01.10.00
$> ls -En /etc/profile
-rwxr-xr-x  --s-  1 0        1          10797 Jul 21  2004 /etc/profile
$> oedit /etc/profile
```

Figure 16-51 Using oedit in z/OS V1R10 to edit a file owned by UID 0

Figure 16-52 on page 368 shows that the oedit session in this case simply allows read-only access.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       /etc/profile                                  Columns 00001 00072
000002 #       Licensed Materials - Property of IBM                           *

 ...
 000022 #
 000023 # Some environme | File opened for read only | tenate data set names
 000024 # or directories |_____________________________| (':') as a delimiter
 Command ===>                                              Scroll ===> CSR
  F1=Help      F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F12=Cancel
```

Figure 16-52 oedit session in z/OS V1R10 only allows read-only access to the data

Figure 16-53 shows the next situation, starting an oedit session after switching to superuser mode.

```
$> su
#> oedit /etc/profile
```

Figure 16-53 Using oedit in z/OS V1R10 to edit a file after switching to UID 0

Figure 16-54 verifies that this time the shell authorization is used for the edit session, as well.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       /etc/profile                                  Columns 00001 00072
 000002 #       Licensed Materials - Property of IBM                           *
 ...
 000022 #
 000023 # Some environment variables allow you to concatenate data set names
 000024 # or directories. Most use the colon character (':') as a delimiter
 Command ===>                                              Scroll ===> CSR
  F1=Help      F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F12=Cancel
```

Figure 16-54 oedit session in z/OS V1R10 in authorized mode “inherited” from the shell

Figure 16-55 shows the same situation in z/OS V1R11 as before, a shell session in superuser mode.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC70: z/OS Version 01.11.00
#> oedit /etc/profile
```

Figure 16-55 Using oedit in z/OS V1R11 to edit a file owned by UID 0

In this case the edit mode is switched to the view mode and you can only read data. This is shown in Figure 16-56 on page 369.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
VIEW       /etc/profile                                    View substituted
000002 #       Licensed Materials - Property of IBM                          *
...
000022 #
000023 # Some environment variables allow you to concatenate data set names
000024 # or directories. Most use the colon character (':') as a delimiter
Command ===>                                                  Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

Figure 16-56 oedit session in z/OS V1R11 automatically switched to view mode

APAR OA29012

Closely related to this problem, ISPF APAR OA29012 has been opened. The solution for this APAR should solve the problem.

In the meantime you can use one of the following circumventions:

- Edit files in superuser mode from the ISHELL, or simply while the ISHELL is active in superuser mode in your TSO environment.
- Or, use the **soedit** command from the UNIX System Services Tools package with the most recently updated command **SU** that circumvents this edit problem.

Tip: This version of **SU** is available in *z/OS Distributed File Service zSeries File System z/OS V1R11 Implementation*, SG24-6580.

Figure 16-57 illustrates using **soedit** instead of **oedit** for editing the same file.

```
$> soedit /etc/profile
```

Figure 16-57 Using soedit in z/OS V1R11 to edit a file owned by UID 0

In this case the edit mode is kept as desired; see Figure 16-58.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       /etc/profile                                  Columns 00001 00072
****** ***************************** Top of Data ******************************
000002 #       Licensed Materials - Property of IBM                          *
...
000022 #
000023 # Some environment variables allow you to concatenate data set names
000024 # or directories. Most use the colon character (':') as a delimiter
Command ===>                                                  Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

Figure 16-58 soedit session in z/OS V1R11 in desired edit mode

16.7 Exploiting the SYSREXX enhancements

With the SYSREXX enhancements, you can use UNIX System Services functions by using the SYSCALL interface. Here we provide a REXX example that you can use on a console to

unmount a UNIX System Services file system using the standard UNIX System Services API. No such command for a standard unmount processing is available for usage by an operator.

16.7.1 Unmounting a UNIX System Services file system by an operator

You can mount file systems placed into a BPXPRMxx parmlib member within a MOUNT statement by using the system command **SET OMVS=(xx)**. However, there is no corresponding unmount function.

Figure 16-59 shows how to use the REXX named USSUMNT by an operator to unmount a file system.

```
D OMVS,F,N=OMVS.HERING.TEST.ZFS
BPXO045I 17.37.21 DISPLAY OMVS 632
OMVS     0011 ACTIVE             OMVS=(1B)
TYPENAME   DEVICE ----------STATUS---------- MODE  MOUNTED    LATCHES
ZFS         12632 ACTIVE                     RDWR  05/11/2009  L=172
  NAME=OMVS.HERING.TEST.ZFS                      21.10.16    Q=172
  PATH=/u/hering/test
  AGGREGATE NAME=OMVS.HERING.TEST.ZFS
  OWNER=SC70     AUTOMOVE=Y CLIENT=N
@USSUMNT
Syntax: USSUMNT fs_name < Drain | Force | Immed | normal | RESet |
REMount | Samemode
@USSUMNT OMVS.HERING.TEST.ZFS
Unmount processing has been performed successfully.
IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
```

Figure 16-59 Unmounting a UNIX System Services file system by an operator

The REXX USSUMNT is available as additional material with *UNIX System Services z/OS Implementation*, SG24-7035.

16.8 Alternate sysplex root support

Several clients have experienced issues involving the system root being a single point of failure in a sysplex. When the sysplex root gets corrupted and becomes unavailable and unrecoverable, it could lead to a multi-system outage situation (MSO) and require a sysplex-wide IPL.

In a sysplex configuration, the alternate sysplex root file system becomes a standby for the sysplex root file system. The alternate can be used to replace the current sysplex root file system if the sysplex root file system becomes unavailable.

You establish the alternate sysplex root file system by using the ALTROOT statement in the BPXPRMxx parmlib member during OMVS initialization, or by using the **SET OMVS** command.

Using alternate sysplex root support, you can dynamically replace a failed or failing sysplex root automatically, or replace it manually by using an external command.

This support provides the following advantages:

- It eliminates the single point of failure for the sysplex root file system.
- It allows for a dynamic replacement of a failed or failing sysplex root.

- It prevents that a single point sysplex root failure causes a multi-system outage.

16.8.1 Using automatic replacement

To set up an alternate sysplex root file system, complete the following steps.

1. Allocate the alternate sysplex root file system and set up the mount points and symlinks *exactly* as in the current sysplex root; see Figure 16-61 on page 372.
   - You can use the **pax** shell command instead of the copytree utility to populate the alternate sysplex root.
   - The ALTROOT mount point needs to be created before an alternate sysplex root file system can be established using the BPXPRMxx parmlib member. There is no default mount point.
2. Establish the alternate sysplex root using the new ALTROOT statement in a BPXPRMxx parmlib member:

   ```
   ALTROOT FILESYSTEM ('PLEX75.SYSPLEX.ROOTALT1.ZFS') MOUNTPOINT('/rootalt')
   ```

   - Mount the ALTROOT file system as read-only and with AUTOMOVE=Yes.
   - Establish it during OMVS initialization or by using the **SET OMVS** command as shown in Figure 16-60.

```
SET OMVS=(RA)
IEE252I MEMBER BPXPRMRA FOUND IN SYS1.PARMLIB
BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
IEF196I IEF285I   SYS1.PARMLIB
IEF196I IEF285I   VOL SER NOS= BH5CAT.
IEF196I IEF285I   CPAC.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z19CAT.
IEF196I IEF285I   SYS1.IBM.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z1BRC1.
BPXF013I FILE SYSTEM PLEX75.SYSPLEX.ROOTALT1.ZFS 031
WAS SUCCESSFULLY MOUNTED.
```

Figure 16-60 Establishing the altroot sysplex root with command SET OMVS

Important: Make sure that the alternate sysplex root does not reside in the same volume, device, and control unit as the current sysplex root.

Figure 16-61 on page 372 shows an example of using the copytree utility to create the alternate sysplex root.

```
#> zfsadm define PLEX75.SYSPLEX.ROOTALT1.ZFS -volume BH5ST1 -cylinders 2 2
IOEZ00248I VSAM linear dataset PLEX75.SYSPLEX.ROOTALT1.ZFS successfully created.
#> zfsadm format PLEX75.SYSPLEX.ROOTALT1.ZFS -compat
IOEZ00077I HFS-compatibility aggregate PLEX75.SYSPLEX.ROOTALT1.ZFS has been successfully
created
#> /usr/sbin/chmount -w /
#> mkdir -m 700 /rootalt
#> /usr/sbin/chmount -r /
#> /usr/sbin/mount -f PLEX75.SYSPLEX.ROOTALT1.ZFS /rootalt
#> copytree / /rootalt
Copying / to /rootalt
From filesystem PLEX75.SYSPLEX.ROOT.ZFS
  To filesystem PLEX75.SYSPLEX.ROOTALT1.ZFS
Scanning for file nodes...
Skipping mountpoint: //u
Skipping mountpoint: //SC75
Skipping mountpoint: //SC74
Skipping mountpoint: //pp
Skipping mountpoint: //Z1BRC1
Skipping mountpoint: //rootalt
Processing 28 nodes
Creating directories
Creating other files
Setting file attributes
Creating mount points

******************

Copy complete.  Error count= 0
Directory errors:   0
Directories copied: 15
File errors:        0
Files copied:       1
Symlink errors:     0
Symlinks copied:    12
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
#> /usr/sbin/unmount /rootalt
#>
```

Figure 16-61 Allocating and filling the alternate root file system

ALTROOT mount status

By completing those tasks, you establish an alternate sysplex root in the shared file system configuration. The alternate sysplex root is mounted in read-only mode at the specified mount point and is designated as AUTOMOVE. When the alternate sysplex root becomes the current sysplex root, it is mounted in read-only mode and is designated as AUTOMOVE regardless of the current sysplex root settings.

Note: Although the internal structure of the sysplex root is more complex, as it should be, the processing works as desired.

Comments

Consider the following information:

- The alternate sysplex root file system is a hot standby for the sysplex root file system that is used to replace the current sysplex root file system when the sysplex root file system becomes unowned or unrecoverable.
- Ensure that the alternate sysplex root does *not* reside in the same volume, device, and control unit as the current sysplex root.
- Validation on the mount points in the alternate root, compared to the current root, will be performed during replacement processing. If mount points are missing or incorrect, then the replacement fails.
- The sample BPXPRMxx in the SYS1.SAMPLIB data set provides a detailed explanation of ALTROOT statements and accepted keywords.
- The current sysplex root and the alternate sysplex root file system type do not have to be identical; they can be either zFS or HFS.

Restrictions for replacement

The following constraints must be met before using alternate sysplex root support:

- The system must be in a shared file system configuration.
- All systems in the shared file system environment must be at the minimum system level of z/OS V1R11.
- The ALTROOT mount point must not exceed 64 characters, and it must reside in the root directory of the current sysplex root file system.
- The UID, GID, and permission bits of the root directory in the alternate sysplex root must match the root directory in the current sysplex root.
- If the SECLABEL class is active and the MLFSOBJ option is active, then the multilevel security label for the alternate sysplex root must be identical to the assumed multilevel security label of the current sysplex root.
- All systems in the shared file system configuration must have direct access to the new file system and must be able to locally mount it.
- The file system type for the alternate sysplex root and the current sysplex root must be either HFS or ZFS.
- The alternate sysplex root PFS must be active on all systems in the shared file system configuration.
- The real path name for the mount points in the current sysplex root must not exceed 64 characters in length.
- The sysplex root or any directories in the sysplex root file system must not be exported by the DFS or SMB server.

Important: If you make changes to the current sysplex root after the alternate sysplex root has been successfully established, you must make the same changes to the alternate sysplex root.

16.8.2 Displaying the alternate sysplex root

The output of **D OMVS,O** command will display the status of the alternate sysplex root.

- If the alternate sysplex root is established and active, the name of alternate root is displayed as shown in Figure 16-62.

```
D OMVS,O
BPXO043I 09.02.37 DISPLAY OMVS 445
OMVS     0010 ACTIVE             OMVS=(RA)
...
ALTROOT         = PLEX75.SYSPLEX.ROOTALT1.ZFS
```

Figure 16-62 Displaying the alternate sysplex root being active on D OMVS

- If the alternate sysplex root is inactive, an empty ALTROOT line is displayed instead:

```
ALTROOT =
```

The **F BPXOINIT,FILESYS=DISPLAY,GLOBAL** command also displays information about the status of the alternate sysplex root.

- If the alternate sysplex root is established in the sysplex, the name is displayed as shown in Figure 16-63.

```
F BPXOINIT,FILESYS=DISPLAY,GLOBAL
BPXM027I COMMAND ACCEPTED.
BPXF242I 2009/05/18 12.31.40 MODIFY BPXOINIT,FILESYS=DISPLAY,GLOBAL
SYSTEM   LFS VERSION ---STATUS-------------------- RECOMMENDED ACTION
SC74       1. 11.  0 VERIFIED                     NONE
SC75       1. 11.  0 VERIFIED                     NONE
CDS VERSION=  2        MIN LFS VERSION=  1. 11.  0
DEVICE NUMBER OF LAST MOUNT=      88
MAXIMUM MOUNT ENTRIES=     500   MOUNT ENTRIES IN USE=      40
MAXIMUM AMTRULES=           50   AMTRULES IN USE=            2
MAXSYSTEM=                   4
ALTROOT=  PLEX75.SYSPLEX.ROOTALT1.ZFS
BPXF040I MODIFY BPXOINIT,FILESYS PROCESSING IS COMPLETE.
```

Figure 16-63 Displaying the alternate sysplex root being active on F BPXOINIT,FILESYS

- If the alternate sysplex root is not active, the following line is displayed instead:

```
ALTROOT=  N/A
```

16.8.3 Disabling the alternate sysplex root

The alternate can be disabled for the following reasons:

- You unmount the alternate sysplex root file system.
- A downlevel system, previous to z/OS V1R11, is joining the UNIX System Services shared file system configuration.
- You define a BPXPRMxx parmlib member with a specific ALTROOT statement:

```
ALTROOT NONE
```

This is a useful way to disable the alternate root, if necessary. This method deletes any outstanding BPXF253E eventual action operator messages.

Note: In this case the alternate sysplex root file system is left mounted in the sysplex as a regular file system.

16.8.4 New console messages

Figure 16-64 provides a list of new messages that have been introduced in relation to the alternate sysplex root support.

```
BPXF248I THE NEW SYSPLEX ROOT FILE SYSTEM IS MISSING THE FOLLOWING MOUNT POINT:
         NAME: filesysname PATH: path
BPXF249I THE MOUNT POINT PATH FOR THE FOLLOWING FILE  SYSTEM EXCEEDS THE
         MAXIMUM LENGTH:
         NAME: filesysname
BPXF252I ALTROOT FILE SYSTEM fsname WAS NOT MOUNTED.
         RETURN CODE = return_code REASON CODE = rsn_code
BPXF254I ALTROOT STATEMENT IN PARMLIB MEMBER ONLY VALID IN SHARED FILE SYSTEM
         ENVIRONMENT.
BPXF255I ALTROOT NONE PARMLIB STATEMENT SUCCESSFULLY PROCESSED ON THIS SYSTEM.
BPXF256I fsname IS NOW ACTIVE AS CURRENT SYSPLEX ROOT.
BPXF257I SYSPLEX ROOT REPLACEMENT FAILED:
         RETURN CODE = return_code REASON CODE = rsn_code
BPXF258I SYSPLEX ROOT REPLACEMENT FAILED:
         <text>
BPXF259I ALTROOT FAILED TO MOUNT ON THIS SYSTEM.
               RETURN CODE = return_code  REASON CODE = rsn_code
BPXF253E ALTROOT INACTIVE:
       - ALTROOT FILE SYSTEM IS NOT MOUNTED OR IS UNMOUNTED.
       - ALTROOT FILE SYSTEM IS CURRENTLY UNOWNED.
       - NOT ALL SYSTEMS ARE AT THE REQUIRED RELEASE.
       - ALTROOT IS NOW ACTIVE AS CURRENT SYSPLEX ROOT.
       - ALTROOT MOUNT FAILED ON SOME SYSTEMS.
```

Figure 16-64 Messages related to processing of use of the alternate sysplex root

16.8.5 Command replacement

You can initiate the replacement of a failed or failing sysplex root manually by using the existing operator command **F OMVS,NEWROOT** with a new option FORCE on any system in the sysplex.

Important: This method does not use the alternate sysplex root file system that is established using the BPXPRMxx parmlib member. It uses the file system that is specified on the command invocation.

You need to prepare a second file system that can replace the sysplex root, as illustrated in Figure 16-61 on page 372. Let us assume we do this and the new file system name is PLEX75.SYSPLEX.ROOTALT2.ZFS. Figure 16-65 on page 376 shows the command to use.

```
 F OMVS,NEWROOT=PLEX75.SYSPLEX.ROOTALT2.ZFS,COND=FORCE
*010 BPXI085D REPLACEMENT OF SYSPLEX ROOT IS REQUESTED. REPLY 'Y' TO
 PROCEED. ANY OTHER REPLY TO CANCEL.
 R 10,Y
 IEE600I REPLY TO 010 IS;Y
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING 967
 COMPLETED SUCCESSFULLY.
 IOEZ00048I Detaching aggregate PLEX75.SYSPLEX.ROOT.ZFS
```

Figure 16-65 Replacing a sysplex root manually using F OMVS,NEWROOT with option FORCE

A BPXI085D WTOR message is issued to the console to confirm the FORCE option.

16.8.6 Additional requirements

Note the following requirements for this support:

- The new sysplex root must not be mounted, HSM migrated, or in use.
- All systems in the shared file system configuration must be at z/OS V1R11 to use the command option.
- Validation on the mount points is performed during replacement processing. If mount points are missing or incorrect, the replacement will fail.

16.8.7 Migration and coexistence considerations

The alternate sysplex root file system support requires a minimum system level of z/OS V1R11.

If a downlevel system (z/OSV1R10 or earlier) joins the shared file system configuration after an alternate sysplex root file system has been successfully established, then the established alternate sysplex root file system is not considered an alternate sysplex root file system anymore.

Note: In this situation, the alternate sysplex root file system will be treated as a regular mounted file system in the shared file system configuration.

16.8.8 Automatic replacement of a sysplex root

This section describes the case when an active sysplex root is lost and then automatically replaced by the alternate sysplex root. Figure 16-66 on page 377 shows the active alternate sysplex root.

```
D OMVS,O
BPXO043I 17.14.27 DISPLAY OMVS 586
OMVS     0010 ACTIVE             OMVS=(RA)
CURRENT UNIX CONFIGURATION SETTINGS:
...
ALTROOT         = PLEX75.SYSPLEX.ROOT.ZFS
F BPXOINIT,FILESYS=DISPLAY,GLOBAL
BPXM027I COMMAND ACCEPTED.
BPXF040I MODIFY BPXOINIT,FILESYS PROCESSING IS COMPLETE.
BPXF242I 2009/08/17 17.16.55 MODIFY BPXOINIT,FILESYS=DISPLAY,GLOBAL
590
SYSTEM   LFS VERSION ---STATUS---------------------- RECOMMENDED ACTION
SC74       1. 11.  0 VERIFIED                        NONE
SC75       1. 11.  0 VERIFIED                        NONE
CDS VERSION=  2        MIN LFS VERSION=  1. 11.  0
DEVICE NUMBER OF LAST MOUNT=     142
MAXIMUM MOUNT ENTRIES=     500   MOUNT ENTRIES IN USE=      33
MAXIMUM AMTRULES=           50   AMTRULES IN USE=           2
MAXSYSTEM=                   4
ALTROOT=  PLEX75.SYSPLEX.ROOT.ZFS
```

Figure 16-66 Displaying the alternate sysplex root being active

The current root is mounted as NOAUTOMOVE, as shown in Figure 16-67.

```
D OMVS,F,N=PLEX75.SYSPLEX.ROOT.HFS
BPXO045I 17.31.58 DISPLAY OMVS 596
OMVS     0010 ACTIVE             OMVS=(RA)
TYPENAME   DEVICE ----------STATUS---------- MODE  MOUNTED    LATCHES
HFS             1 ACTIVE                     READ  08/17/2009  L=56
  NAME=PLEX75.SYSPLEX.ROOT.HFS                      16.48.58    Q=56
  PATH=/
  OWNER=SC74     AUTOMOVE=N CLIENT=N
```

Figure 16-67 Current root mounted at SC74 with automove(no)

However, when UNIX processing on SC74 begins to be taken down by the command **F OMVS,SHUTDOWN**, the messages shown in Figure 16-68 are displayed.

```
*BPXI066E OMVS SHUTDOWN COULD NOT MOVE OR UNMOUNT ALL FILE SYSTEMS
 BPXM054I FILE SYSTEM PLEX75.SYSPLEX.ROOT.HFS 020
 FAILED TO UNMOUNT.  RET CODE = 00000072, RSN CODE = 058800AA
 IOEZ00048I Detaching aggregate PLEX75.SC74.SYSTEM.ZFS
*010 BPXI070E     USE SETOMVS ON ANOTHER SYSTEM TO MOVE NEEDED FILE
 SYSTEMS, THEN REPLY WITH ANY KEY TO CONTINUE SHUTDOWN.
```

Figure 16-68 Messages BPXI066E and BPXI070E shown on stopping OMVS on SC74

The problem is that the entire UNIX System Services file system structure is lost when removing the sysplex root in this situation. After replying to the prompt message, the current sysplex root is no longer accessible. It gets automatically replaced by the alternate sysplex root being available; see Figure 16-69 on page 378.

```
 R 10,
 IEE600I REPLY TO 010 IS;
...
 BPXN001I UNIX SYSTEM SERVICES PARTITION CLEANUP IN PROGRESS 580
 FOR SYSTEM SC74
...
 BPXF213E FILE SYSTEM PLEX75.SYSPLEX.ROOT.HFS 581
 IS NO LONGER ACCESSIBLE.
 BPXN002I UNIX SYSTEM SERVICES PARTITION CLEANUP COMPLETE 582
 FOR SYSTEM SC74
 IOEZ00048I Detaching aggregate PLEX75.SYSPLEX.ROOT.ZFS
 BPXF063I FILE SYSTEM PLEX75.SYSPLEX.ROOT.ZFS 584
 WAS SUCCESSFULLY UNMOUNTED.
...
*BPXF253E ALTROOT INACTIVE: 587
 ALTROOT IS NOW ACTIVE AS CURRENT SYSPLEX ROOT.
 BPXF256I PLEX75.SYSPLEX.ROOT.ZFS IS NOW ACTIVE AS CURRENT SYSPLEX
 ROOT.
...
*BPXI056E OMVS SHUTDOWN REQUEST HAS COMPLETED SUCCESSFULLY
...
 D OMVS,O
 BPXO043I 18.08.02 DISPLAY OMVS 592
 OMVS     0010 ACTIVE             OMVS=(1A)
...
 ALTROOT         =
```

Figure 16-69 Automatic replacement of the sysplex root when it is no longer accessible

In this situation it is good practice to establish a new alternate sysplex root, to be prepared for any future sysplex root problems.

16.9 CINET pre-router changes

The Common INET pre-router provides a multiple transport provider environment for applications and users. The CINET pre-router learns interface list and routing information from each transport provider and makes routing decisions based on that information.

The CINET pre-router is enhanced to provide support for multiple IPv4 and IPv6 default routes. This provides accurate and efficient default route routing. Previously multiple IPv4 and IPv6 default routes from multiple transport providers were not properly handled.

Using Multiple Default Routes Support, the pre-router can select the best default route for routing, thus providing accurate and efficient default route routing.

16.9.1 Changed default route selection

When searching for default routes, if there are multiple active default routes from multiple transport providers, the best route type and route metric default route is selected. If there are no active default routes, the default transport provider is selected.

Note: The default transport provider is specified with the DEFAULT keyword in the SUBFILESYSTYPE statement in the BPXPRMxx parmlib member.

16.9.2 Displaying information in z/OS V1R10

Figure 16-70 shows the output of system command **D OMVS,CINET** in a system running z/OS V1R10.

```
D OMVS,CINET=ALL
BPXO047I 22.31.04 DISPLAY OMVS 935
OMVS     0011 ACTIVE             OMVS=(1A)
INADDRANYPORT: 10000  INADDRANYCOUNT: 2000
IPV4 HOME INTERFACE INFORMATION
 TP NAME  HOME ADDRESS        FLAGS
  TCPIP     127.000.000.001    DRS
  TCPIP     009.012.006.009    DRS
  TCPIP     009.012.004.102    DRS
  TCPIP     009.012.006.031    DRS
  TCPIP     010.001.101.064    DRS
  TCPIP     010.001.101.064    DRS
```

Figure 16-70 Running command D OMVS,CINET=ALL in z/OS V1R10

16.9.3 Displaying information in z/OS V1R11

Figure 16-71 on page 380 shows the output of system command **D OMVS,CINET** in a system running z/OS V1R11.

Note: The DRS flag indicator for default routes is no longer displayed on the output of the **D OMVS,CINET** command.

```
D OMVS,CINET=ALL
BPXO047I 22.26.21 DISPLAY OMVS 822
OMVS     0011 ACTIVE             OMVS=(1B)
INADDRANYPORT: 10000  INADDRANYCOUNT: 2000
IPV4 HOME INTERFACE INFORMATION
 TP NAME  HOME ADDRESS        FLAGS
  TCPIP     127.000.000.001
  TCPIP     009.012.004.202
  TCPIP     009.012.004.102
  TCPIP     009.012.004.203
  TCPIP     010.001.101.070
  TCPIP     010.001.101.070

IPV4 HOST ROUTE INFORMATION
 TP NAME  HOST DESTINATION     TYPE     METRIC
  TCPIP     127.000.000.001    STAT          0
  TCPIP     009.012.004.202    STAT          0
  TCPIP     009.012.004.102    STAT          0
  TCPIP     009.012.004.203    STAT          0
  TCPIP     010.001.101.063    STAT          0
  TCPIP     010.001.101.064    STAT          0
  TCPIP     010.001.101.065    STAT          0
  TCPIP     010.001.101.070    STAT          0
IPV4 NETWORK ROUTE INFORMATION
 TP NAME   NET DESTINATION     NET MASK           TYPE       METRIC
  TCPIP     009.012.004.000    255.255.252.000    STAT           0
  TCPIP     009.012.004.000    255.255.252.000    STAT           0
  TCPIP     010.001.101.000    255.255.255.000    STAT           0
  TCPIP     000.000.000.000    000.000.000.000    STAT           1
  TCPIP     000.000.000.000    000.000.000.000    STAT           1
```

Figure 16-71 Running command D OMVS,CINET=ALL in z/OS V1R11

16.10 UNIX System Services file system enhancements

The following enhancements have been added, which resolve several problems that needed to be addressed:

- AUTOMOUNT has been changed to discontinue the automated unmount attempts after a hard unmount failure occurs. This now allows manual intervention.
- A new osi_ctl() function has been provided to enable physical file systems to provide PFS-specific status information.
- The user, date and time, system, and master configuration file for the automount command are audited and displayed. The command **D OMVS,PFS** has been changed to display this information.
- A syslog message is issued when an unowned file system has been recovered.
- The output of **D OMVS,WAITERS** has been changed to include the date and time the activity was started.

The enhancements provide improved usability and easier problem determination.

- An automount managed file system can be manually unmounted when it failed to unmount.
- The user can be notified when the automount policy was changed.

- An interface is provided for a PFS to give PFS-specific status information. You can display PFS status information with the **D OMVS,PFS** command.
- A message indicates the recovery of an unowned file system.
- The **D OMVS,WAITERS** command indicates the date and time when the activity started.

16.10.1 AUTOMOUNT unmount

When an automount unmount fails due to a PFS-related problem, message BPXF250I is issued:

```
BPXF250I AUTOMOUNT FACILITY CANNOT UNMOUNT FILE SYSTEM filesystem_name
```

16.10.2 AUTOMOUNT information

After running the automount command, if the automount policy has been changed, message BPXF260I is displayed in syslog. This message includes the following information:

- Date and time the automount command was run
- System name where the automount command was run
- User ID that executed the automount command
- Pathname or data set name of the automount policy used

Figure 16-72 shows an example of the messages displayed if automount is run in a system running z/OS V1R11.

```
BPXF260I AUTOMOUNT POLICY WAS CHANGED AT 2009/05/14 22:56:44
BY USER HERING ON SYSTEM SC70 WITH POLICY
/etc/auto.master
```

Figure 16-72 Message BPXF260I shown if automount command is run in z/OS V1R11

In a sysplex sharing environment, if automount was run in a system running with a prior release, message BPXF261I is issued; see Figure 16-73.

```
BPXF261I AUTOMOUNT POLICY WAS CHANGED AT 2009/05/14 23.09.00
BY A MEMBER SYSTEM RUNNING AT A PRIOR RELEASE OF zOS.
```

Figure 16-73 Message BPXF261I shown if automount command is run in z/OS V1R10

Note: The messages BPXF260I and BPXF261I are only shown in z/OS V1R11 (or above).

16.10.3 Displaying OMVS PFS information

The **D OMVS,PFS** command has been changed to display more information.

Figure 16-74 on page 382 shows the old message BPXO046I displayed in z/OS V1R10.

```
D OMVS,P
BPXO046I 23.22.46 DISPLAY OMVS 924
OMVS     0011 ACTIVE             OMVS=(1A)
PFS CONFIGURATION INFORMATION
 PFS TYPE  DESCRIPTION          ENTRY      MAXSOCK  OPNSOCK  HIGHUSED
  NFS      REMOTE FILE SYSTEM   GFSCINIT
  CINET    SOCKETS AF_INET      BPXTCINT    10000       67      1255
  UDS      SOCKETS AF_UNIX      BPXTUINT    10000        5         6
  AUTOMNT  LOCAL FILE SYSTEM    BPXTAMD
  TFS      LOCAL FILE SYSTEM    BPXTFS
  ZFS      LOCAL FILE SYSTEM    IOEFSCM
  HFS      LOCAL FILE SYSTEM    GFUAINIT
  BPXTXPFS CROSS SYSTEM PFS     BPXTXPFS
  BPXFTCLN CLEANUP DAEMON       BPXFTCLN
  BPXFTSYN SYNC DAEMON          BPXFTSYN
  BPXFPINT PIPES                BPXFPINT
  BPXFCSIN CHARACTER SPECIAL    BPXFCSIN

 PFS NAME    DESCRIPTION           ENTRY      STATUS  FLAGS
  TCPIP       SOCKETS              EZBPFINI   ACT     SC
  TCPIPOE     SOCKETS              EZBPFINI   INACT
  TCPIPMVS    SOCKETS              EZBPFINI   INACT
...

 PFS TYPE    PARAMETER INFORMATION
  ZFS         PRM=(64,00)
  HFS
              CURRENT VALUES: FIXED(0) VIRTUAL(2010)
```

Figure 16-74 Old message BPXO046I displayed in z/OS V1R10

Figure 16-75 on page 383 shows the new message BPXO068I displayed in z/OS V1R11.

```
D OMVS,P
BPXO068I 23.30.14 DISPLAY OMVS 025
OMVS     0011 ACTIVE             OMVS=(1B)
PFS CONFIGURATION INFORMATION
 PFS TYPE   ENTRY      ASNAME    DESC      ST    START/EXIT TIME
  NFS       GFSCINIT   MVSNFSC7  REMOTE    A     2009/05/11 00.13.36
  CINET     BPXTCINT             SOCKETS   A     2009/05/11 00.13.36
  UDS       BPXTUINT             SOCKETS   A     2009/05/11 00.13.36
  AUTOMNT   BPXTAMD              LOCAL     A     2009/05/11 00.13.36
  TFS       BPXTFS               LOCAL     A     2009/05/11 00.13.36
  ZFS       IOEFSCM    ZFS       LOCAL     A     2009/05/11 18.15.04
  HFS       GFUAINIT             LOCAL     A     2009/05/11 00.13.22

 PFS TYPE  DOMAIN        MAXSOCK  OPNSOCK  HIGHUSED
  CINET    AF_INET         10000       46        49
  UDS      AF_UNIX         10000        4         5

 SUBTYPES OF COMMON INET
 PFS NAME   ENTRY      START/EXIT TIME       STATUS  FLAGS
  TCPIP      EZBPFINI   2009/05/11 00.13.46   ACT     SC
  TCPIPOE    EZBPFINI                        INACT
  TCPIPMVS   EZBPFINI                        INACT
...

 PFS TYPE     FILESYSTYPE PARAMETER INFORMATION
  ZFS          PRM=(70,00)
  HFS
               CURRENT VALUES: FIXED(0) VIRTUAL(2010)

 PFS TYPE  STATUS INFORMATION
  AUTOMNT  TIME=2009/05/14 23:10:46 SYSTEM=SC70     USER=HERING
           POLICY=/etc/auto.master
```

Figure 16-75 New message BPXO068I displayed in z/OS V1R11

16.10.4 Unowned file system information

In a shared file system configuration, message BPXF213E is issued to advise that a file system is not accessible.

```
BPXF213E FILE SYSTEM filesystem_name IS NO LONGER ACCESSIBLE.
```

Message BPXF251I will be issued by the new owning system when an unowned file system has been recovered and becomes active again.

```
BPXF251I FILE SYSTEM filesystem_name HAS BEEN RECOVERED AND IS NOW ACTIVE.
```

In a sysplex environment, an unowned file system is recovered by the Blackhole Daemon. This daemon wakes up periodically and looks for file systems eligible for takeover (for example, in case of AUTOMOVE=Y). Restarting the original owner also recovers unowned file systems.

16.10.5 Changed OMVS waiters information

The output of **D OMVS,WAITERS** has been changed to indicate the date and time when the activity started; see Figure 16-76.

```
D OMVS,WAITERS
BPXO063I 23.57.29 DISPLAY OMVS 037
OMVS     0011 ACTIVE             OMVS=(1B)
MOUNT LATCH ACTIVITY: NONE
FILE SYSTEM LATCH ACTIVITY: NONE
OUTSTANDING CROSS SYSTEM MESSAGES: NONE

OTHER WAITING THREADS:
    USER   ASID     TCB               PID                        AGE
-------------------------------------------------------------------------
  D9D4ADMT 006A  007B7828          196663                     00.00.00
     TIME: 2009/05/14 23.57.28
     IS DOING: Osi Wait
  D9D4ADMT 006A  007CCD90          196663                     95.43.09
     TIME: 2009/05/11 00.14.19
     IS DOING: Osi Wait
  FTPMVS1  005D  007FF1D8          196662                     52.58.48
     TIME: 2009/05/12 18.58.40
     IS DOING: Osi Wait
  PMAP     0064  007CED90        33751092                     35.38.13
     TIME: 2009/05/13 12.19.15
     IS DOING: Osi Wait
```

Figure 16-76 Running D OMVS,W in z/OS V1R11

For all activities and waiting threads displayed, the extra line starting with `TIME:` has been added as additional information. Figure 16-77 shows the old display taken from a z/OS V1R10 system.

```
D OMVS,WAITERS
BPXO063I 09.19.04 DISPLAY OMVS 050
OMVS     0011 ACTIVE             OMVS=(1A)
MOUNT LATCH ACTIVITY: NONE
FILE SYSTEM LATCH ACTIVITY: NONE
OUTSTANDING CROSS SYSTEM MESSAGES: NONE

OTHER WAITING THREADS:
    USER   ASID     TCB               PID                        AGE
-------------------------------------------------------------------------
  GPMSERVE 007F  007CF178        67371181                     00.00.27
     IS DOING: Osi Wait
  GPMSERVE 007F  007CF750        67371181                     00.00.27
     IS DOING: Osi Wait
  MVSNFSC7 0059  007CE300        50593837                     00.00.15
     IS DOING: Osi Wait
```

Figure 16-77 Running D OMVS,W in z/OS V1R10

16.10.6 Migration and coexistence considerations

In a UNIX System Services sysplex sharing environment, if systems run with mixed releases of z/OS, then operators will receive responses and see behavior differing for scenarios similar to those described.

16.11 User syscall trace

Until now, the z/OS UNIX application environment did not provide a simple way to trace application activity. Users had to put printf() statements in their applications, for example, to trace events.

The enhancement described here enables a user to capture a z/OS UNIX System Services system call trace. This simplifies both development and problem determination for applications running in the z/OS UNIX environment.

Using user syscall trace, you can:

- Capture system call trace data for an already-running application
- Start up an application with system call tracing activated

This provides the following benefits:

- It simplifies development and problem determination for z/OS UNIX-based applications.
- It simplifies problem determination for preexisting and new applications.
- It mitigates the need to alter applications to gather trace data.

16.11.1 Using the user syscall trace

- You can use the new command **BPXTRACE** to start an application with tracing, or to capture trace data for an already-running application.
- You can use the SIGTRACE signal to activate or deactivate tracing for a running application.
- You can display the tracing status of a process from a shell session by using the **ps** command, or from an operator console by using **D OMVS**.

16.11.2 BPXTRACE command

This is a new command for capturing a z/OS UNIX system call trace. It is available from both TSO and z/OS UNIX shell sessions.

You can use this command to perform these tasks:

- Start a z/OS UNIX application with system call tracing active
- Activate system call tracing against an already- running z/OS UNIX application or set of applications
- Capture system call frequency data for an application

Note: Using the **bpxtrace** command is the preferred method for performing trace functions.

Starting a z/OS UNIX application with system call tracing active

Figure 16-78 shows an example of running a command with tracing active.

```
$> bpxtrace -c -o /tmp/hering.trace.output sleep 5
$> cat /tmp/hering.trace.output | head
       PID ASID TCB    Local time      System call           Additional trace data
     65900 0052 7FF1D8 12:49:16.625074 Exit stat             rv=00000000
     65900 0052 7FF1D8 12:49:16.625078 Call access           parms: 00000009 /bin/test
00000401 12724D80 00000081 053B006C
     65900 0052 7FF1D8 12:49:16.625247 Exit access           rv=00000000
     65900 0052 7FF1D8 12:49:16.625259 Call sigaction        parms: 00000002 1275E940
00000000 00000000 12724AC0 00000000 00000000 1275E040 06C716E8 00000000 053B006C
     65900 0052 7FF1D8 12:49:16.625260 Exit sigaction        rv=00000000
     65900 0052 7FF1D8 12:49:16.625269 Call sigaction        parms: 00000002 1275E928
00000000 00000000 12724AB8 12719B48 12724CD8 1275E040 00000000 00000000 053B006C
     65900 0052 7FF1D8 12:49:16.625271 Exit sigaction        rv=00000000
     65900 0052 7FF1D8 12:49:16.625295 Call close            parms: 0000000B 00000000
00000000 053B006C
     65900 0052 7FF1D8 12:49:16.625360 Exit close            rv=00000000
$>
```

Figure 16-78 Running a command with tracing being active

When starting a z/OS UNIX application with system call tracing active, the BPXTRACE command is used with option **–c**. Use option **–o** to direct the output to a z/OS UNIX file. The default is to direct the output to STDOUT. Because this can be a significant amount of data, always route it to a file.

The trace output for each system call shows the following information:

- Process ID
- ASID
- TCB
- Time involved with the system call

The entry and exit to each system call is traced:

- The entry record shows the parameter data.
- The output shows the return value from the system call.

Starting the trace against a running z/OS UNIX application

The **bpxtrace** command can be used to start a trace against an already-running z/OS application or set of applications.

- You can use option **–s** n conjunction with option **–p** to start the trace against an individual process:

  ```
  bpxtrace -s -p $$
  ```

- You can specify a list of process IDs by using option **-p** more than one time:

  ```
  bpxtrace -s -p pid1 –p pid2
  ```

- You can use option **–u** to start a trace against all processes running for the specified user ID:

  ```
  bpxtrace -s -u hering
  ```

Note: Using specification **$$** in a shell command refers to the actual process ID.

Capturing the trace output for an application being traced

You can use option **-i** to capture the trace output for application being traced. The output can be directed to a z/OS UNIX file using the option **–o**.

```
bpxtrace -i -p $$ -o /tmp/hering.trace.output
```

Ending the trace for a running application

You can use option **-e** to end the trace against a running application. When ending a trace, the trace output can also be directed to a z/OS UNIX file by using the **-o** option.

```
bpxtrace -e -p pid -o /tmp/hering.trace.output
```

Capturing system call frequency data for a running application

You can use the **bpxtrace** command to capture system call frequency data for a z/OS UNIX application to help tune an application.

You can use option **–f counts** to capture the system call frequency data, as shown in Figure 16-79.

```
$> bpxtrace -e -p $$ -o /tmp/hering.trace.output -f counts
$> cat /tmp/hering.trace.output
1JOBNAME HERING3  STEPNAME STEP1          SYSTEM EC6    ...   1  14:36:54 05/17/09
+_____________________________________________________________________________ ...
0 COMPONENT TRACE SHORT FORMAT
  COMP(SYSOMVS)
  OPTIONS((SYSCALL SCCOUNTS))
      Syscall    Syscall Name   Function Name            Count    Syscalls/Sec
      00000019     BPX1FST      fstat                     24     ******
      0000001F     BPX1LSK      lseek                     18     ******
      00000057     BPX1SPM      sigprocmask               16     ******
      0000002B     BPX1RED      read                      13     ******
      0000000C     BPX1ACC      access                    10     ******
      00000064     BPX1EX2      i.exec2                   10     ******
      00000011     BPX1CLO      close                      4     ******
      0000004C     BPX1KIL      kill                       4     ******
      00000026     BPX1OPN      open                       4     ******
      0000000B     BPX1TSP      tcsetpgrp                  2     ******
      00000056     BPX1SPG      setpgid                    2     ******
      0000004D     BPX1MSS      mvssigsetup                2     ******
      000000BD     BPX1SPN      spawn                      2     ******
      00000036     BPX1WRT      write                      2     ******
      00000050     BPX1SIA      sigaction                  2     ******
      0000005E     BPX1WAT      wait                       1     ******
      FuncCode   FuncCode Name       Count    Functions/Sec
$>
```

Figure 16-79 capturing system call frequency data

A list of system calls along with their count and frequency per second is reported.

Signal SIGDUMP forcing an abend EC6

Using the **bpxtrace** command generates a SIGDUMP signal to generate a dump to capture the trace data.This will result in an abend EC6 with reason code 0D2FFD27 being displayed in the system log; see Figure 16-80 on page 388.

```
SYSTEM COMPLETION CODE=EC6  REASON CODE=0D2FFD27
 TIME=12.49.21  SEQ=14974  CPU=0000  ASID=0056
 PSW AT TIME OF ERROR  070C1000   866603F4  ILC 2  INTC 0D
   NO ACTIVE MODULE FOUND
   NAME=UNKNOWN
   DATA AT PSW  066603EE - C02818F5  0A0D98EC  D00C07FE
   GR 0: 00000000   1: 04EC6000
      2: 9275897A   3: 821B1680
      4: 00000000   5: 0D2FFD27
      6: 7F5715E8   7: 7F605006
      8: 125128F0   9: 00000000
      A: 125128F0   B: 7F571C00
      C: 06660768   D: 7F605200
      E: 8665F3C8   F: 0D2FFD27
 END OF SYMPTOM DUMP
```

Figure 16-80 Abend EC6 with reason code 0D2FFD27

16.11.3 Using the SIGTRACE signal

The z/OS UNIX shell and tcsh shell both added support for the new signal, SIGTRACE. You can use the **kill** command to send the SIGTRACE signal to a process, which simply toggles the trace between activate and deactivate.

```
kill -s TRACE $$
```

Further information and notes for SIGTRACE

- Signal SIGTRACE is not a catchable signal.
- Within a shell environment, the SIGTRACE signal has little use outside the **kill** command.
- The SIGTRACE signal will not be returned as a wait/exit status value.
- The SIGTRACE signal will not be “received,” so a trap condition for SIGTRACE, established by the **trap** command, will never be raised.
- From a C application use functions raise() or kill() to send a SIGTRACE signal to a given process.
- From Assembler, use the callable service kill (BPX1KIL, BPX4KIL) BPX1KIL to send signal SIGTRACE.

16.11.4 Displaying trace setting

A new process attribute support has been added that indicates whether tracing is activated.

Support for the new process attribute on the ps command

Using the format specification **-o attr**, the **ps** command output indicates the process attribute value **T** when the trace is activated.

Figure 16-81 on page 389 shows an example using this format specification.

```
$> ps -o pid,attr,args
       PID ATTR COMMAND
  16843165 T    ps -o pid,attr,args
     66006 -    sh -L
  83952088 -    /bin/sh -S -- 4cmds na
  16843225 T    sh
$>
```

Figure 16-81 Using command ps for displaying trace status

Support for the new process attribute on D OMVS output

You can use the **D OMVS** command to display process attributes for a process. The new process attribute value **T** indicates that the trace is activated.

Figure 16-82 shows an example of using **D OMVS** to provide this information.

```
D OMVS,U=HERING
BPXO070I 15.40.18 DISPLAY OMVS 695
OMVS     0010 ACTIVE             OMVS=(BA)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
HERING   HERING2  0058   83952030   16843225 1CI---T- 15.39.58      .0
  LATCHWAITPID=         0 CMD=go
HERING   HERING   001F   16843221          1 MR------ 14.44.41     6.3
  LATCHWAITPID=         0 CMD=OMVS
HERING   HERING   001F      66006   16843221 1W------ 14.44.41     6.3
  LATCHWAITPID=         0 CMD=sh -L
HERING   HERING   001F   83952088      66006 1W------ 14.44.47     6.3
  LATCHWAITPID=         0 CMD=/bin/sh -S -- 4cmds na
HERING   HERING1  0041   16843225   83952088 1WI---T- 14.44.47     2.7
  LATCHWAITPID=         0 CMD=sh
```

Figure 16-82 Using MVS system command D OMVS for displaying of trace status

The output indicates the trace status in column 7 of the **STATE** field.

16.12 UNIX System Services kernel RAS enhancements

This section provides information about two UNIX System Services RAS enhancements:

- Compatibility of service getthent() with getpsent()
- SAF error information enhancements

16.12.1 Compatibility of service getthent() with getpsent()

Callable service __getthent() (BPX1GTH | BPX4PGTH) allows up to 2048 bytes of data to be returned for the path and command or argument data. For the path name, 1024 bytes are reserved and 1024 bytes are reserved for the command and its arguments.

If the path is not requested or does not take up the full 1024 bytes, then the data for the command and its argument is still limited to 1024 bytes. If there is more room, the data for the command and its argument should be allowed to expand to greater than 1024 bytes.

This is addressed by a new option in the __getthent() PgthA input area to request that greater than 1024 bytes of data can be returned for the command and its arguments.

Benefit of the enhancement for getthent()

Service w_getpsent() (BPX1GPS) is replaced with __getthent(). No enhancements are being made to BPX1GPS and there is no AMODE64 call for it. The __getthent() service needs to be able to include all the function that BPX1GPS provided, namely to allow the data for the command and its arguments to be greater than 1024 bytes when space in the output buffer permits this.

The enhancement for __getthent() exactly provides:

- The necessary compatibility with the sunset w_getpsent() (BPX1GPS) service.
- This can be exploited by AMODE31 applications using the w_getpsent() command and wanting to move to AMODE64.

16.12.2 SAF error information enhancements

z/OS UNIX System Services uses SAF services to authenticate users and to determine users' access to resources. A single z/OS UNIX System Service can use several SAF services for a single request.

When an SAF service is unsuccessful and results in the z/OS UNIX System Service also being unsuccessful, often the z/OS UNIX return values do not provide enough information to identify the unsuccessful SAF service and all of its return codes. Without this key information, it is often impossible to determine the root cause of the SAF error.

With the enhancement, users who need to resolve a security-related problem with z/OS UNIX System Services can get the desired information to solve the problem.

Note: This is intended to reduce the number of security-related non-defect PMRs.

Description of the enhancement

One of the following services might be unsuccessful due to a SAF failure and the information returned by the service might not provide the caller with enough information to identify the SAF service and the cause of the failure:

- __passwd (BPX1PWD)
- getgrname (BPX1GGN)
- getgroups (BPX1GGR)
- setuid()/seteuid/setreuid (BPX1SUI)
- pthread_security_np (BPX1TLS)
- __login (BPX1SEC)

z/OS UNIX System Services will then save information about the SAF service error and the z/OS UNIX System Service in new fields of the THLI control block, and create a CTRACE EXCEPTION record that identifies the SAF error and the z/OS UNIX System Service.

Displaying the z/OS UNIX SAF error CTRACE

You can use the **`BPXTRACE -S`** command to display z/OS UNIX SAF error CTRACE data.

- BPXTRACE is a REXX exec that can be run from a UNIX System Services shell or TSO.

- EUID 0 users receive all SAF exception CTRACE records.
- Users with EUID<>0 receive SAF exception CTRACE records matching their EUID value.
- The **bpxtrace -S -o myctrace.data** command writes the data into the `myctrace.data` file in the current working directory.

16.13 Automatic UID and GID assignment

Using the UNIX System Services default segment for UID and GID has several limitations. Manually assigning UIDs and GIDs is administrative effort.

In z/OS V1R11, RACF can define “on-demand” unique UID and GID values for users and groups without existing OMVS segments. This provides a useful alternative to assigning identifiers in advance using **RACF ADDUSER/ALTUSER** and **ADDGROUP/ALTGROUP** commands.

Note: This new function is contained in several callable services that are invoked by various UNIX system services.

16.13.1 New automatic assignment using BPX.UNIQUE.USER

To exploit this new possibility, define the new FACILITY profile BPX.UNIQUE.USER and optionally, a user profile in APPLDATA field as shown in Figure 16-83.

Use either:

```
RDEFINE FACILITY BPX.UNIQUE.USER
```

Or:

```
RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('USER01')
```

Figure 16-83 Creating new FACILITY profile BPX.UNIQUE.USER

For a user or group without an OMVS segment, the service will create and store a unique UID or GID. If a user name is specified in APPLDATA, its other OMVS fields are copied to the target user when the new UID is saved.

In this situation the USP bit USP_DEFAULT, indicating “default used”, will not be set.

The caller will not know whether the UID/GID pair returned previously existed or was newly created.

Note: Callable service externals are unchanged.

You will never have a combination of BPX.UNIQUE.USER and BPX.DEFAULT.USER being used; it is always one or the other.

Note: After the profile BPX.UNIQUE.USER is created, the default UID will no longer be assigned.

16.13.2 UID and GID value range

The same code logic is used as for the automatic assignment done for command processors when creating a new UID automatically, as shown in Figure 16-84.

```
ADDUSER USER01 OMVS(AUTOUID)
```

Figure 16-84 Creating a new UID automatically

This causes a lookup in BPX.NEXT.USER and ensures uniqueness; Figure 16-85 shows an example.

```
RALTER FACILITY BPX.NEXT.USER APPLDATA('10000/80000')
```

Figure 16-85 Setting up BPX.NEXT.USER APPLDATA

Leaving the UID range blank or using “NOAUTO” indicates that the default information is not to be used for the user. The same is true for GID.

Note: This allows automatic assignment for just UIDs or just GIDs or both.

Note the following requirements to be able to use this function:

- The RACF database must be at level AIM stage 3.
- Profile SHARED.IDS in class UNIXPRIV.

Note: This technique guarantees uniqueness in the scope of the GRS complex.

Specifying NOAUTO as a qualifier in the APPLDATA, or omitting the qualifier, prevents automatic ID assignment. For example, if you use employee serial numbers as a convention in assigning UIDs and you do not want to use automatic assignment, but you do want to use automatic GID assignment starting at 3000, issue:

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('NOAUTO/3000')
```

Using ranges can be useful in an RRSF environment. Specify a starting and ending value separated by a dash (–) if you want to include a range of values. Both values must be valid UID or GID values, and the second value must be greater than the first. Ranges can be specified independently for UIDs or GIDs, as shown in Figure 16-86.

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000-80000/3000-10000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000/3000-10000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000-80000/3000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('NOAUTO/3000-10000')
```

Figure 16-86 Defining UID and GID ranges for BPX.NEXT.USER

16.13.3 Installation of the new function

To enable the new function, perform these tasks:

- Define FACILITY class profile BPX.UNIQUE.USER.
- Define UNIXPRIV class profile SHARED.IDS.

- Define FACILITY class profile BPX.NEXT.USER with valid ID values or ranges in its APPLDATA field.
- Enable the database for application identity mapping (AIM) stage 3.

16.14 dbx demand load

You can use a new utility, **dbgld**, to create a module level debug side file. dbx can use this information to load only the debug information that it needs, thereby significantly reducing the debugger start-up time.

Without this utility, 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 will occur every time the debug session is restarted.

16.14.1 Advantages of using the dbgld utility

Using the dbgld utility provides several benefits:

- It reduces debugger startup time dramatically.
- Just one debug side file per module needs to be maintained.
- Symbol lookup is faster.
- You need less memory.
- A fewer number of file handles are open.

16.14.2 Using the dbgld utility

The dbgld utility syntax is shown in Figure 16-87.

```
dbgld [ -v ] file
```

Figure 16-87 Syntax of utility dbgld

-v This writes the version information to stderr.

file This is the module name, and it can be one of the following:

- The absolute path name of a z/OS UNIX file
- The relative path name of a z/OS UNIX file
- A fully qualified MVS dataset, either a sequential dataset or a PDS member.

The output of the dbgld utility is a file with the name of the module followed by an extension of .mdbg. The file will always be written in the current directory. For example, if the module name is `/mypath/mymodule`, then a file called `mymodule.mdbg` will be created in the current directory. If the file already exists, it will be overwritten.

Note: dbx achieves this performance boost by calling the Demand Load APIs provided by Common Debug Architecture runtime.

Figure 16-88 shows an example of calling dbx without using dbgld. Message FDBX9997 is shown because the utility has not been used to create a module map first.

```
$> cc -go hello hello.c
$> echo quit | time dbx ./hello
FDBX0089: dbx for z/OS with 64-bit support.
FDBX0399: Compiled: Jan 29 2009 at 12:34:57 (v1.11)
FDBX0400: OS level: 11.00 01; LE level: 4.1.11; (Local)
FDBX6499: CDA levels: ELF=D0911.20090310, DWARF=D0911.20090310,
                      DDPI=D0911.20090310
FDBX0100: Type 'help' for help.
FDBX0048: Set an event like 'st in main' then use 'c' to start debugging.
FDBX6432: Processing load module "./hello"
FDBX9997: The loaded module does not contain module map which may lead to bad
performance. Suggest to use dbgld to create module map to the executable before
debugging.
FDBX6421: Loaded debug data from "/u/hering/cc/hello.dbg"
FDBX0150: Debug target is 31-bit.


real    0m 2.19s
user    0m 0.02s
sys     0m 0.00s
$>
```

Figure 16-88 Calling dbx without using utility dbgld

Finally, Figure 16-89 shows the same dbx call after using dbgld.

```
$> dbgld ./hello
$> ls -1 hello hello.mdbg
hello
hello.mdbg
$> echo quit | time dbx ./hello
FDBX0089: dbx for z/OS with 64-bit support.
FDBX0399: Compiled: Jan 29 2009 at 12:34:57 (v1.11)
FDBX0400: OS level: 11.00 01; LE level: 4.1.11; (Local)
FDBX6499: CDA levels: ELF=D0911.20090310, DWARF=D0911.20090310,
                      DDPI=D0911.20090310
FDBX0100: Type 'help' for help.
FDBX0048: Set an event like 'st in main' then use 'c' to start debugging.
FDBX6432: Processing load module "./hello"
FDBX0150: Debug target is 31-bit.


real    0m 1.20s
user    0m 0.02s
sys     0m 0.00s
$>
```

Figure 16-89 Calling dbx after using utility dbgld

17

ISPF enhancements

This chapter describes a number of ease-of-use enhancements that have been made to ISPF for z/OS V1R11. This chapter describes the ISPF enhancements introduced in z/OS V1R11 as listed here:

- Edit COMPARE command: SYSIN data set enhancement
- New Edit line commands for HEX display
- DSLIST: Support for data set name level prefix
- z/OS UNIX System Services directory list enhancements
- Panel source statement input exit
- Extended address volume (EAV) data set support
- Extended member statistics
- Configure qualifiers for PDF utility output data sets
- Display SVC 99 return code information for dynamic allocation failures

17.1 ISPF Edit COMPARE command

Sometimes it is necessary to change the compare behavior for the edit **COMPARE** command because it only compares data between specific columns. Prior to z/OS V1R11, a user was required to preallocate a data set for a SYSIN DD containing SuperC process statements that alter the comparison before issuing the ISPF **COMPARE** command.

The ISPF editor **COMPARE** command is enhanced to allow the name of a data set containing SuperC process statements and allocated to the SYSIN DD to be specified as part of the command syntax. This new support has two new ISPF Editor line commands, **HX** and **HXX**, to allow the display of individual records in hexadecimal format.

The COMPARE command

The **COMPARE** command is used to compare a member, data set or z/OS UNIX file being edited with another member, data set, or z/OS UNIX file. The new syntax of the **COMPARE** command is shown in Figure 17-1, where **dsname** is the name of a member, data set, or z/OS UNIX file to which the current file is compared. The new parameter is:

```
supercdsname The name of a data set containing SuperC process statements.
```

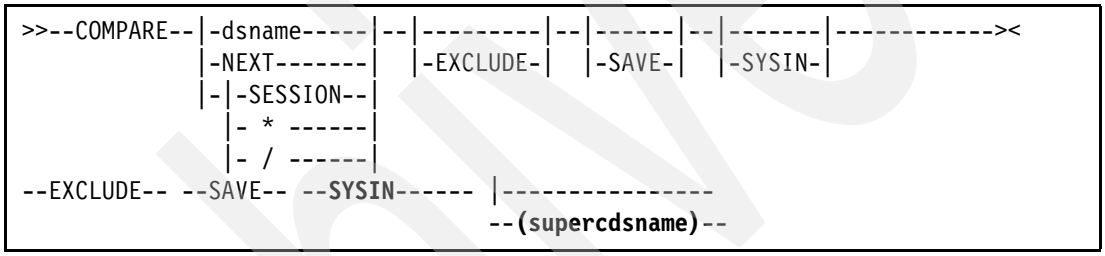


Figure 17-1 Syntax of the COMPARE command

In z/OS V1R11, the edit **COMPARE** command syntax is enhanced to allow the user to specify a data set containing additional SuperC process statements (see Figure 17-1) without the need to preallocate the data set to the SYSIN DD. This enhancement improves the usability of the ISPF editor.

SYSIN keyword

The SYSIN keyword on the **COMPARE** command is enhanced to support a sub-parameter.

- A user can specify the SuperC process statements data set name as a sub-parameter of the SYSIN keyword.
  - Syntax: SYSIN(dsn)
- Alternatively, the user can request the display of a pop-up panel, shown in Figure 17-2 on page 397, where the data set name can be entered that contains SuperC process statements, by specifying the following option on the **COMPARE** command.
  - Syntax: SYSIN(/)

When the SuperC process data set is specified with the **COMPARE** command of ISPF, the command does following:

- It allocates the SuperC process statements data set to the SYSIN DD.
- It invokes SuperC for compare processing.
- It frees the SYSIN DD.

Note: Specifying the SYSIN keyword without a sub-parameter causes the COMPARE command to behave the same as in prior releases.

Figure 17-2 shows a pop-up panel, ISPEUPS, when SYSIN(/) is specified with the **COMPARE** command with a filename, as in the following example:

```
COMPARE filename SYSIN(/)
```

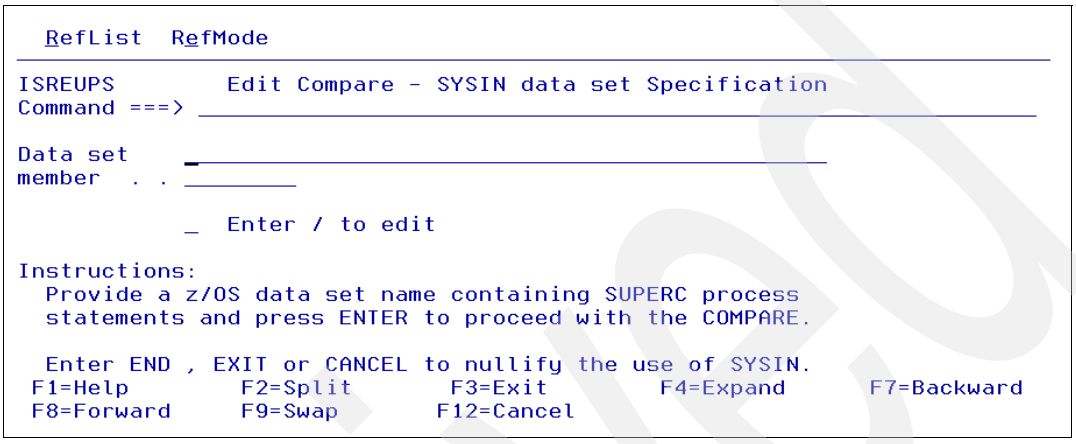


Figure 17-2 Edit COMPARE command SYSIN data set specification

Publication changes

For more information about this topic, *z/OS ISPF Edit and Edit Macros*, SC34-4820.

17.2 New edit line commands for HEX display

In previous releases, ISPF edit provides the **HEX** primary command to allow the user to display and edit data in hexadecimal format. The **HEX** command converts all data into hexadecimal format. The hexadecimal display format uses 4 lines on a panel to display 1 line of data. To address this problem, the ISPF editor is enhanced with z/OS V1R11 to support hex displays. This allows the user to display and edit only selected records in hexadecimal format. This enhancement improves the usability of the ISPF editor.

17.2.1 New line commands for HEX displays

The following new edit line commands display selected records in hexadecimal format:

HX This displays a single record in hexadecimal format.

- **n** - This indicates the number of lines to be displayed in hexadecimal format. If you do not type a number, or if the number you type is 1, then only the line on which you type HX is displayed in hexadecimal format.

HXX This displays multiple records by identifying the start and end of a block of records to be displayed in hexadecimal format. A pair of **HXX** line commands delimits a block of records to be displayed in hexadecimal format.

HX line command usage

The **HX** line command as shown in Figure 17-3 with n=2 (hx2) displays two lines of the file in hexadecimal mode after the Enter key is pressed. The hex display is shown in the second part of the figure.

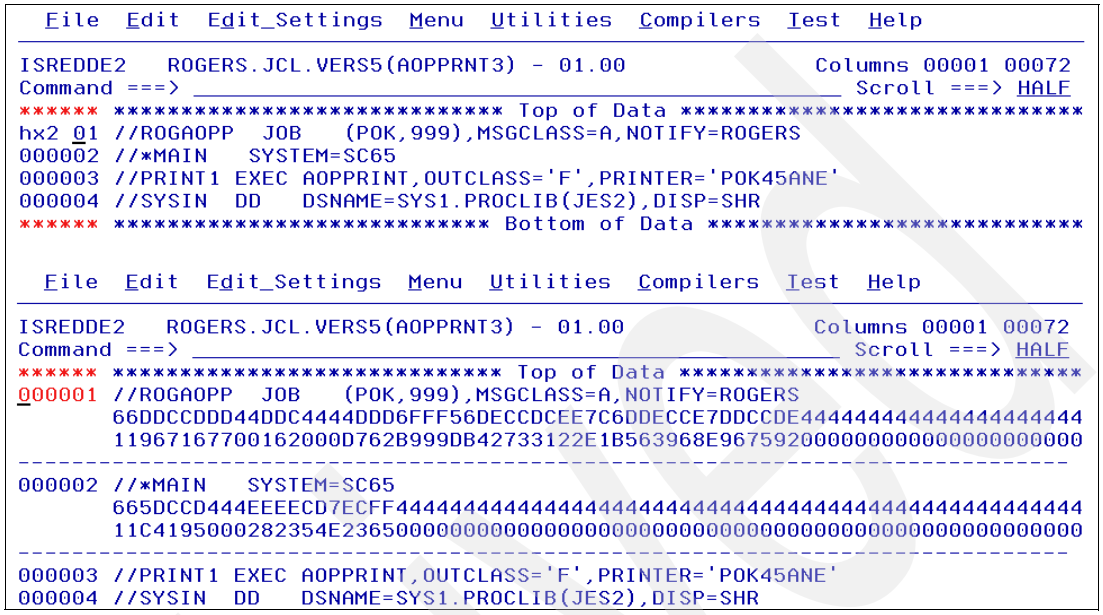


Figure 17-3 HX line command converting to text to hex format

HXX line command usage

The **hxx** line command is used as shown in Figure 17-4 in lines 10 to 13.

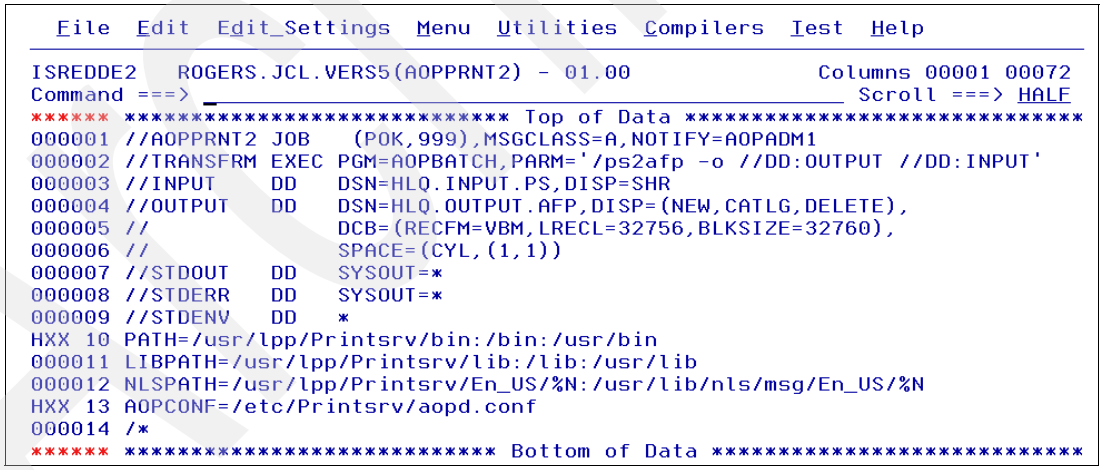


Figure 17-4 Using the hxx line commands to display the line in hex format

This converts those four lines into hex, as shown in Figure 17-5 on page 399.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Iest  Help
ISREDDE2   ROGERS.JCL.VERS5(AOPPRNT2) - 01.00              Columns 00001 00072
Command ===> _                                              Scroll ===> HALF
000003 //INPUT    DD   DSN=HLQ.INPUT.PS,DISP=SHR
000004 //OUTPUT   DD   DSN=HLQ.OUTPUT.AFP,DISP=(NEW,CATLG,DELETE),
000005 //              DCB=(RECFM=VBM,LRECL=32756,BLKSIZE=32760),
000006 //              SPACE=(CYL,(1,1))
000007 //STDOUT   DD   SYSOUT=*
000008 //STDERR   DD   SYSOUT=*
000009 //STDENV   DD   *
000010 PATH=/usr/lpp/Printsrv/bin:/bin:/usr/bin
       DCEC76AA969996D989AA9A68897688976AA9688944444444444444444444444444444444
       7138E1429137717995329512951295A1295A1429129500000000000000000000000000000
-------------------------------------------------------------------------------
000011 LIBPATH=/usr/lpp/Printsrv/lib:/lib:/usr/lib
       DCCDCEC76AA969996D989AA9A69887698876AA9698844444444444444444444444444444
       3927138E1429137717995329513921392A1392A1429139200000000000000000000000000
-------------------------------------------------------------------------------
000012 NLSPATH=/usr/lpp/Printsrv/En_US/%N:/usr/lib/nls/msg/En_US/%N
       DDEDCEC76AA969996D989AA9A6C96EE66D76AA96988699A69A86C96EE66D444444444444
       5327138E1429137717995329515 5D421C5A1429139215321427155D421C5000000000000
-------------------------------------------------------------------------------
000013 AOPCONF=/etc/Printsrv/aopd.conf
       CDDCDDC768A86D989AA9A6899848998444444444444444444444444444444444444444444
       1673656E153317995329511674B3656000000000000000000000000000000000000000000
-------------------------------------------------------------------------------
000014 /*
****** ***************************** Bottom of Data ******************************
```

Figure 17-5 Lines displayed in hex format

17.3 DSLIST support for data set name level prefix

Unlike other ISPF functions, the Data Set List Utility or DSLIST command panel does not automatically include the user's TSO prefix as the first qualifier of the Dsname level. z/OS V1R11 addresses this inconsistency with the Data Set List Utility Prefix Dsname Level option. When selected, this option causes ISPF to automatically include the user's TSO prefix as the first qualifier in the Dsname level; see Figure 17-6 on page 400.

With z/OS V1R11, use this option to control if your TSO prefix is prefixed as the first qualifier of the Dsname Level. When the option is selected, as shown in Figure 17-6 on page 400, your TSO prefix (if running with PREFIX ON) is prefixed as the first qualifier of the Dsname Level if the level is not quoted. When the option is not selected, the Dsname Level is not prefixed, and cannot be enclosed in quotes. This new option, Prefix Dsname Level, is added to the Data Set List Utility entry panel.

Note: If users select this option and they have specified a PREFIX in their TSO profile, then the prefix will be automatically added as the first qualifier for any Dsname Level entered that is not enclosed in single quotes.

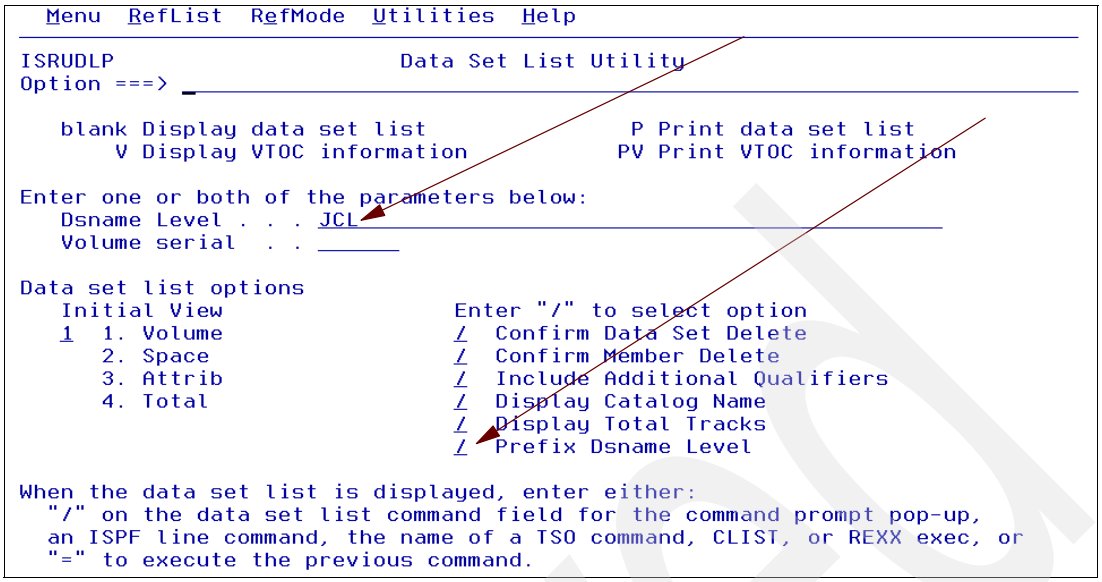


Figure 17-6 DSLIST TSO prefix option

Figure 17-7 shows the data sets that are displayed when Enter is selected in Figure 17-6.

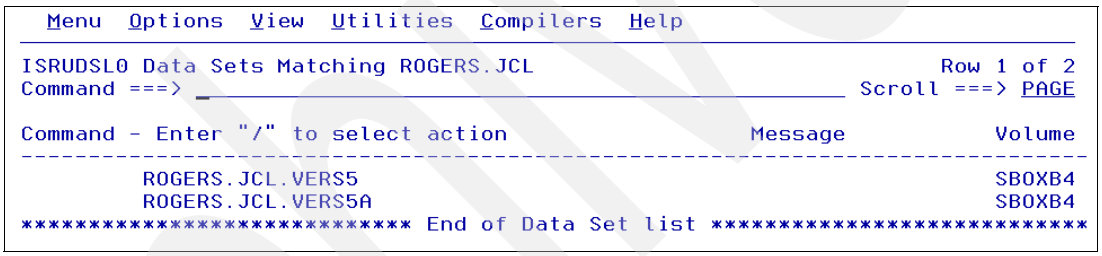


Figure 17-7 Data sets displayed when using qualifier JCL in previous figure

Publication changes

This new feature is described in new “Prefix Dsname level” in *z/OS ISPF User’s Guide Volume II*, SC34-4823.

17.4 z/OS UNIX System Services directory list enhancements

z/OS V1R8 introduced the initial support to ISPF to process z/OS UNIX files that included the z/OS UNIX Directory List Utility. With z/OS V1R11, more functions for processing z/OS UNIX files are added to the z/OS UNIX Directory List Utility.

The z/OS UNIX Directory List Utility is enhanced to support new line commands as functions to display or update information for files and directories.

AA Modify auditor audit options

> **Note:** A user must be defined with AUDITOR authority in the security system to change the auditor audit options for a file or directory.

FS Display file system attributes

MF Modify file format and tags

MG Modify owning group

MO Modify owning user

UA Modify user audit options

Modify auditor audit options (AA)

The Modify z/OS UNIX File Auditor Audit Options panel allows an auditor to define the access attempts that are to be audited by the security system. Figure 17-8 shows the path name in the file system to change the audit selections. The auditor can specify auditing to occur for read, write, and search or execute attempts on the file or directory as shown on Figure 17-10 on page 402.

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
ISREDM01                        Edit Entry Panel
Command ===> ____________________________________________________________

ISPF Library:
   Project . . . ROGERS__
   Group . . . . JCL_____ . . . ________ . . . ________ . . . ________
   Type  . . . . VERS5___
   Member  . . . _______      (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
   Name . . . . . /u/rogers_____________________________________________  +
   Volume Serial . . ______    (If not cataloged)

Workstation File:
   File Name  . . ____________________________________________________

                                  Options
Initial Macro  . . . . ________    /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________    _  Mixed Mode
Format Name  . . . . . ________    _  Edit on Workstation
Data Set Password  . .             _  Preserve VB record length
Record Length  . . . . _____       _  Edit ASCII data
```

Figure 17-8 Accessing a z/OS UNIX directory

This function is invoked using the AA line command, as shown in Figure 17-9, or by using the Modify pull-down menu on the z/OS UNIX File Information panel.

```
  Menu  Utilities  View  Options  Help
ISRUUDL0                z/OS UNIX Directory List          Row 6 to 25 of 27
Command ===> ____________________________________________ Scroll ===> PAGE

Pathname . : /u/rogers

Command  Filename        Message          Type Permission Audit  Ext  Fmat
--------------------------------------------------------------------------
________ .sh_history                      File rw-------  fff--- --s- ----
________ apf.file                         File rwxr-xr-x  fff--- a-s- ----
aa_____  cachex                           Dir  rwxr-xr-x  fff---      ----
________ cachey                           Dir  rwxr-xr-x  fff---      ----
________ cache1                           Dir  rwxr-xr-x  fff---      ----
________ cache2                           Dir  rwxr-xr-x  fff---      ----
________ cbprnt                           File rwx------  fff--- --s- nl
________ fill                             Dir  rwxr-xr-x  fff---      ----
________ fill3                            Dir  rwxr-xr-x  fff---      ----
```

Figure 17-9 Using the AA line command to change the auditing for the specified filename

The Modify z/OS UNIX File Auditor Audit Options panel, shown in Figure 17-10 on page 402, allows an auditor to define the access attempts that will be audited by the security system.

The auditor can specify auditing to occur for Read, Write, and Execute attempts on the file or directory.

Note: A user must be defined with AUDITOR authority in the security system to change the auditor audit options for a file or directory.

```
ISRUULAA       Modify z/OS UNIX File Auditor Audit Options
Command ===> ____________________________________________________________

Pathname . : /u/rogers/cachex
Type . . . : Dir

Read 1  1. None     Write 1  1. None     Execute 1  1. None
        2. Failure           2. Failure             2. Failure
        3. Success           3. Success             3. Success
        4. Both              4. Both                4. Both
```

Figure 17-10 Modify z/OS UNIX File Auditor Audit Options panel

Figure 17-10 displays fields for specifying the Read, Write, and Execute (or search) audit settings. For each of these fields you enter one of the listed numbers corresponding to one of the following results for the access attempt.

None No audit record is to be written for this type of access

Failure Write an audit record if this type of access fails

Success Write an audit record if this type of access is successful

Both Write an audit record for both failed and successful access attempts

Display file system attributes (FS)

The z/OS UNIX File System Attributes panel displays information about a file system for a file or directory; see Figure 17-11. This function is invoked using the **fs** line command, as shown in the figure, or by using the Display pull-down menu on the z/OS UNIX File Information panel.

```
 Menu  Utilities  View  Options  Help
 ISRUUDL0                 z/OS UNIX Directory List          Row 6 to 25 of 27
 Command ===> ____________________________________________ Scroll ===> PAGE

 Pathname . : /u/rogers

 Command  Filename        Message          Type Permission Audit  Ext  Fmat
 ---------------------------------------------------------------------------
 ________ .sh_history                      File rw-------  fff--- --s- ----
 ________ apf.file                         File rwxr-xr-x  fff--- a-s- ----
 fs_______ cachex         Information      Dir  rwxr-xr-x  fff---      ----
 ________ cachey                           Dir  rwxr-xr-x  fff---      ----
 ________ cache1                           Dir  rwxr-xr-x  fff---      ----
 ________ cache2                           Dir  rwxr-xr-x  fff---      ----
```

Figure 17-11 Using the fs line command to display the file attributes

Next, Figure 17-12 on page 403 is displayed showing the attributes of the file.

```
ISRUULFS               z/OS UNIX File System Attributes
Command ===>
                                                                More:     +
Pathname : /u/rogers/cachex

File system name . : OMVS.ROGERS.HFS
Mount point  . . . : /u/rogers

Status . . . . . . : Available
File system type . : HFS
Mount mode . . . . : R/W

Device number  . . : B2C8
Type number  . . . : 1
DD name  . . . . . : SYS00883

Ignore SETUID  . . : NO
Bypass Security  . : NO
Automove . . . . . : YES
Owning system  . . : SC65

CCSID  . . . . . :
Text Convert . . . : NO
Seclabel . . . . . :

Block size . . . . : 4096
Total blocks . . . : 1800
Available blocks . : 1721
Blocks in use  . . : 48

Data blocks read  . . . : 17
Data blocks written . . : 0
Directory blocks r/w  . : 88

Mount parameters
```

Figure 17-12 File system attributes of the file specified using the fs line command

Modify file format and tag (MF)

This function is invoked using the **MF** line command or by using the Modify pull-down menu on the z/OS UNIX File Information panel.

The Modify z/OS UNIX File Format panel allows the user to change the format and tag information for a file by specifying **MF** as a line command on the panel (see Figure 17-11 on page 402), next to a filename cachex. Figure 17-13 on page 404 displays the panel.

The **MF** (modify format) line command can be entered against any directory entry except a symbolic link file. This line command causes the Modify z/OS UNIX File Format panel to be displayed.

The format and tag information can be modified by changing the values of the following fields on this panel.

For the file format, enter one of the listed numbers corresponding to one of the following formats required for the file.

NA No format specified

Binary Binary data

NL Text file; lines delimited by the newline character

CR Text file; lines delimited by the carriage-return character

LF Text file; lines delimited by the line-feed character

CRLF Text file; lines delimited by carriage-return and line-feed characters

LFCR Text file; lines delimited by line-feed and carriage-return characters

CRNL Text file; lines delimited by carriage-return and newline characters

For the CCSID field, enter the numeric coded character set identifier (CCSID) associated with the file. The numeric value must be between 0 and 65535. You can set this field to blanks or entering a value of 0 to indicate there is no CCSID associated with the file.

For automatic conversion, select this option to identify the file as a candidate for automatic conversion provided by z/OS UNIX Enhanced ASCII support.

Note: A superuser or the owner can change the file format of a file. A superuser, the owner, or a user with write permission can change the tag information (CCSID and automatic conversion setting) for a file. File tag information cannot be set for a z/OS UNIX directory. Therefore, when processing a directory, the **CCSID** and **Automatic Conversion** fields are protected.

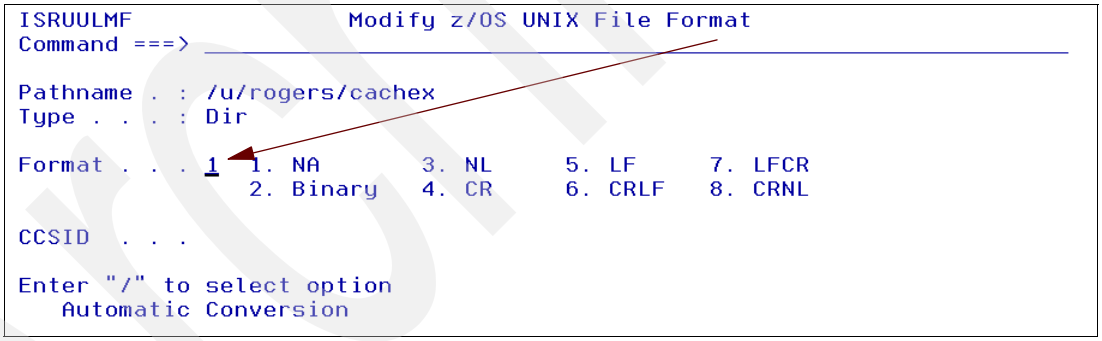


Figure 17-13 Modify z/OS UNIX File Format panel

When you enter a number from 1 to 8 at the point shown by the arrow in Figure 17-13, that field is immediately changed for the file format.

To change the CCSID, enter the numeric coded character set identifier (CCSID) associated with the file. The numeric value must be between 0 and 65535. You can set this field to blanks or entering a value of 0 to indicate there is no CCSID associated with the file.

Modify owning group (MG)

The Modify z/OS UNIX Owning Group panel allows the user to change the owning group ID for a file or directory. This function is invoked using the **MG** line command or by using the Modify pull-down menu on the z/OS UNIX File Information panel. Placing the **MG** line command next to the filename, **cachex**, (see Figure 17-11 on page 402) displays the panel shown in Figure 17-14 on page 405.

```
ISRUULMG            Modify z/OS UNIX File Owning Group
Command ===> ____________________________________________________________

Pathname . : /u/rogers/cachex
Type . . . : Dir

GID Number   2_________
Group ID . . SYS1____
```

Figure 17-14 Panel to modify the owning Group ID

In Figure 17-14, the owning group for a file or directory is modified by changing one of the following fields on this panel, as follows:

GID Number Enter the GID of the new group. This must be a number, in the range 1 to 2147483647, and must be defined as a z/OS UNIX GID in your security data base.

Group ID Enter the group ID of the new group. The group ID must be defined as a z/OS UNIX group in your security database.

Note: Only a superuser or the owner can change the owning group of a file or directory. If you are the owner, you can change the group to your group or any of your supplementary groups.

Modify owning user (MO)

The Modify z/OS UNIX Owning User panel allows the user to change the owning user ID for a file or directory; see Figure 17-15. The owning user for a file or directory can be modified by either, entering the `MO` (Modify Owner) line command against an entry on the z/OS UNIX Directory List display, or by selecting the Owning User option on the Modify action bar on the z/OS UNIX File Information panel.

The owning user can be modified by changing one of the following fields on this panel, as follows:

UID Number Enter the UID of the new owner. This must be a number, in the range 1 to 2147483647, and must be defined as a z/OS UNIX UID in your security data base.

User ID Enter the user ID of the new owner. The user ID must be defined in your security database and have the authority to use z/OS UNIX resources.

Note: Only a superuser can change the owning user of a file or directory.

```
ISRUULMO            Modify z/OS UNIX File Owning User
Command ===> ____________________________________________________________

Pathname . : /u/rogers/cachex
Type . . . : Dir

UID Number   0_________
User ID  . . STC_____
```

Figure 17-15 Modify z/OS UNIX File Owning User panel

Modify user audit options (UA)

The Modify z/OS UNIX File User Audit Options panel allows the user to define the access attempts that can be audited by the security system. The user can specify auditing to occur for read, write, and execute attempts on the file or directory as shown in Figure 17-16.

The user auditing options for a file or directory can be modified by either entering the **UA** (User Auditing) line command against an entry on the z/OS UNIX Directory List display or by selecting the User Auditing option on the Modify action bar on the z/OS UNIX File Information panel.

These options allow you to define the access attempts that will be audited by the security system. You can specify auditing to occur for read, write, and search or execute attempts on the file or directory.

Figure 17-16 displays fields for specifying the Read, Write and Execute (or search) audit settings. For each of these fields you enter one of the listed numbers corresponding to one of the following results for the access attempt:

None	No audit record is to be written for this type of access.
Failure	Write an audit record if this type of access fails.
Success	Write an audit record if this type of access is successful.
Both	Write an audit record for both failed and successful access attempts.

Note: A superuser or the owner can change the user audit options for a file or directory.

```
ISRUULUA          Modify z/OS UNIX File User Audit Options
Command ===> ____________________________________________________________

Pathname . : /u/rogers/cachex
Type . . . : Dir

Read 2  1. None      Write 2  1. None      Execute 2  1. None
        2. Failure            2. Failure             2. Failure
        3. Success            3. Success             3. Success
        4. Both               4. Both                4. Both
```

Figure 17-16 Modify z/OS UNIX File User Audit Options panel

Publication changes

The new line commands and functions for the z/OS Unix Directory List Utility are in *z/OS ISPF User's Guide Volume II*, SC34-4823.

17.5 Panel source statement input exit

ISPF applications often need to display various information about the panel, based on conditions tested at execution time. This is normally handled by using either a single panel with a dynamic area for the data that can change, or by using multiple panels showing separate information, with the appropriate panel displayed at execution time. Using dynamic areas can be complex, but using multiple panels means there are more application parts to maintain.

To address this problem, in z/OS V1R11 ISPF provides the ability for applications to define a panel input exit that can be used to modify the panel source as it is read by ISPF. The)INEXIT section (see Figure 17-17 on page 407), which must be specified as the first statement in the

panel source member, identifies a program that is called by ISPF for each source record read for the panel. The program is passed the panel source record and can change the record, delete the record, or insert a new record.

Defining the)INEXIT section

The)INEXIT section, which must be specified as the first statement in the panel source member, identifies a program that is called by ISPF for each source record read for the panel. The program is passed the panel source record and can change the record, delete the record, or insert a new record.

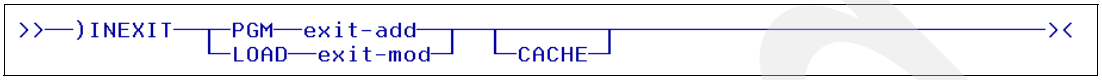


Figure 17-17)INEXIT section

Where:

PGM This indicates the exit routine is already loaded. The exit routine address is passed in exit-add.

exit-add This is the name of a dialog variable containing the address of the exit routine.

LOAD This indicates the exit routine is to be dynamically loaded by ISPF.

exit-mod This is the name of the exit routine that is to be dynamically loaded by ISPF.

CACHE This requests ISPF retain a copy of the panel in virtual storage and use this copy for subsequent displays.

ISPF uses the standard parameter list format to pass parameters. Register 1 points to a list of addresses where each address points to a separate parameter.

The parameters that are passed from ISPF to the panel exit are as follows:

Panel Name This is the name of the panel for which the panel input exit is being invoked.

Panel record buffer address This is the address of the buffer area containing the data for the latest record read from the panel member.

Panel record buffer length This is the length of the buffer area containing the data for the latest record read from the panel member.

Panel record length This is the length of the latest record read from the panel member. Its format is a full word fixed value.

Flags 4 bytes of bit flags passed to the exit by ISPF and defined as:

0 End of file indicator:

0 End of file not reached for panel member.

1 End of file reached for panel member.

1-31 Reserved.

Data area address This is a full word that the exit can use to save the address of a data area obtained by the exit and used to retain or pass information between invocations of the exit.

ISPF uses the return code from the panel input exit to determine how the panel record has been processed; see Figure 17-18 on page 408.

```
0  Process the current panel record (exit may have modified the record data)
2  Exit has inserted a new record. The current record will be passed on the
next call to the exit.
4  Delete the current panel record
8  Stop calling the panel input exit. ISPF continues to process the remaining
records from the panel member.
20) (or other)Severe error in the exit routine. The DISPLAY service terminates
with a severe error condition (return code 20).
```

Figure 17-18 Return Codes from the panel input exit execution

Panel input exit processing

ISPF calls the panel input exit for each record in the panel member after the initial record with the)INEXIT statement. Calls to the exit continue until one of the following occurs:

- The)END statement is processed.
- The exit passes a return code of 8 back to ISPF.
- The exit passes a return code of 20 or an unrecognized return code back to ISPF.
- ISPF encounters a terminating or severe error condition during panel processing.

To modify a panel record, the panel input exit must:

- Make the required changes to the data in the panel record buffer area.
- Set the return code to a value of 0.

To insert a new panel record, the panel input exit must:

- Store the data for the new panel record in the buffer area.
- Set the return code to a value of 2.

When the panel input exit inserts a new panel record, the record passed by ISPF on that call is again passed on the subsequent call to the exit.

To delete a panel record, the panel input exit must:

- Set the return code to a value of 4.

Panel input exits

The panel input exits cannot issue calls to any ISPF services apart from the VCOPY service. The VCOPY service can be used by the exit to get the values for ISPF dialog variables. Examples of using panel input exits are provided in the following members in the ISPF samples data set ISP.SISPSAMP:

ISPPXMNP This is the menu panel that can have options added by input exit ISPPXMNX.

ISPPXMNX This is the input exit that can dynamically add options to menu panel ISPPXMNP.

ISPPXINP This is the menu panel containing special *INCLUDE statements processed by input exit ISPPXINX.

ISPPXINX This is the input exit that processes the *INCLUDE statements in menu panel ISPPXINP. The *INCLUDE statements identify a member in the ISPPLIB DD. The exit reads the member and includes the records as source for the menu panel.

ISPPXDAP This is the data display panel showing the values for static and dynamic system symbols.

ISPPXDAX This is the input exit for panel ISPPXDAP. It causes the panel to display either the static symbols, the dynamic symbols, or both, depending on the value found in ISPF dialog variable DISPREQ.

Publication changes

The new)INEXIT panel source statement and the processing for panel input exits are described in *z/OS ISPF Dialog Developer's and Reference*, SC34-4821.

17.6 Extended address volume data set support

With z/OS V1R11, DFSMS supports the allocation of sequential extended format data sets in the Extended Address Space (EAS) of an Extended Address Volume (EAV). ISPF is enhanced to provide interfaces to this new DFSMS support, as follows:

- Enable sequential extended format data sets to be allocated in the Extended Address Space (EAS) of an EAV
- Display the extended attributes setting for a data set

Users have a panel interface for allocating a data set in the EAS and checking the extended attributes setting of a data set. Option A (Allocate New Data Set) of the Data Set Utility (ISPF Option 3.2) is enhanced with a new **Extended Attributes** field. If OPT is specified in this field, the data set can have extended attributes and reside in EAS, as shown in Figure 17-19.

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
ISRUAASE                    Allocate New Data Set
Command ===> ____________________________________________________________
                                                                More:     +
Data Set Name  . . . : ROGERS.EAVNEW.VOLUME

Management class . . . MCDB22__        (Blank for default management class)
Storage class  . . . . EAVGK___        (Blank for default storage class)
 Volume serial . . . . GKDD65          (Blank for system default volume) **
 Device type . . . . . _______         (Generic unit or device address) **
Data class . . . . . . EAVGK___        (Blank for default data class)
 Space units . . . . . CYLINDER__      (BLKS, TRKS, CYLS, KB, MB, BYTES
                                        or RECORDS)
 Average record unit   _           (M, K, or U)
 Primary quantity  . . 30000_____       (In above units)
 Secondary quantity    0_____          (In above units)
 Directory blocks  . . 0_____          (Zero for sequential data set) *
 Record format . . . . FB____
 Record length . . . . 80____
 Block size  . . . . . 27920_
 Data set name type    _______         (LIBRARY, HFS, PDS, LARGE, BASIC, *
                                        EXTREQ, EXTPREF or blank)
 Extended Attributes   OPT _           (NO, OPT or blank)
 Expiration date . . . _________       (YY/MM/DD, YYYY/MM/DD
Enter "/" to select option              YY.DDD, YYYY.DDD in Julian form
_  Allocate Multiple Volumes            DDDD for retention period in days
                                        or blank)
```

Figure 17-19 ISPF Allocate New Data Set panel

The "Data Set Information" panels available from the Data Set Utility (ISPF option 3.2) and the Data Set List Utility (ISPF option 3.4) line commands **I** and **S** are enhanced to display the "Extended Attributes" setting for a data set as shown in Figure 17-20 on page 410.

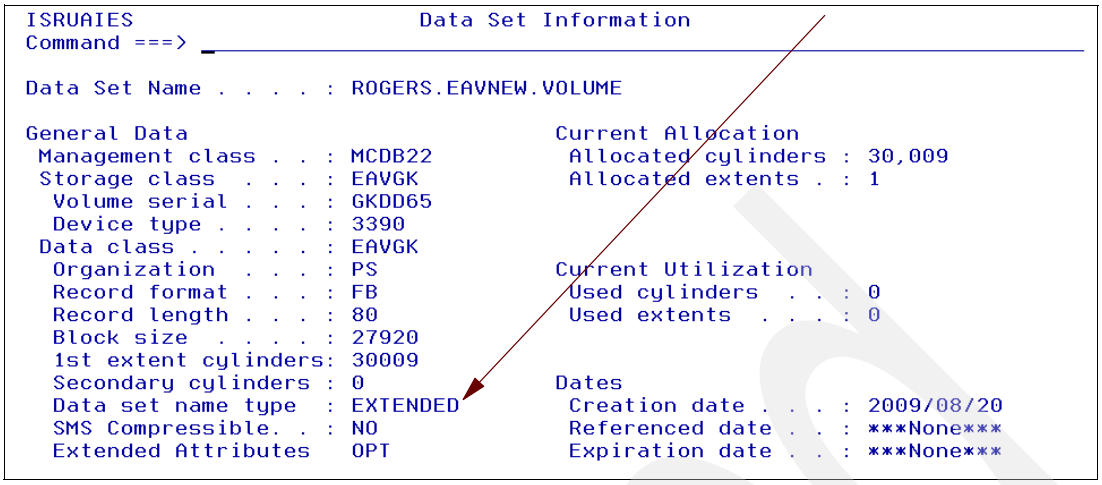


Figure 17-20 ISPF Data Set Information of a EAS data set allocated.

Publication changes

For more information about the new dialog variables that return the “Extended Attributes” setting for services DSINFO and LMDLIST, see *z/OS Interactive System Productivity Facility (ISPF) Services Guide*, SC34-4819.

For more information about the new **Extended Attributes** field on the “Allocate New Data Set” panel, see *z/OS Interactive System Productivity Facility (ISPF)* User’s Guide Volume II, SC34-4823.

17.7 Extended member statistics

Member line count values maintained in the ISPF statistics entry in the PDS/E directory are limited to a maximum of 65535 (half-word) and in fact are truncated to that value when the number of lines is exceeds this value. With PDS/E being increasingly used to store large amounts of data, the limit on line count values becomes a more evident issue.

With z/OS V1R11, the ISPF statistics area in the PDS/E directory is expanded to allow line count values to be maintained in full-word fields. This supports line count values of up to 2,147,483,647.

ISPF configuration table

New keywords are added to the ISPF configuration table to allow sites to decide whether to enable support for extended member statistics.

- STATS_EXT_ENABLED YES | NO

  This keyword allows you to use commands, services, and utilities that cause extended statistics to be created. The default is NO.

- STATS_EXT ON | OFF

  This keyword allows users to control whether the editor maintains ISPF extended statistics for PDS members. The default is OFF.

- FORCE_STATS_EXT YES | NO

  This keyword allows you to force the specified STATS_EXT value. The default is NO.

ISPF edit STATS macro

The STATS macro command sets stats mode, which creates and maintains statistics for a member of a partitioned data set. The STATS assignment statement either sets stats mode, or retrieves the setting of stats mode and places it in a variable.

The ISPF edit STATS primary and macro commands are enhanced to support the new EXT option. When EXT is specified, the member being edited is enabled for extended line number statistics. EXT can only be specified if ISPF configuration keyword STATS_EXT_ENABLED is set to YES.

- STATS ON | OFF | **EXT**

  The option EXT(YES|NO) has been added to the ISPF services that can update a PDS/E member's statistics. These services are:

  – LMMADD (Add a member)

  – LMMREP (Replace a member)

  – LMMSTATS (set and store, or delete ISPF statistics).

  When EXT(YES) is specified, ISPF stores the member line count statistics in extended format. New variables that provide the extended statistics for a member are returned when STATS(YES) is specified for the LMMFIND and LMMLIST services.

New variables returned when STAT(YES) is specified for the LMMFIND and LMMLIST services, as follows:

ZLEXT A value of YES indicates the member has extended statistics. If ZLEXT is YES, the following variables are also returned and contain values from 0 to 2147483647:

- ZLCNORCE Current number of records
- ZLINORCE Beginning number of records
- ZLMNORCE Number of changed records

Members with extended statistics are highlighted in the member list, as shown in Figure 17-21.

```
  Menu  Functions  Confirm  Utilities  Help
 ------------------------------------------------------------------------------
 ISRUDSM EDIT       HAIMO.ISPSTATS.TEST                    Row 00001 of 00003
 Command ===> _                                            Scroll ===> HALF
           Name     Prompt       Size   Created          Changed          ID
 _________ ONE                     1   2009/08/11  2009/08/11 18:10:05  HAIMO
 _________ THREE                   1   2009/08/11  2009/08/11 18:10:05  HAIMO
 _________ TWO
           **End**
```

Figure 17-21 Members with EXT defined are highlighted in the DSLIST (3.4) panel

The ISPF functions that are used to reset (create/update) member statistics now provide an option to generate extended statistics for the selected members, as shown in Figure 17-22 on page 412.

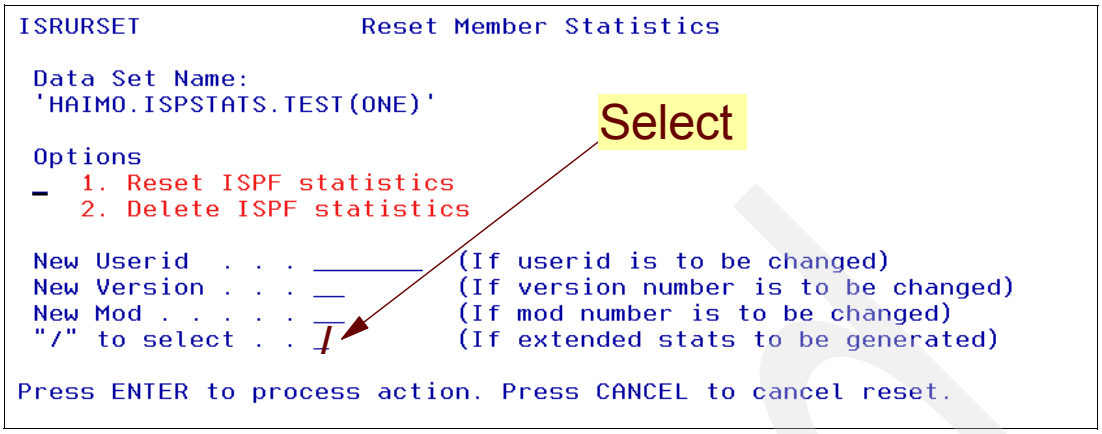


Figure 17-22 Reset member statistics panel

The ISPF functions that are used to reset ISPF statistics are shown in Figure 17-23.

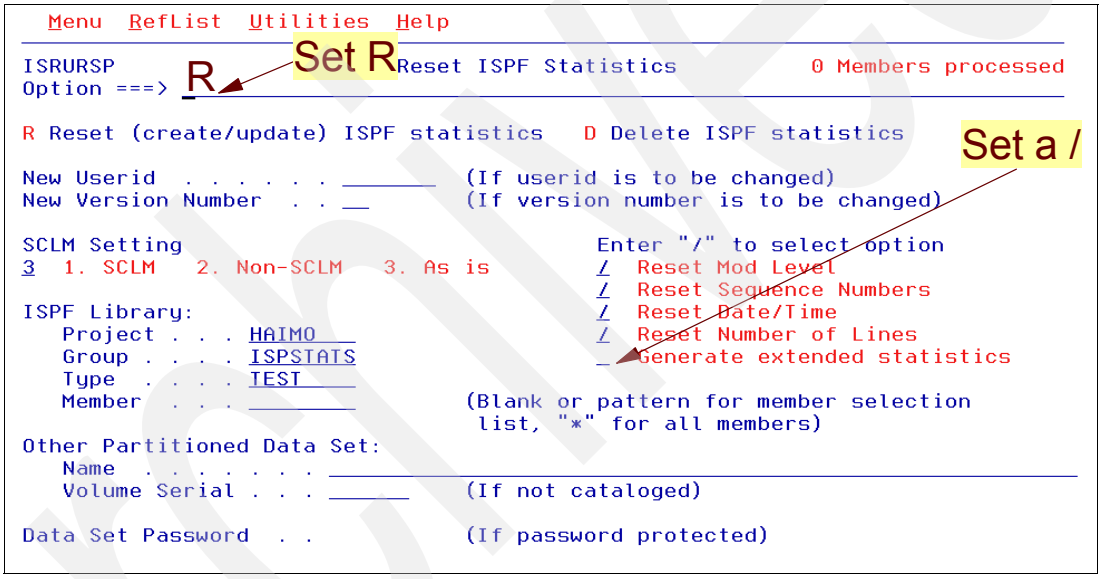


Figure 17-23 Reset ISPF Statistics panel

Publication changes

z/OS V1R11 changes are documented in the following publications:

- *z/OS Interactive System Productivity Facility (ISPF) Planning and Customizing*, SC34-4814

  This book describes the new extended statistics configuration options.

- *z/OS Interactive System Productivity Facility (ISPF) Messages and Codes*, SC34-4815

  This book describes the changes to the layout of the ISPF statistics entry in the PDS/E directory.

- *z/OS Interactive System Productivity Facility (ISPF) Services Guide*, SC34-4819

  This book describes the changes to the ISPF services that update or return extended statistics.

- *z/OS Interactive System Productivity Facility (ISPF)* User's Guide Volume I, SC34-4822

  This book describes the member list changes to highlight members with extended statistics.
- *z/OS Interactive System Productivity Facility (ISPF) User's Guide Volume II*, SC34-4823

  This book describes the changes to the reset member statistics functions to support extended statistics.

17.8 Configure qualifiers for PDF utility output data sets

The default names for PDF utility output data sets consists of a default suffix appended to the user's TSO prefix or user ID. This can result in users who have the same user ID on separate systems in a sysplex suffering enqueue contention or data loss when using these PDF utilities.

With z/OS V1R11, a new configuration option is provided which, when selected, causes the addition of a qualifier with the default suffix for PDF utility output data set names.

The new ISPF site-wide profile customization configuration setting USE_ADDITIONAL_QUAL_FOR_PDF_DATA_SETS does the following:

- It specifies whether an additional qualifier is included in the default data set name for data sets generated by PDF utilities.
- If YES, then the value specified for ISPF_TEMPORARY_DATA_SET_QUALIFIER is used as the additional qualifier.
  - The additional qualifier is included at the start of the default suffix.
- The default is NO.

The additional qualifier is used for the following data sets:

- Listing and update data sets generated by the SuperC (ISPF options 3.12), SuperCE (3.13) and Search-For (3.14) and Search-ForE (3.15) utilities
- The listing data sets produced by the data set list and member list **SRCHFOR** commands
- The trace data sets generated by the I**SPVCALL**, **ISPDPTRC**, and **ISPFTTRC** commands
- The data set created by the ISPF table utility (3.16) **FILE** and **FEXPORT** commands
- The listing data set for the edit **COMPARE** primary command
- The output data set generated by the ISRDDN utility **CLIST**, **SAVE**, and **DUP** primary commands

Publication changes

The following publications describe the changes involved with this new support:

- *z/OS Interactive System Productivity Facility (ISPF) Planning and Customizing*, SC34-4814

  This book describes the new USE_ADDITIONAL_QUAL_FOR_PDF_DATA_SETS configuration keyword.
- *z/OS Interactive System Productivity Facility (ISPF) Edit and Edit Macros*, SC34-4820

  This book describes the possible additional qualifier for the listing data set for the **COMPARE** command.

- *z/OS Interactive System Productivity Facility (ISPF) User's Guide Volume I*, SC34-4822

  This book describes the possible additional qualifier for the output data sets for the ISRDDN utility **CLIS**T, **SAVE**, and **DUP** commands.

- *z/OS Interactive System Productivity Facility (ISPF) User's Guide Volume II*, SC34-4823

  This book describes the possible additional qualifier for the data sets generated by the SuperC, SuperCE, Search-For, Search-ForE utilities and the ISPF table utility **FILE** and **FEXPORT** commands.

17.9 SVC 99 return code information

A number of ISPF functions that use dynamic allocation did not extract the SVC 99 error information and make it available for display on the help panel ISRSVCER for the dynamic allocation error message ISRD024.

z/OS V1R11 changes ensures all ISPF functions using dynamic allocation provide the SVC 99 error information for panel ISRSVCER.

Users are not required to perform actions to utilize this enhancement; it is implemented by an internal change to ISPF.

18

RRS enhancements

Many computer resources are so critical to a company's work that the integrity of these resources must be guaranteed. If changes to the data in the resources are corrupted by a hardware or software failure, human error, or a catastrophe, the computer must be able to restore the data. These critical resources are called protected resources or, sometimes, recoverable resources.

Resource recovery is the protection of the resources. Resource recovery 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 that can be accessed through separate resource managers and reside on separate 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 new functions for the RRS subsystem introduced in z/OS V1R11, as follows:

- Timestamps on RRS panels
  - Provide a new panel to set RRS global options
  - Use a format to identify date and time values on RRS panels
  - Label all log browser output timestamps as LOCAL
- RRS exploits a new GRS latch identifier

18.1 Time stamps on RRS panels

A requirement for an RRS dialog option now allows the user to specify whether output TIMESTAMP values are to be displayed in a local format or a GMT format. This request arises from a requirement to use the UR detailed display panel (ATRFPURD). The panel includes the UR Create Time, which is currently displayed in GMT only.

RRS now adds a dialog panel to its main panel, ATRFPCMN, to allow users to specify global options to be applied to all RRS panels. If subsequent individual panels support the same option, then the specified value on these panels overrides the value previously set for the global option. ISPF-related code will be enhanced to process the global options. Time stamps will default to GMT. Figure 18-1 shows the modified RRS ISPF entry panel. The new option 0 was added.

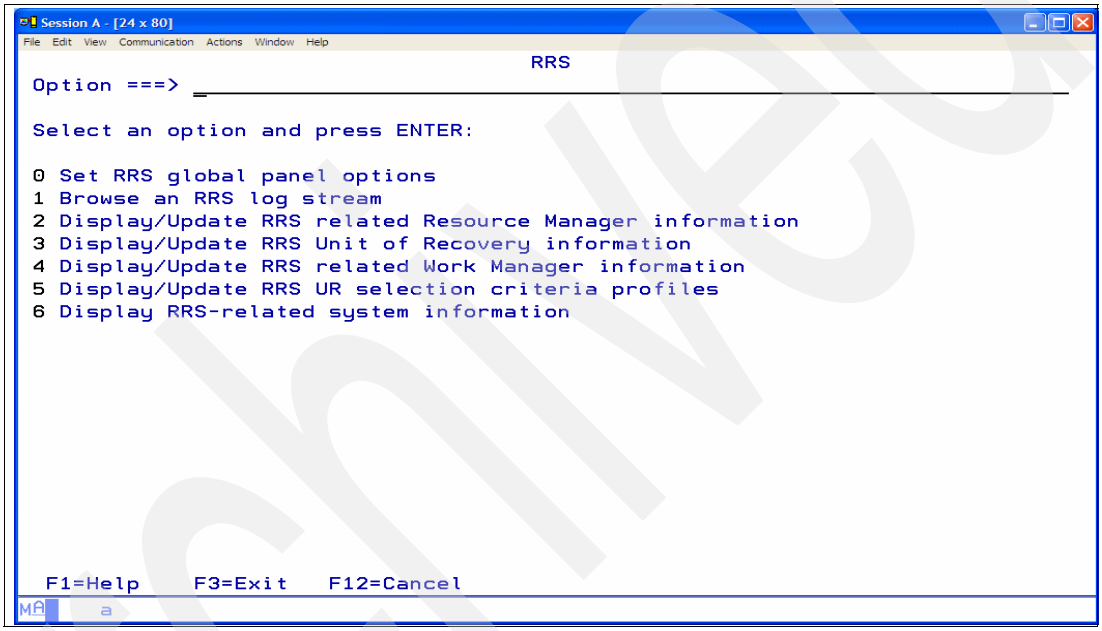


Figure 18-1 RRS entry panel

Label all log browser output time stamps as LOCAL. This will be seen from the Browse an RRS log stream panel option, ATRQSRV utility, and from the ATRBATCH sample JCL.

Note: This support is only available from UR panels; it will not be added to the ATRQUERY callable service or the ATRQSRV utility.

After entering option zero (0), you switch to the new RRS options dialog shown in Figure 18-2 on page 417. In this dialog you can choose how to display the time in the RRS browser output. You either choose GMT or LOCAL. In our sample we chose LOCAL; the default is GMT.

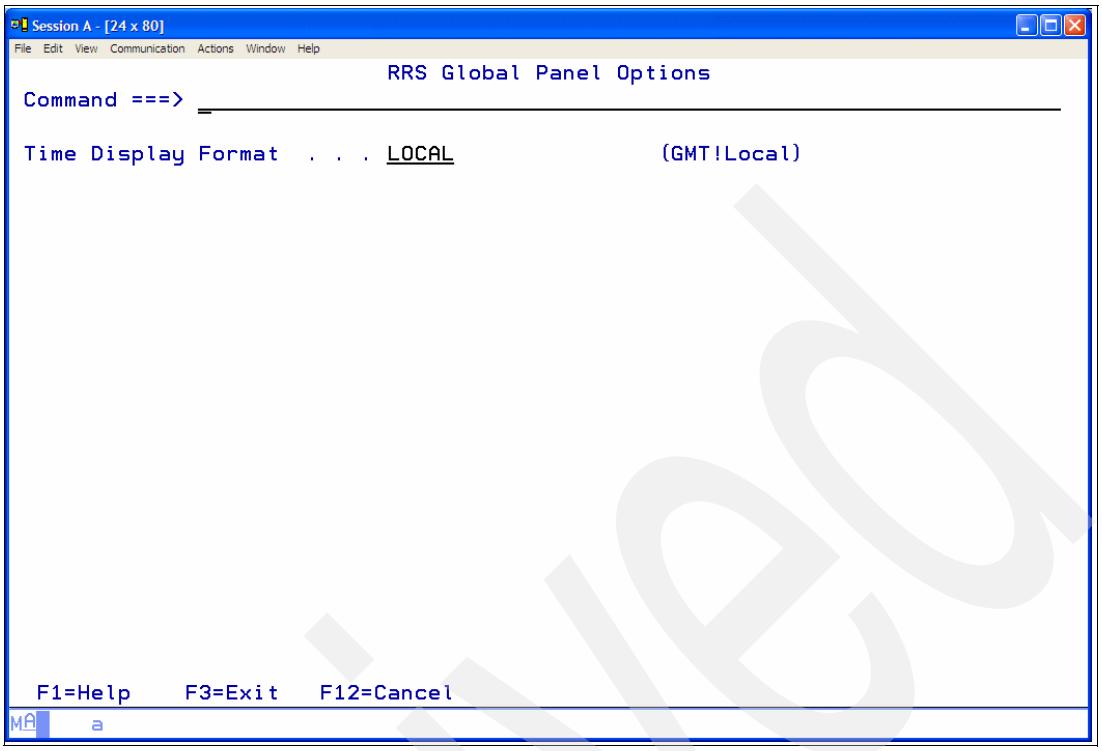


Figure 18-2 RRS time format setup panel

This option affected all RRS browser output. Figure 18-3 shows you an example of the new time option. As you see, the time zone is added after the regular time stamp.

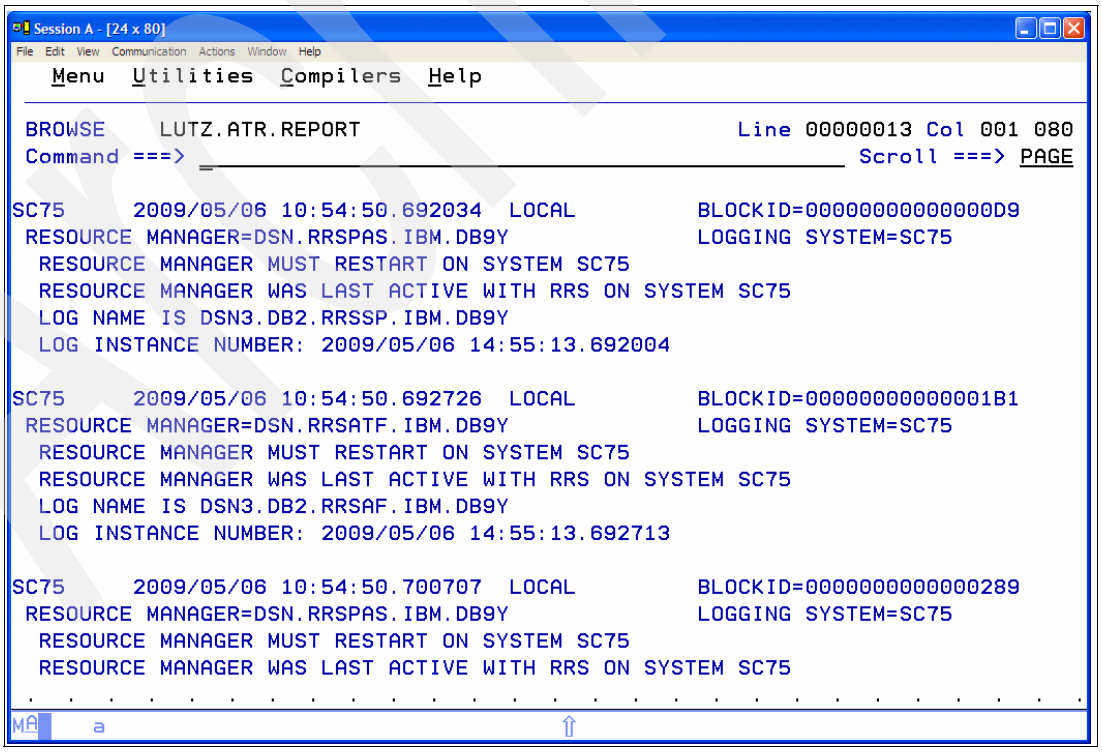


Figure 18-3 Sample RRS report panel

18.2 Latch IDs

RRS uses numerous latches to establish an orderly flow during transaction processing. If a unit of work hangs up while holding a critical latch, other work units can also wait for the latch, thereby causing a hung transaction. The Display GRS Contention (**D GRS,C**) command will establish that there is contention, but it does not provide sufficient information for an operator or systems programmer to quickly narrow the problem down to a particular latch or latches causing the problem and the affected transaction.

Figure 18-4 shows an example latch contention of latch number 20. Starting with Release 11 of z/OS, RRS will exploit the new GRS Latch Identifier support to help an installation identify which RRS resource is involved in the bottleneck.

```
SY1  ISG343I 12.28.05 GRS STATUS 844
NO ENQ RESOURCE CONTENTION EXISTS
LATCH SET NAME:  SYS.ATRRMCPO.00000001
CREATOR JOBNAME: RRS       CREATOR ASID: 0031
  LATCH NUMBER:  20
    REQUESTOR ASID EXC/SHR   OWN/WAIT WORKUNIT TCB ELAPSED TIME
    RRS       0031 EXCLUSIVE OWN      026C5100  N  00:02:17.493
    RRS       0031 EXCLUSIVE WAIT     024E7480  N  00:02:09.160
```

Figure 18-4 RRS LATCH contention

To understand how to analyze the problem, you can use the following command:

```
D GRS,ANALYZE,LATCH,DEPENDENCY,DETAIL
```

Figure 18-5 shows the output of the **D GRS** system command. It shows that there a hang situation on latch RM:SY1.D1.RRJLAZ10.RM1.

```
SY1  ISG374I 12.33.50 GRS ANALYSIS 848
DEPENDENCY ANALYSIS: ENTIRE SYSTEM
----- LONG WAITER #1
        JOBNAME: RRS       (ASID=0031, WEB=024E7480)
        REQUEST: EXCLUSIVE                LT:7E94F07800000088
WAITING  00:07:11 FOR RESOURCE (CREATOR ASID=0031)
SYS.ATRRMCPO.00000001                    LST:7EAACF0000000141
20:RM:SY1.D1.RRJLAZ10.RM1
        JOBNAME: RRS       (ASID=0031, WEB=026C5100)
        REQUEST: EXCLUSIVE                LT:7F52001000000147
ANALYSIS ENDED: THIS UNIT OF WORK IS NOT WAITING
```

Figure 18-5 Sample output of D GRS command

Latch contention identifiers

For general detection purposes, IBM introduced several new RRS Latch contention identifiers to recognize the latch that is causing the contention.

- Context Data - CD
- Resource Managers - RM
- System Hash Table- SHT
- System Bucket Hash Table - SHT Bucket
- Sysplex Hash Element- SHE
- Units of Recovery - UR

- UR Main State Log
- UR Compression State Log

Table 18-1 lists all the new RRS Latch identifiers.

Table 18-1 RRS Latch Identifiers

Latch set	Latch identifier	Description	RRS panel option
CD Context Data	CD URID:urid	urid - Unit of Recovery Identifier.	3 - Display/Update RRS Unit of Recovery information and search on the urid.
RM Resource Manager	RM:rmname	rmname - Resource Manager Name.	2 - Display/Update RRS-related Resource Manager information and search on the rmname.
SHT System Hash Table	SHT System:sysname	sysname – name of the system that owns the SHT. “Unknown” can be displayed for the system name. The Unknown SHT table keeps track of transactions that are moving between systems, sometimes due to system restart problems.	If SHT contention persists, contact your IBM Support Center.
SHT System Bucket Hash Table	SHT Bucket number System:sysname	number - bucket number. sysname - name of the system that owns the SHT. “Unknown” can be displayed for the system name. The Unknown SHT table keeps track of transactions that are moving between systems, sometimes due to system restart problems.	If SHT contention persists, contact your IBM Support Center.
SHE Sysplex Hash Element	SHE SURID:surid	surid - Sysplex Unit of Recovery Identifier.	3 - Display/Update RRS Unit of Recovery information and search on the urid.
UR Units of Recovery	UR URID:urid	urid - Unit of Recovery Identifier.	3 - Display/Update RRS Unit of Recovery information and search on the urid.
UR Main State Log	MainQ Log:log	log - UR Log Stream Name.	If State Log contention persists, contact your IBM Support Center.

Latch set	Latch identifier	Description	RRS panel option
UR Compression State Log	CompQ Log:log	log – UR Log Stream Name.	If State Log contention persists, contact your IBM Support Center.

Note: A contention is not always a problem. However, if contention persists longer than a reasonable amount of time, then take action.

19

SMF dump program

To dump an SMF log stream to archive the data to a permanent medium such as tape, or so that existing user-written analysis routines can run against the dump data sets, you can dump SMF log stream data by issuing the **SWITCH SMF** command. This command first dumps the SMF data in the buffers out to the log streams, and then passes control to the IEFU29L SMF log stream dump exit. The operator can use the SMF log stream dump program, IFASMFDL, to dump the specified log stream data to dump data sets.

This chapter covers the new ARCIVE/DUMP functionality of the IFASMFDL SMF program, and discusses the new RELATIVEDATE option in the IFASMFDL dump program:

- IFASMFDL program enhancements
- RELATIVEDATE option

19.1 IFASMFDL enhancements

The base SMF logstream shipped the IFASMFDL program with only a DUMP option. Feedback indicated that some kind of CLEAR-like function was required for SMF logstreams. In addition, clients required the ability to delineate on a time boundary, something that could have been done with dataset recording by executing an **I SMF** command to switch datasets at a given time. Now, with the IFASMFDL dump program, SMF data can be removed from the logstream with the ARCHIVE and DELETE options:

ARCHIVE Dump SMF records and delete them

DELETE Delete without dumping

The ARCHIVE option will first dump the data from the logstream to a dataset, and then perform a delete of the data from the logstream. The DELETE option will only perform a delete of the data from the logstream without dumping it.

Unlike the DUMP option, ARCHIVE and DELETE operate on logger block boundaries. This means that there may be a fuzziness in the selected output records. This is why the MAXDORM parmlib option is now supported. By using the MAXDORM function, you can limit the amount of fuzziness in the selected data. Figure 19-1 shows an example of the use of the new ARCHIVE option IFASMFDL program.

```
//LUTZSMF JOB (MISY,TXX,T870270),KUEHNER,MSGCLASS=X,
//           MSGLEVEL=(1,1),
//           NOTIFY=&SYSUID,CLASS=D
/*JOBPARM S=SC70
//ARCHIVE  EXEC PGM=IFASMFDL
//SYSPRINT DD   SYSOUT=*
//DUMP01   DD   DSN=LUTZ.SMF.DUMP,DISP=(,CATLG),
//   SPACE=(CYL,(10,10)),UNIT=3390
//SYSMDUMP DD   DSN=LUTZ.SMF.SYSMDMP,DISP=(,CATLG),
//   SPACE=(CYL,(50,5)),UNIT=3390
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    LSNAME(SMF_TYPEALL,OPTIONS(ARCHIVE))
    OUTDD(DUMP01,TYPE(0:255))
    RELATIVEDATE(BYDAY,7,3)
/*
//
```

Figure 19-1 Sample SMF ARCHIVE job

The MAXDORM support for logstreams was crucial to supporting the new ARCHIVE and DELETE functions. MAXDORM has the same effect as it did with dataset recording, except that it will work with each logstream now. To limit the amount of fuzziness, use a smaller MAXDORM value. This causes the buffers to be purged to the logstream more often, and each time this is done a new logger block is created for these records.

19.2 RELATIVEDATE option

The new RELATIVEDATE option will allow the user to easily select a range of records based on the current day. Because there is no existing function to delineate a range of time in the

logstream, and to dump successive days out of a logstream requires manual intervention, this new parameter was developed. RELATIVEDATE gives the user the power to specify the unit of time (BYDAY, BYWEEK, BYMONTH), as well as the number of units to move backwards, and the number of units to gather.

This option is used to specify a date range based on the current day, week or month.

RELATIVEDATE(u, x, y)

u BYDAY, BYWEEK or BYMONTH

x Number of units to move back

y Number of units to gather

For example, RELATIVEDATE(BYWEEK,7,3) as shown in Figure 19-2 will move back seven weeks and gather three weeks. An associated parameter is the WEEKSTART parameter. Because some locales start the week on Sunday and some on Monday, this keyword allows you to alter that behavior.

```
//LUTZSMF JOB (MISY,TXX,T870270),KUEHNER,MSGCLASS=X,
//            MSGLEVEL=(1,1),
//            NOTIFY=&SYSUID,CLASS=D
/*JOBPARM S=SC70
//ARCHIVE  EXEC PGM=IFASMFDL
//SYSPRINT DD   SYSOUT=*
//DUMP01   DD   DSN=LUTZ.SMF.DUMP,DISP=(,CATLG),
//   SPACE=(CYL,(10,10)),UNIT=3390
//SYSMDUMP DD   DSN=LUTZ.SMF.SYSMDMP,DISP=(,CATLG),
//   SPACE=(CYL,(50,5)),UNIT=3390
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    LSNAME(SMF_TYPEALL,OPTIONS(DUMP))
    OUTDD(DUMP01,TYPE(0:255))
    RELATIVEDATE(BYDAY,7,3)
/*
//
```

Figure 19-2 RELATIVEDATE example job

20

High performance FICON for z

This chapter describes a new I/O facility called High Performance FICON for System z (zHPF) that is provided on z10 systems.

FICON uses the Fibre Channel Single Byte Command Sets-3 (FC-SB-3) standard as defined by the International Committee of Information Technology Standards (INCITS), and published as ANSI standards. Late in 2008 IBM introduced System z High Performance FICON (zHPF) on the System z10. zHPF uses the architecture as described in the draft proposal for Fibre Channel Single Byte Command Sets - 4 (FC-SB-4). FC-SB-4 is intended to be a complete replacement of the FC-SB-3 standard.

20.1 Overview of System z High Performance FICON

High Performance FICON for System z (zHPF) is one of the latest IBM enhancements to the z/Architecture and FICON interface architecture to dramatically improve the performance of OLTP workloads. These architectural enhancements are designed to substantially improve I/O performance by reducing the number of Channel Command Words (CCWs) and Information Units (sequences), which removes overhead on the storage subsystem and the FICON channel microprocessor.The rationalization of layers of handshakes is established without compromising the reliability, availability and serviceability (RAS) implemented with FICON technology today.

These enhancements have been introduced to optimize online transaction processing (OLTP) workloads using numerous access methods such as DB2, VSAM, PDSE, HFS, zFS. When exploited by the FICON channel, the z/OS operating system and the storage subsystem, zHPF helps reduce overhead and improve performance. Additionally, the changes to the architectures offer end-to-end system enhancements to improve reliability, availability, and serviceability.

20.2 Using the FICON architecture for I/O operations

Before a System z FICON channel can send any I/O over the link, the link must be initialized. There are specific steps used in this initialization process.

After the initialization process is complete, I/O operations can be started on the FICON channel. A FICON channel operates in what is known as *command mode*, exploiting FC-SB-3 protocols. A System z10 FICON channel can operate in command mode and *transport mode* (also known as zHPF), exploiting the FC-SB-4 protocols.

When both the System z10 FICON channel and the control unit indicate support for zHPF, then the channel will perform the I/O operation in transport mode. If the control unit does not support zHPF, the System z10 FICON channel will operate in command mode. zHPF has been enabled in z/OS V1R11 to perform transport mode operations. Both modes (command and transport) are discussed later in this chapter.

20.2.1 Mainframe I/O technology milestones: ESCON to zHPF

The evolution of mainframe I/O architecture includes many significant milestones. When these are reviewed from the outset, the significance of zHPF can be put into context. In particular when we compare a start I/O using ESCON to that of FICON and now zHPF, the significance of these iterative changes becomes apparent.

The differences between ESCON architecture and FICON and zHPF architecture are listed in Table 20-1.

Table 20-1 Channel architecture comparisons

ESCON	Native FICON, CTC, zHPF
Circuit switching	Packet switching
Read or write Half-duplex data transfers	Simultaneous read and write Full-duplex data transfers
Connection-oriented	Connection-less

ESCON	Native FICON, CTC, zHPF
Pre-established dedicated path	Packets individually routed
Connection is locked when data sent	Connection released when data sent
Synchronous transfers	Asynchronous transfers
CCW architecture	CCW architecture for FICON TCW architecture for zHPF

20.2.2 FICON I/O request

Before the I/O operation starts and FC frames are transmitted over the fiber cable, the channel program must be created and passed to the channel subsystem (CSS). The Input/Output Supervisor (IOS) is a z/OS component between the CSS and the application program, which issues the start subchannel command. The application program uses an access method that calls the I/O driver to build the channel program. The channel program can use transport commands when operating in transport mode, or channel commands when operating in command mode.

- Transport mode is used by the System z High Performance FICON (zHPF) architecture. A transport mode I/O operation uses a transport control word (TCW) to send commands and data over the FC link and does not use CCWs. The channel and control unit must support zHPF.
- Command mode is used by the existing FICON architecture where an I/O operation consists of CCWs and data.

When the SSCH is passed to the CSS (see Figure 20-1 on page 428) and the FICON channel is selected, the FC architecture and FICON protocols are used to build the frames required to send over the FC link.

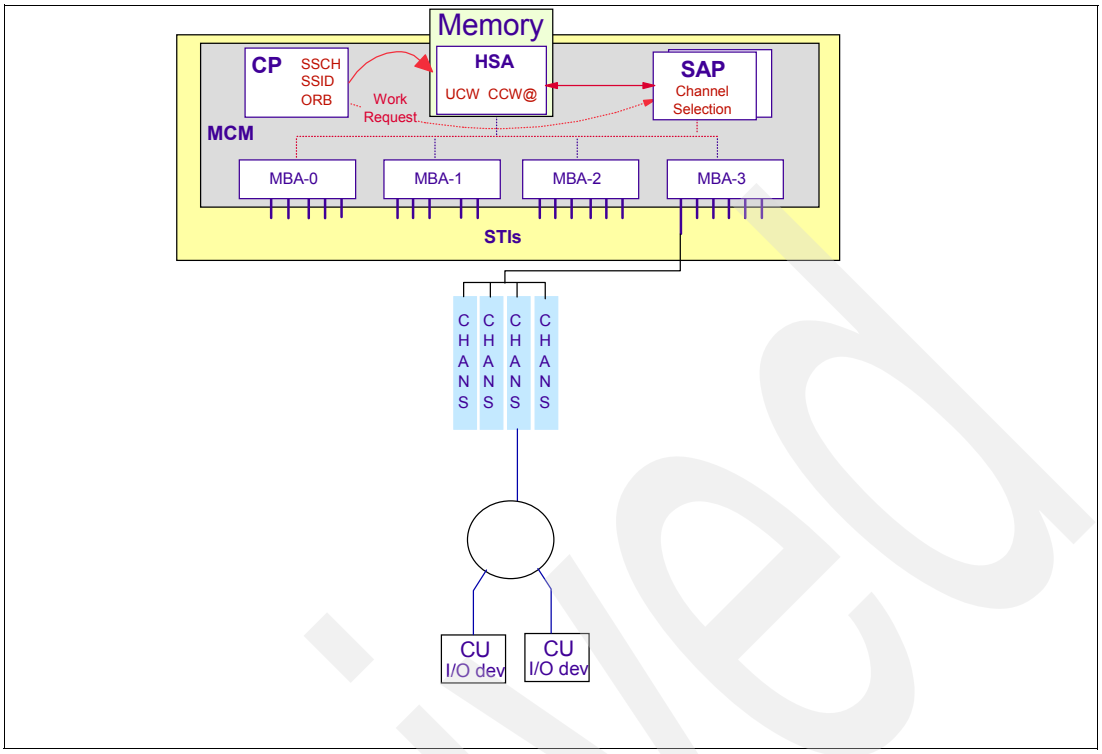


Figure 20-1 I/O operation overview

FICON I/O requests

A FICON I/O request flows as follows:

1. An application or system component invokes an I/O request. The application or access method provides CCWs or TCW/TCCB, as well as additional parameters in the Operation Request Block (ORB).
2. The request is queued on the unit control block (UCB) and the input output supervisor (IOS) will service the request from the UCB on a priority basis.
3. IOS issues a start subchannel (SSCH) instruction with the subsystem identification word (SSID) representing the device and ORB as operands.
4. The ORB contains start-specific control information, and it indicates whether the channel is operating in transport mode (zHPF support) or command mode. It also indicates the starting address of the channel program's (CPA) channel command.
5. The CSS selects the most appropriate channel and passes the I/O request to it.
6. The channel fetches the following from storage:
   - If in command mode, the channel command words and associated data (for write operations)
   - If in transport mode, the Transport Control Word, Transport Command Control Block and associated data (for Write Operations)
7. The channel assembles the required parameters and fields of the FC-2 and FC-SB-3 or FC-SB-4 for the I/O request and passes them to the Fibre channel adapter (which is part of the FICON channel).

The System z FICON architecture defines the protocol for CCW and data pipelining, and how FICON channels operate (known as command mode). Command mode uses the CCWs and data as shown in Figure 20-2 to perform an I/O operation.

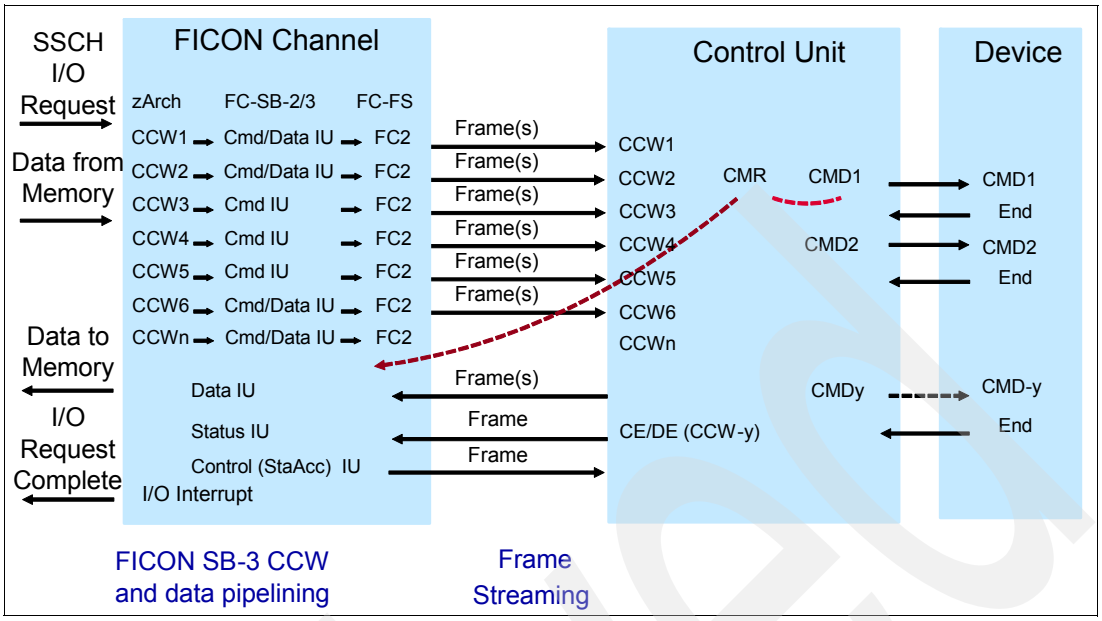


Figure 20-2 FICON command mode operation

ESCON channels

A fundamental difference with ESCON is the CCW chaining capability of the FICON architecture. Although the ESCON channel program operation requires a Channel End/Device End (CE/DE) after execution of each CCW, FICON supports CCW chaining without requiring a CE/DE at the completion of each CCW operation.

The ESCON channel transfers the CCW to the control unit and waits for a CE/DE presented by the control unit after execution of the CCW by the device (CCW interlock). After receiving a CE/DE for the previous CCW, the next CCW is transferred to the control unit for execution.

With a FICON channel, CCWs are transferred to the control unit without waiting for the first command response (CMR) from the control unit or for a CE/DE after each CCW execution. The device presents a logical end to the control unit after each CCW execution. After the last CCW of the CCW chain has been executed by the CU/device, the control unit presents a CE/DE to the channel.

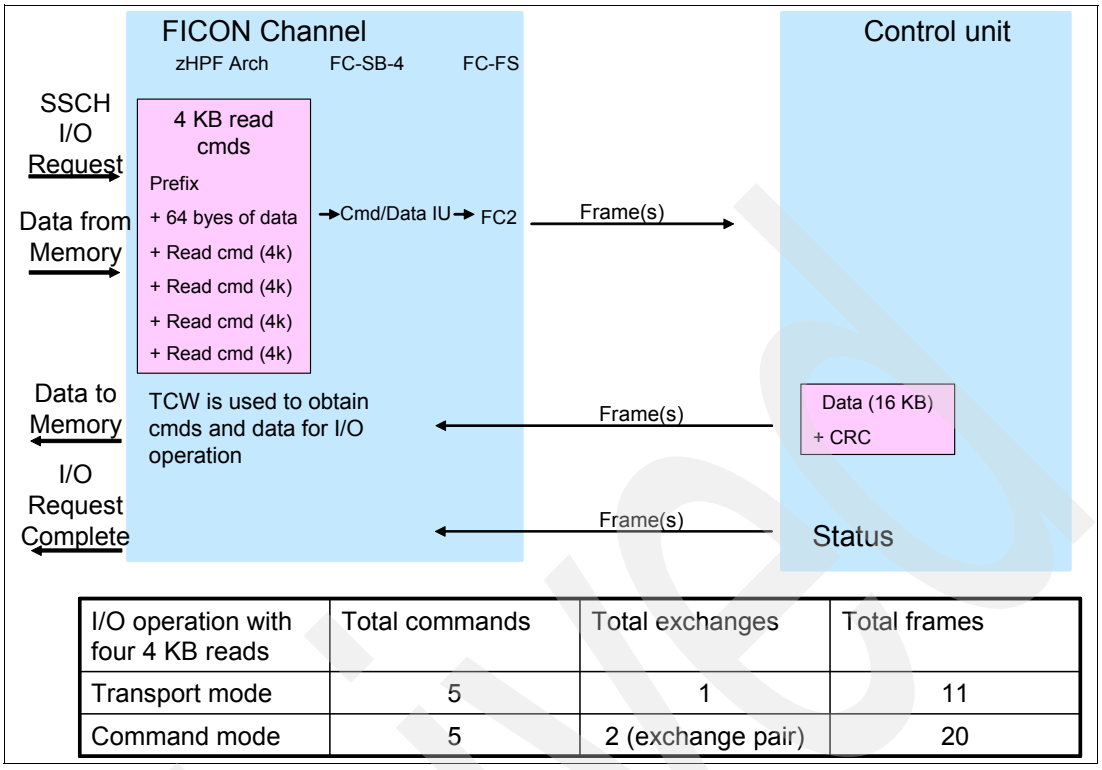
In addition, FICON channels can multiplex data transfer for several devices at the same time. This also allows workloads with low to moderate control unit cache hit ratios to achieve higher levels of activity rates per channel.

20.2.3 Transport mode

A new form of channel program is introduced with zHPF, which is, as previously mentioned, called transport mode.

How transport mode handles an I/O operation differs significantly from the CCW operation for command mode. While in command mode, each single CCW is sent to the control unit for execution. In contrast, in transport mode all commands are sent in a single frame to the control unit.

The FICON channel uses transport mode if the control unit supports zHPF. When operating in transport mode, an I/O operation will use a transport control word (TCW) to send commands and data; see Figure 20-3.



I/O operation with four 4 KB reads	Total commands	Total exchanges	Total frames
Transport mode	5	1	11
Command mode	5	2 (exchange pair)	20

Figure 20-3 FICON transport mode operation

The FICON Express8, FICON Express4, and FICON Express2 features support transport mode. A parameter in the Operation Request Block (ORB) is used to determine how the FICON channel will operate (command mode or transport mode). The mode used for an I/O operation also depends on the settings in the z/OS operating system.

The application or access method provides the channel program commands and parameters in the ORB. Bit 13 in word 1 of the ORB specifies how to handle the channel program in either command mode or transport mode.

20.2.4 zHPF support determination

zHPF support is determined at link initialization.The initialization steps used to determine if transport mode will be used.

During link initialization both the channel and the control unit indicate whether they support zHPF.

20.2.5 Open exchange

An open exchange is part of FICON (and FC) terminology. Many I/O operations can be in progress over FICON channels at any one time. For example, a disk I/O operation might temporarily disconnect from the channel while performing a seek operation or while waiting for a disk rotation. During this disconnect time, other I/O operations can be managed.

- Command mode open exchanges
  - In command mode, the number of open exchanges is limited by the FICON Express feature. FICON Express8, FICON Express4, FICON Express2 allow up to 64 open

exchanges. One open exchange (this is an exchange pair) in command mode is the same as one I/O operation in progress.

- Transport mode open exchanges

  In transport mode, one exchange is sent from the channel to the control unit. Then the same exchange ID is sent back from the control unit to the channel to complete the I/O operation. The maximum number of simultaneous exchanges that the channel can have open with the CU is 750 exchanges. The CU sets the maximum number of exchanges in the status area of the transport mode response IU (the default number is 64, and this can be increased or decreased).

20.3 Programming view

For the past 45 years, specific programming has been provided on the successive IBM System/360 embodiments to handle I/O operations up to the zSeries hardware machines.

This specific programming has been known to produce channel programs, the same way classical programs are written for general purpose processors.

Each channel command word (CCW) contains the basic instructions of a channel program. The CCW specifies the channel command to execute (read, write, search, and so on), as well as flags and the location of the data in main storage.

Data location in main storage is not subject to address translation (such as with DAT for general purpose processors). However, it has been enhanced in various ways to indirectly reference the data, through mechanisms known as Indirect Data Address Words (IDAWs) or Modified Indirect Data Address Words (MIDAWs).

20.3.1 I/O command words

An I/O command word specifies a command and contains information associated with the command. When the zHPF facility is installed, there are two elementary forms of I/O command words:

- The channel command word (CCW)
- The device command word (DCW)

In command mode, a CCW is 8 bytes in length and specifies the command to be executed. For commands that initiate certain operations, the CCW also designates the storage area associated with the operation, the count of data bytes, the action to be taken when the command completes, and other options. All I/O devices recognize CCWs.

In transport mode of zHPF, a DCW is 8 bytes in length and specifies the command to be executed, the count of data bytes, and other options. I/O devices that support zHPF recognize DCWs.

Figure 20-4 on page 432 illustrates a CKD channel program.

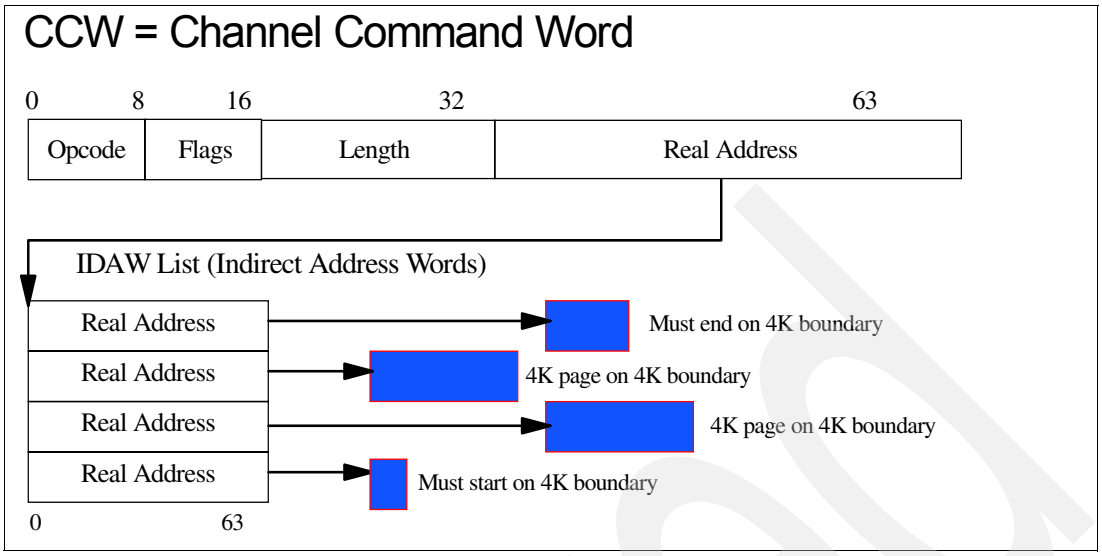


Figure 20-4 CKD channel program

20.3.2 CCW channel program

A channel program that is comprised of one or more CCWs is called a CCW channel program (CCP). Such a channel program contains one or more CCWs that are logically linked and arranged for sequential execution by the channel subsystem, according to the opcodes or the flags, as depicted in Figure 20-5.

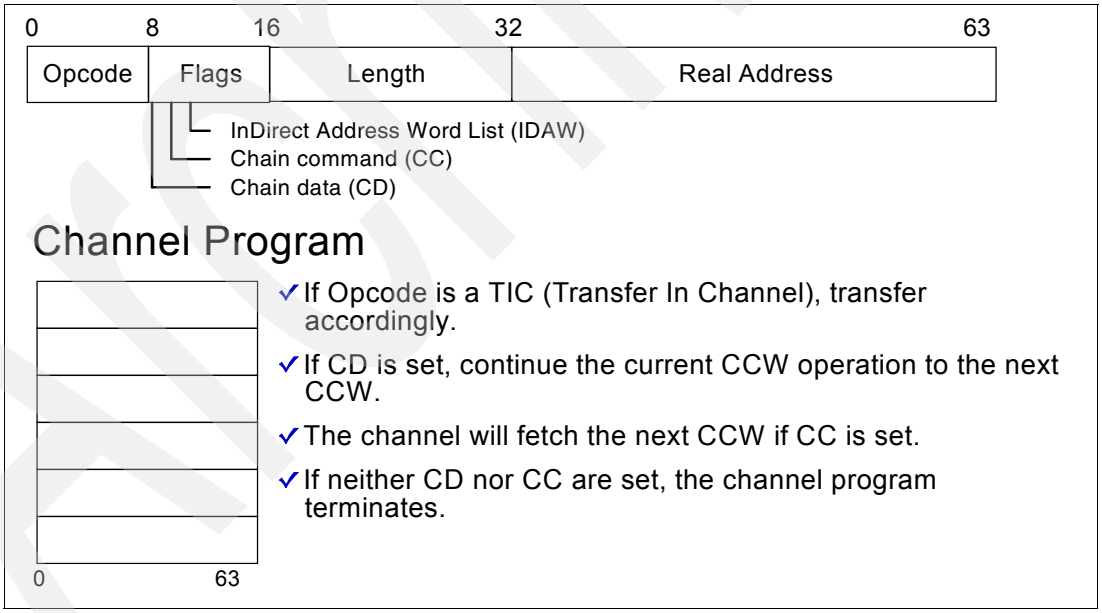


Figure 20-5 CCW chaining

Figure 20-6 on page 433 shows a conceptual example of a simple CCW channel program in program storage. This channel program contains three CCWs and specifies the transfer of data to or from contiguous areas of storage. The first CCW designates a control command. The remaining CCWs designate the transfer of data.

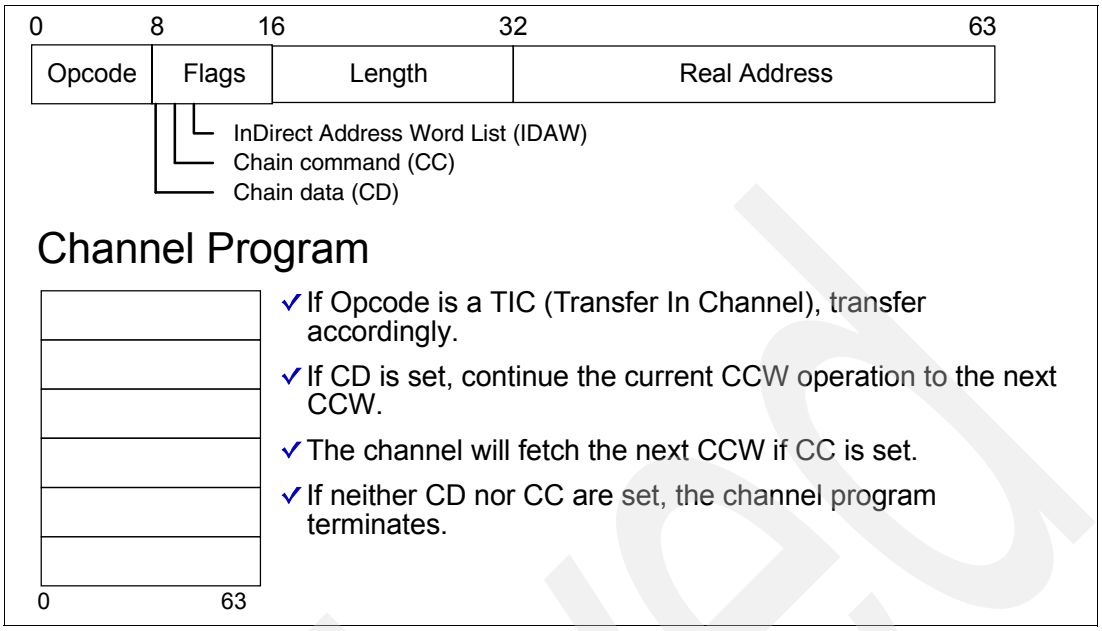


Figure 20-6 Write Count Key Data using data chaining

20.3.3 Modified Indirect Data Address Word

On System z, the Modified Indirect Data Address Word (MIDAW) provides alternatives to using CCW data chaining in channel programs. The MIDAW facility has been added to z/Architecture and can coexist with the current CCW IDAW facility.

MIDAW allows scattering of data in memory for non-contiguous real pages. (This is sometimes known as scatter-read or scatter-write.) Although the CCW IDAW function requires all but the first and last IDAW in a list to deal with complete 2 KB or 4 KB units of data, the MIDAW facility allows page boundary crossing on either 2 KB or 4 KB boundaries. This allows access to data buffers anywhere in a 64-bit buffer space. See Figure 20-7 on page 434 for an example of MIDAW usage.

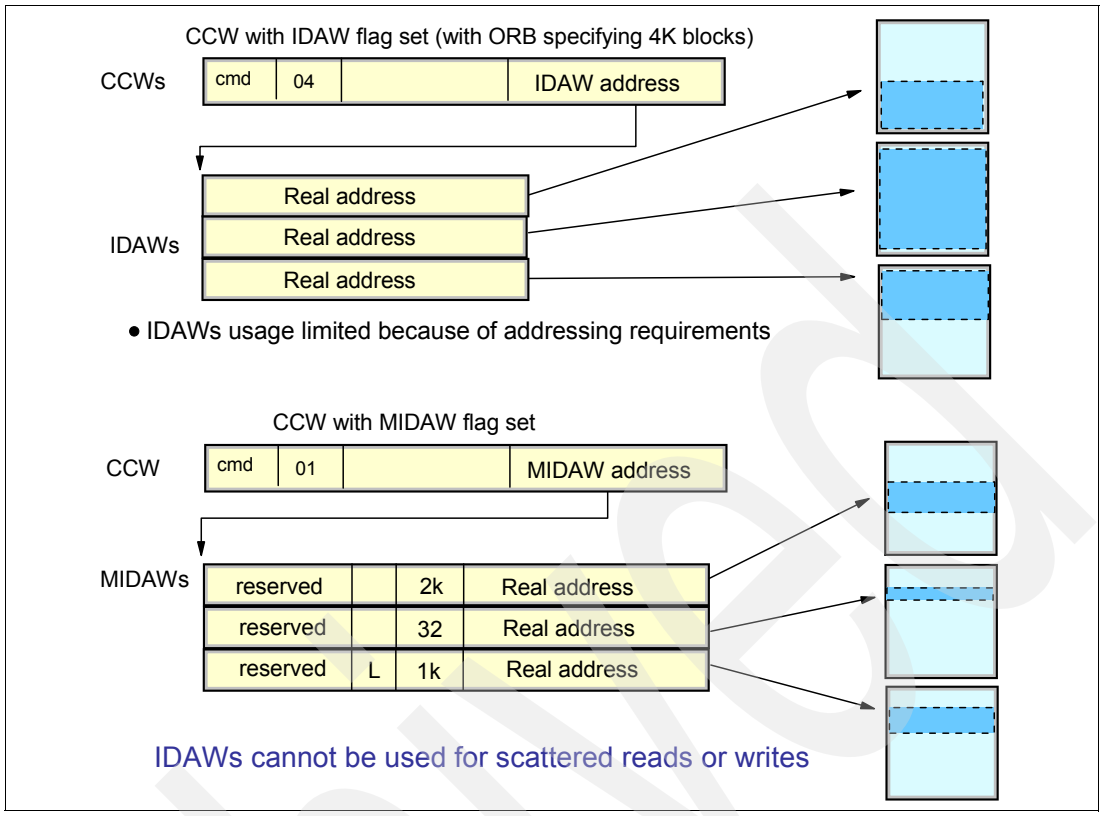


Figure 20-7 Command mode MIDAWs

IDAWs were introduced when virtual storage and DAT were incorporated within the processor. However, IDAWs cannot be used for scattered reads or writes, which occur when data records of a database are updated by transactions. For this reason, MIDAWs have been introduced as illustrated in Figure 20-8.

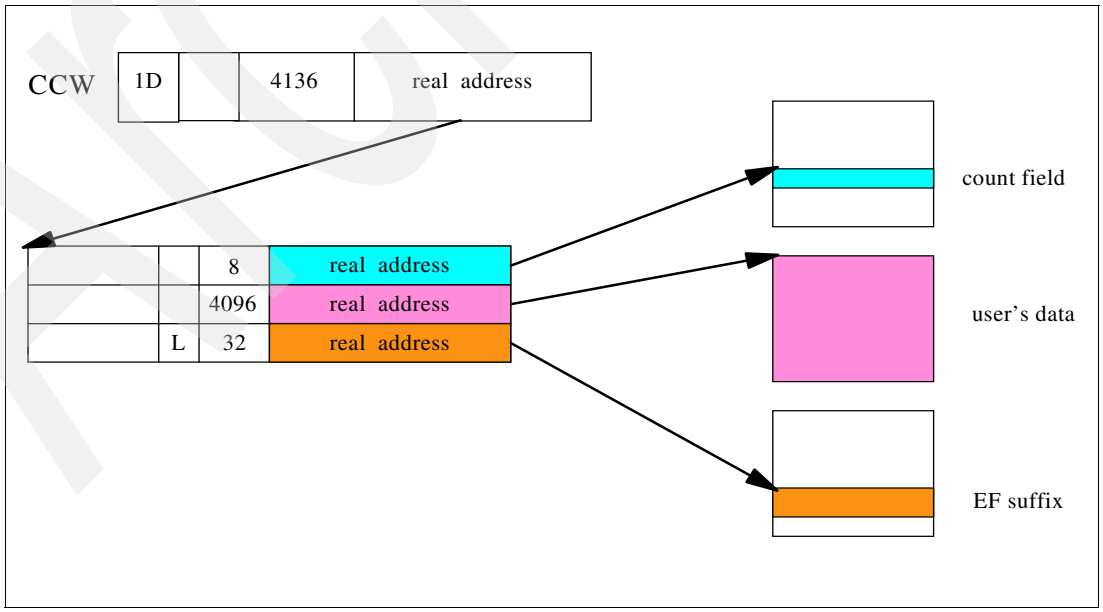


Figure 20-8 Write Count Key Data using MIDAWs

20.4 zHPF

The zHPF facility can be available on hardware models implementing z/Architecture. The facility includes enhancements and performance extensions that provide the program with a means to transport lists of commands to an I/O device for processing without decoding each command.

Use of the zHPF facility can be restricted to devices accessible by certain channel path types.

The zHPF facility includes the following:

- Changes to the operation-request block (ORB), interruption-response block (IRB), and subchannel-information block (SCHIB).
- A form of channel program consisting of a transport control word (TCW). The TCW designates a transport command control block (TCCB) and transport status block (TSB). The TCCB includes a transport-command area (TCA) which contains a list of up to 30 I/O commands that are in the form of device command words (DCWs).

  The TSB contains completion status and other information related to the TCW channel program that is complementary to the completion information contained in the IRB. A TCW can specify the transfer of either input data or output data.
- The ability to directly or indirectly designate the storage area for an I/O operation, the TCCB, or both. When a storage area is designated directly, the TCW specifies the location of a single, contiguous block of storage. When a storage area is designated indirectly, the TCW designates the location of a list of one or more transport indirect data address words (TIDAWs). TIDAW lists, and the storage area designated by each TIDAW in a list, are restricted from crossing 4 K-byte boundaries.
- An interrogate operation, which can be initiated by CANCEL SUBCHANNEL to determine the state of an I/O operation.

20.5 Transport command word

As depicted in Figure 20-9 on page 436, the transport command word (TCW) designates a transport command control block (TCCB) that contains a list of commands to be transported to, and executed by, an I/O device. The TCW also designates the storage areas for the commands in the TCCB, as well as a transport status block (TSB) to contain the status of the I/O operation.

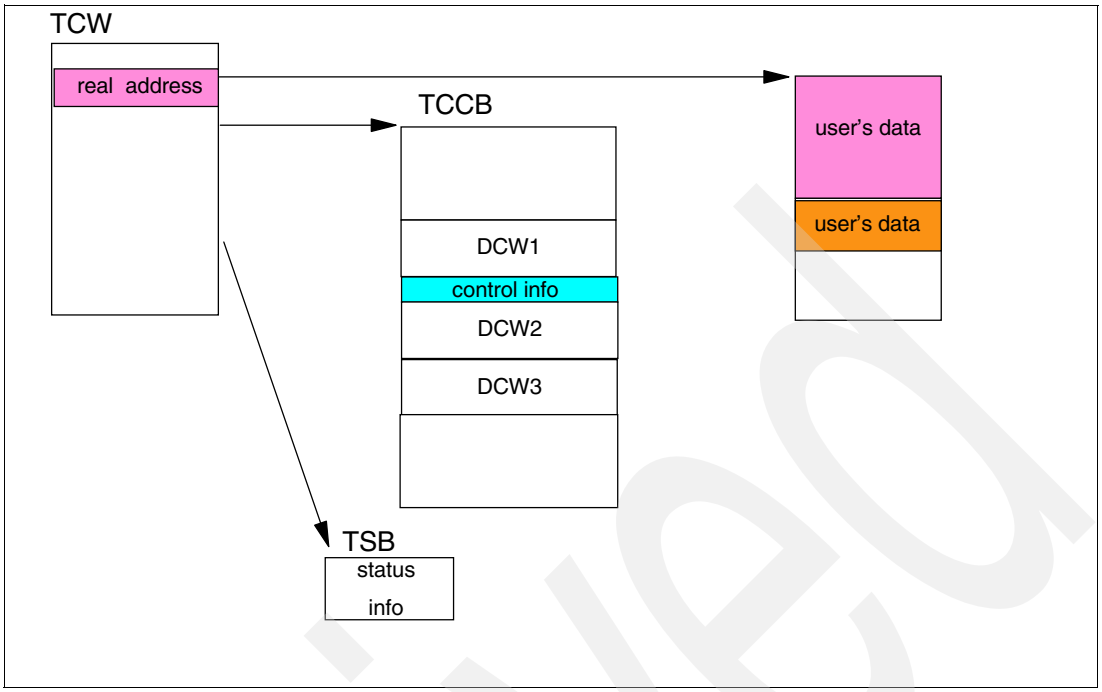


Figure 20-9 TCW channel program

20.5.1 TCW channel program

A channel program that is comprised of a single TCW is called a TCW channel program. A TCW designates a transport command control block (TCCB) that contains from 1 to 30 DCWs. The DCWs within the TCCB are logically linked and arranged for sequential execution. For DCWs that specify control information, the TCCB also contains the control information for those commands.

The TCW also designates the storage area (or areas) for the DCWs that specify the transfer of data from or to the device and the location of a transport status block (TSB) for completion status. The TCCB and the storage areas for the transfer of data may be specified as either contiguous or noncontiguous storage.

Figure 20-9 shows a conceptual example of a TCW channel program in program storage. This channel program is similar to the CCW channel program in Figure 20-6 on page 433. This channel program is comprised of a TCW that designates a TCCB containing three DCWs, all of which specify the transfer of data either to or from contiguous areas of storage.

The first DCW designates a control command and the remaining DCWs designate the transfer of data.

The TCW also designates a TSB for completion status.

When a storage area is designated indirectly, as in Figure 20-10 on page 437, the TCW designates the location of a list of one or more TIDAWs. As mentioned, TIDAW lists, and the storage area designated by each TIDAW in a list, are restricted from crossing 4 K-byte boundaries.

20.5.2 Transport indirect data address word

When the System z channel is operating in transport mode, then transport indirect data address words (TIDAWs) are used. TIDAWs and MIDAWs are similar in concept and provide the capability of scattered reads and writes.

TIDAWs are used when certain flag bits are set in the transport control word. Figure 20-10 is an example of TIDAW usage.

TIDAWs (and MIDAWs) are used with z/OS extended format data sets that use internal structures (usually not visible to the application program) that require scatter-read (or scatter-write) operation.

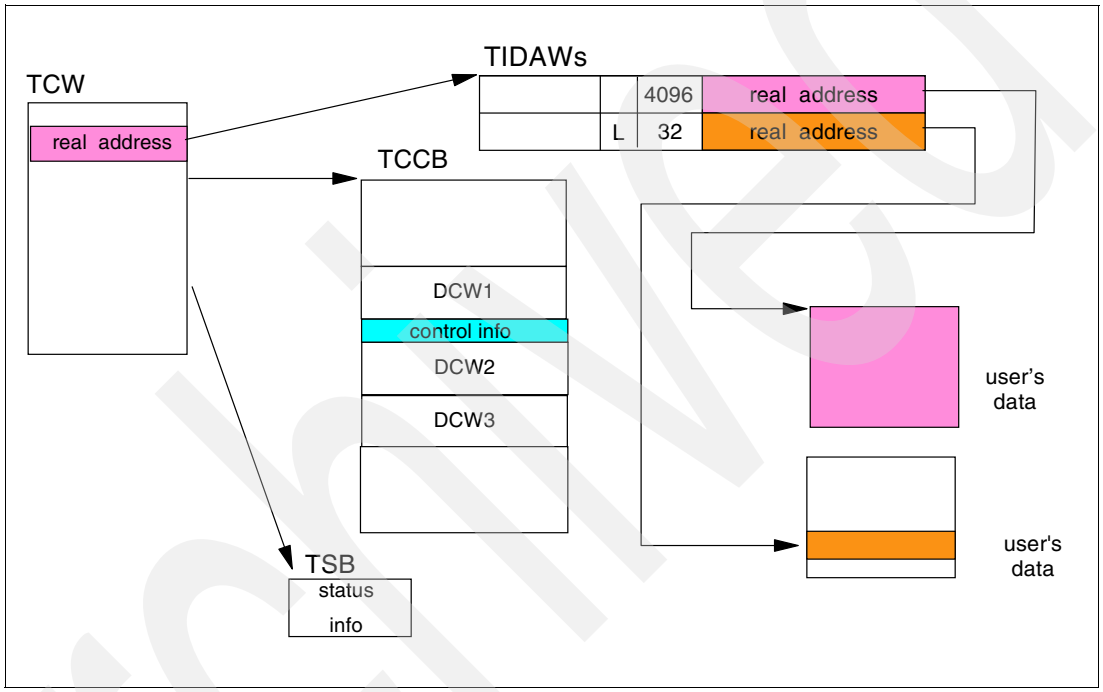


Figure 20-10 TCW with indirect data addressing

20.6 Using zHPF

The zHPF parameter is added in the IECIOSxx parmlib member to enable or disable the High Performance FICON for System z (zHPF) facility on a system. If no ZHPF statement is specified, the zHPF facility is disabled on the system by default.

20.6.1 Displaying zHPF facility status

Use the **DISPLAY IOS,ZHPF** command to display the current status (enabled or disabled) of the High Performance FICON for System z (zHPF) facility.

```
D IOS,ZHPF[,L={a|name|name-a}]
```

Figure 20-11 Displaying the zHPF facility

The status of a zHPF devicecan be seen when device status is displayed, as shown in Figure 20-12.

```
D M=DEV(410)
IEE174I 11.00.11 DISPLAY M 258
DEVICE 0410 STATUS=ONLINE
CHP A0 A1 A2 A3
DEST LINK ADDRESS A0 A1 A2 A3
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 0400 0400 0400 0400
MAXIMUM MANAGED CHPID(S) ALLOWED: 0
DESTINATION CU LOGICAL ADDRESS = 04
SCP CU ND = 002107.000.IBM.TC.02069A00FF04.00FF
SCP TOKEN NED = 002107.000.IBM.TC.02069A00FF04.0400
SCP DEVICE NED = 002107.000.IBM.TC.02069A00FF04.0410
FUNCTIONS ENABLED = MIDAW, ZHPF
```

Figure 20-12 zHPF device status display

Similarly, zHPF status is shown when the channel status is displayed; see Figure 20-13.

```
D M=CHP(A0)
IEE174I 19.38.27 DISPLAY M 343
CHPID A0: TYPE=1A, DESC=FICON POINT TO POINT, ONLINE
DEVICE STATUS FOR CHANNEL PATH A0
0 1 2 3 4 5 6 7 8 9 A B C D E F
0041 + + + + + + + + AL AL AL AL AL AL AL AL
0071 + + + + + + + + UL UL UL UL . . . .
0072 + + + + . UL . . UL . . UL . . UL .
0073 + + UL . UL . UL . UL . UL . UL . UL .
1071 . . . . . . . . UL UL UL UL UL UL UL UL
SWITCH DEVICE NUMBER = B000
DEFINED ENTRY SWITCH - LOGICAL SWITCH ID = 20
ATTACHED ND = 006064.001.MCD.01.000000010C17
PHYSICAL CHANNEL ID = 01D3
FACILITIES SUPPORTED = ZHPF
```

Figure 20-13 zHPF channel status display

20.6.2 SMF and RMF support

SMF TYPE 73 and 79 records have been changed in support of zHPF. There are six new fields added to the RMF Channel activity report to distinguish between FICON and zHPF traffic.

For example, the data in Figure 20-14 on page 439 reflects detail from the RMF Channel Path Activity report. As shown, a single channel (CHPID 13) was performing 18092 zHPF I/Os per second and on average, there were 8.8 zHPF I/Os simultaneously active.

```
CHANNEL PATH UTILIZATION(%) READ(MB/SEC) WRITE(MB/SEC) FICON OPERATIONS zHPF OPERATIONS
ID TYPE G SHR PART TOTAL BUS PART TOTAL PART TOTAL RATE ACTIVE DEFER RATE ACTIVE DEFER
10 FC_S 5 Y 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.0 0.0 0.0 0.0 0.0 0.0
11 FC_S 5 Y 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.0 0.0 0.0 0.0 0.0 0.0
12 FC_S 5 Y 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.0 0.0 0.0 0.0 0.0 0.0
13 FC_S 5 Y 63.97 63.97 12.98 74.11 74.11 0.00 0.00 0.4 1.0 0.0 18092 8.8 0.0
```

Figure 20-14 RMF channel path activity with zHPF

The RATE column represents the number of FICON or zHPF I/Os per second at the total physical channel level (not by LPAR).

The ACTIVE column corresponds to “open exchanges” (the number of I/Os that are simultaneously active within a channel).

The DEFER column represents the number of deferred FICON or zHPF operations per second that were not initiated by the channel due to the lack of available resources.

20.6.3 Missing interrupt handler

zHPF provides additional capabilities such as being able to interrogate the CU before an actual missing interrupt occurs. Because zHPF uses TCWs, which do not time the channel operation in the same way as command mode, a mechanism must be used to prevent unnecessarily invoking the Missing Interrupt Handler (MIH) recovery actions. Transport mode provides an in-band method for z/OS to query the state of the I/O at the control unit without invoking error recovery or retry.

In channel operations there is an MIH time value set for various device classes. The MIH is constantly polling and is used to determine if there are any I/O operations that are not completing. For each of these I/O requests that are not completing, a recovery action will be performed. The recovery action can be one or more of the following:

- Issue a message.
- Generate a logrec record for diagnostic purposes.
- Terminate the I/O request.
- Re-queue the I/O request.

Normally when a channel does not get a CMR response from a started I/O operation within the MIH value, it invokes MIH recovery procedures, which can then terminate the I/O in progress.

The transport mode operation called `interrogate` is used to query the state of the CU. The interrogate process does not affect the state of the primary operation. It is sent before the MIH value is exceeded. The CU provides an interrogate response IU that contains extended status that describes the state of the primary operation. z/OS can then decide whether recovery is needed, or whether resetting the MIH value for this I/O operation is needed.

20.7 Compatibility and coexistence

The zHPF function is currently available in z/OS V1R11 as well as a rollback APAR OA18766, for z/OS V1R8, V1R9, and V1R10 as well as for z/OS V1R7 with the IBM Lifecycle Extension for z/OS V1R7 (5637-A01) with PTFs.

On the control unit side, high performance FICON is supported as a chargeable feature in the IBM DS8000 by machine types 2107, 239x, 2244, 2421, 2422, 2423, and 2424.

zHPF is implemented in the LIC code, driver level 76, of the IBM z10 EC and BC servers.

All online CHPIDs to the disk Logical Control Unit must support zHPF. The inclusion of any non-compliant features (for instance, FICON Express feature codes 2319 and 2320) will result in the entire path group implementing FICON operation, not zHPF. The features supported by zHPF are FICON Express2 and FICON Express4.

zHPF-capable FICON Express4 and FICON Express2 channels and devices support both FICON and zHPF traffic simultaneously. FICON continues to use CCWs and zHPF uses TCWs, so there is compatibility and coexistence.

The MIDAW facility and zHPF are completely independent of each other and there are no interdependencies. However, IBM has incorporated many lessons learned from the MIDAW experience into the design of zHPF.

21

TSO/E LOGONHERE support

This chapter provides detailed information about the LOGONHERE functionality provided in z/OS V1R11. This function for various system programmers is also known as part of the z/VM operating system.

This chapter covers the following enhancements:

- LOGON RECONNECT support
- IKJTSOxx option

21.1 TSO/E reconnect support using LOGONHERE

Starting with z/OS V1R11, TSO/E introduces support to reconnect a 3270 terminal TSO/E session following a connection problem. One of the reasons RECONNECT fails in previous systems is because the system cannot detect, at least in a timely manner, that the user is in a disconnected state. This can occur if the terminal emulator crashes, or if someone tries to logon from another computer without disconnecting first, or their IP address is renewed.

By default in z/OS V1R11, LOGONHERE support for the LOGON command is ON. TSO/E users can reconnect to their session even if no disconnection has been detected. LOGONHERE support makes it easier for users to reconnect from a new terminal or after renewing their IP address.

In a sysplex, users can see the panel shown in Figure 21-1.

Figure 21-1 Logon after connection failure

Operator message

Similarly, an error message is issued to the MVS operator console:

```
IKJ606I TSOLOGON REJECTED. USERID LUTZ IN USE
```

21.1.1 IKJTSOxx parmlib member

In addition to the PASSPHRASE and VERIFYAPPL parameter introduced in z/OS V1R10, IBM provides in z/OS V1R11 a new parameter, LOGONHERE. As mentioned, the default setting when using LOGONHERE is ON which means the logon reconnect support is enabled by default.

Use the IKJTSOxx parmlib member and the LOGON keyword as follows:

LOGON This specifies the system settings for the TSO/E LOGON command:

LOGONHERE(ON|OFF) This specifies whether the RECONNECT option on the TSO/E LOGON panel will be honored even when the system does not detect a disconnected state and the user appears to be logged on. This allows users to reconnect their session from a new terminal without canceling their previous session first, similar to how the LOGONHERE option works under z/VM.

PASSPHRASE(ON|OFF) This specifies whether the TSO/E logon panel allows users to enter up to 100 characters in the password field. If eight characters or less are entered, the input is treated as a password. If more than eight characters are entered, the input is treated as a passphrase. The default specification is OFF.

Note: The toleration support for the LOGON statement in IKJTSOxx parmlib member is added in z/OS V1R9 through APAR OA20525. Toleration for the LOGONHERE keyword is included in the base of z/OS V1R10.

This parmlib setting controls whether or not TSO/E will call the VTAM reconnect service unconditionally after a valid password is entered. Previously, TSO/E checked internal control blocks for indications that the user was in a disconnected state. However, those control blocks were not always accurate. VTAM also had to be updated to make this work seamlessly, even in a sysplex environment. However, there are no external changes to VTAM.

Logon reconnect

Figure 21-2 shows how to use the new functionality of reconnecting session.

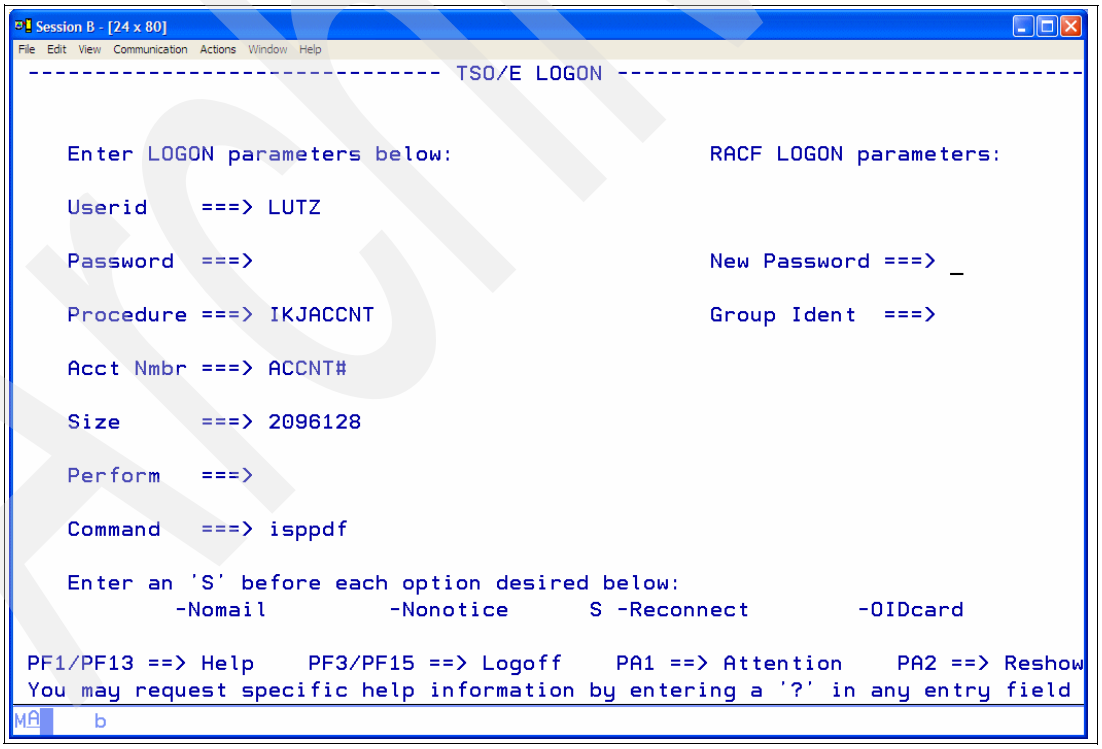


Figure 21-2 TSO logon with RECONNECT option

With z/OS V1R11 you see the successful reconnect, as shown in Figure 21-3. This functions even if the session is not in a disconnect state.

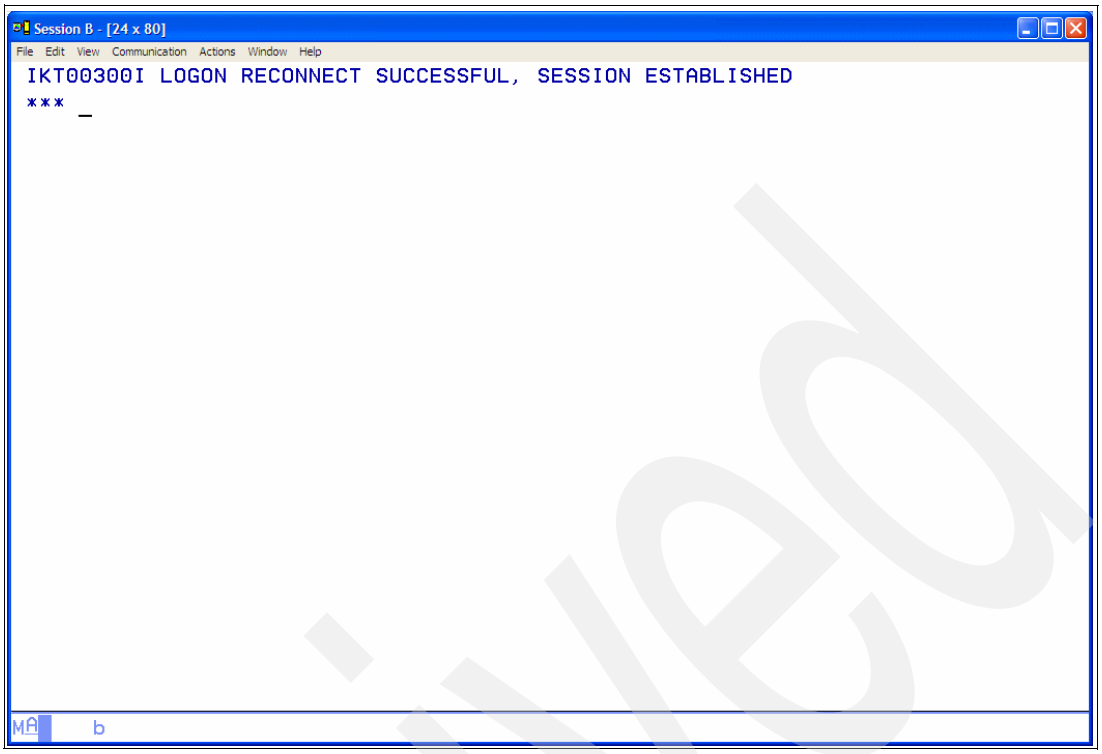


Figure 21-3 Successful reconnection

Operator command for parmlib settings

To determine how the system is set up, you can either use z/OS system command or the TSO **PARMLIB** command. You can use a system command to get the current settings of your system. Figure 21-4 shows the output on a z/OS V1R11 system.

```
D IKJTSO,LOGON
 IKJ738I TSO/E PARMLIB SETTINGS : 101
    SYS1.SYSPROG.PARMLIB(IKJTSO00) on volume SBOX01
    Activated by **IPL** on 2009-10-04 at 15:25:54 from system SC70
    Applies to :    SC70
            CURRENT PARMLIB SETTINGS FOR LOGON:
            PASSPHRASE(OFF)
            VERIFYAPPL(OFF)
            LOGONHERE(ON)
```

Figure 21-4 Output of D IKJTSO,LOGON command

Note: If you route the command from a V1R11 system to a lower level system in the sysplex, the LOGONHERE setting is always OFF.

TSO/E command to display LOGONHERE settings

The second way to determine what setting in effect in the system by using the following TSO command from the ISPF Option 6 command line:

```
PARMLIB LIST(LOGON)
```

The response is shown in Figure 21-5.

```
TSO/E PARMLIB SETTINGS :

  SYS1.SYSPROG.PARMLIB(IKJTSO00) on volume SBOX01
  Activated by ROGERS on 2009-10-09 at 03:52:54 from system SC65
  Applies to :    SC65



          CURRENT PARMLIB SETTINGS FOR LOGON:

          PASSPHRASE(OFF)
          VERIFYAPPL(OFF)
          LOGONHERE(ON)
```

Figure 21-5 Displaying LOGONHERE settings from ISPF Option 6

21.1.2 Changing the settings for logon

In the IEASYSxx parmlib member, the two-character identifier (xx) is appended to IKJTSO to identify the IKJTSOxx parmlib member. If you do not specify the IKJTSO=xx parameter, the system processes the IKJTSO00 parmlib member.

If defined in the parmlib, IKJTSO00 is used automatically during IPL. Another IKJTSOxx member can be selected during IPL by specifying IKJTSO=xx for the IPL parameters. To change the LOGONHERE parameter on the LOGON statement in the SYS1.PARMLIB member IKJTSOxx from the default LOGONHERE(ON) to LOGONHERE(OFF), perform one of the following actions:

- To select another IKJTSOxx member after IPL, you can issue the following TSO/E command:

  ```
  PARMLIB  UPDATE(xx)
  ```

- Or you can use a system command, where xx is the alphanumeric value to be appended to IKJTSO:

  ```
  SET IKJTSO=xx
  ```

Note: You do this by placing the change in a separate xx member, such as the IKJTSOaa member.

You can use the MVS **DISPLAY IKJTSO,LOGON** command or the TSO/E **PARMLIB LIST(LOGON)** command to check the current settings of the LOGONHERE parameter.

Attention: To change the LOGONHERE settings in your system, use a copy of the active IKJTSOxx parmlib member.

22

Language Environment

This chapter describes the enhancements for Language Environment with z/OS z/OS V1R11. The following topics are discussed:

- Updated functions in the new release
- CELQPIPI support
- CICS AFP support
- Decimal floating point support part 3
- Exploitation of large pages
- File I/O tracing
- Heap pool improvements
- C/C++ runtime enhancements
- Changes to assembler macros
- Swap and make context
- C/C++ compiler dependencies

22.1 z/OS V1R11 Language Environment enhancements

The Language Environment is enhanced to support a single unique interface to get the current release information. Applications now can query current releases of Language Environment during program execution to be able to identify what functionality is available.

Note: All output release information is based upon using the z/OS V1R11 level of software. Running older levels of Language Environment on a higher level of z/OS changes the results of several of the examples. Using z/OS V1R11 Language Environment enhancements on an older level of z/OS is not supported.

The following Language Environment calls are modified:

- __librel()
- __osname()
- uname()
- CEEPGID service

These modifications are explained in more detail in the following sections.

__librel()

The __librel() function returns a hexadecimal value of the form pvrrmmmm identifying the level of Language Environment, where:

p Product identifier (4 = z/OS)
v Version
rr Release
mmmm Modification level

Figure 22-1 shows the appropriate source when using the modified __librel() function.

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   printf("The release of the library is:    0x%X\n",__librel());
}
```

Figure 22-1 Example __librel() call

Figure 22-2 shows the output of the __librel() function. In this case, it shows running on z/OS V1R11 and modification level 0000.

```
The release of the library is: 0x410B0000
```

Figure 22-2 Output from __librel() call

__osname()

The __osname() function returns a structure full of information about the operating system. The system name, release, and version match up with the level of operating system on which the application is running. These values do not necessarily reflect the same release as that of Language Environment.

```
#define _POSIX_SOURCE
#include <sys/utsname.h>
#include <stdio.h>
main() {
   struct utsname uts;
   if (__osname(&uts) < 0)
      perror("__osname() error");
else {
      printf("Sysname:  %s\n", uts.sysname);
      printf("Nodename: %s\n", uts.nodename);
      printf("Release:  %s\n", uts.release);
      printf("Version:  %s\n", uts.version);
      printf("Machine:  %s\n", uts.machine);
   }
}
```

Figure 22-3 Sample for osname() function

After compiling the source file and executing your load module, you see a display similar to Figure 22-4.

```
Sysname:  z/OS
Nodename: SC75
Release:  11.00
Version:  01
Machine:  2097
```

Figure 22-4 Output of the osname() function

uname()

The uname() function returns a structure full of information about the operating system, similar to the __osname() function. The difference is that the system name remains constant, with release and version being assigned numerically higher values (for comparison tests) across an operating system name change and version reset. The value returned from this function is equivalent to that of the UNIX System Services callable service BPX1UNA. These values do not necessarily reflect the same release as that of Language Environment. Figure 22-5 on page 450 shows a sample of the uname() function.

```
#define _POSIX_SOURCE
#include <sys/utsname.h>
#include <stdio.h>
main() {
   struct utsname uts;
   if (uname(&uts) < 0)
      perror("uname() error");
      else {
      printf("Sysname:  %s\n", uts.sysname);
      printf("Nodename: %s\n", uts.nodename);
      printf("Release:  %s\n", uts.release);
      printf("Version:  %s\n", uts.version);
      printf("Machine:  %s\n", uts.machine);
   }
}
```

Figure 22-5 Sample for uname() function

Output from the uname() sample code is shown in Figure 22-6.

```
Sysname:  OS/390
Nodename: SC74
Release:  21.00
Version:  03
Machine:  2097
```

Figure 22-6 Output from the uname() sample

CEEPGID service

The CEEGPID callable service returns a version identifier (hexadecimal value) of the form ppvvrrmm identifying the level of Language Environment, where:

pp Product identifier (4 = z/OS)
vv Version
rr Release
mm Modification level

Figure 22-7 on page 451 shows a C example of using the CEEPGID callable service.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <leawi.h>
#include <ceeedcct.h>
int main(void) {
   _INT4 cee_ver_id, plat_id;
   _FEEDBACK fc;
   int Vmask = 0x00FF0000;
   int Rmask = 0x0000FF00;
   int Mmask = 0x000000FF;
   CEEGPID(&cee_ver_id, &plat_id, &fc);
   if ( _FBCHECK ( fc , CEE000 ) != 0 ) {
      printf("CEEGPID failed with message number %d\n", fc.tok_msgno);
      exit(2999);
   }
 if ((plat_id == 3) && (cee_ver_id > 290)) {
  printf("Language Environment version ID: 0x%08X\n",
         cee_ver_id);
  printf("         Version:     %17.1d\n",
         (Vmask & cee_ver_id)>>16);
  printf("         Release:       %17.1d\n",
         (Rmask & cee_ver_id)>>8);
  printf("    Modification:         %17.1d\n",
         Mmask & cee_ver_id);
 }
 else {
  printf("Language Environment version is: %d\n",
         cee_ver_id);
 }
}
```

Figure 22-7 Sample code for using CEEGPID

The output produced by this program is shown in Figure 22-8.

```
Language Environment version ID: 0x04010B00
         Version:                1
         Release:                11
    Modification:                0
```

Figure 22-8 Output from the CEEGPID service

22.2 CELQPIPI support

The user-supplied AMODE 64 GETSTORE and FREESTORE service routines can perform alternate storage management processing. The user-supplied AMODE 64 MSGRTN service routine can perform alternate message handling processing. Now you can specify our own AMODE 64 GETSTORE and FREESTORE service routines when calling CELQPIPI for the INIT_MAIN or INIT_SUB PreInit requests. Furthermore, you are now able to specify your own MSGRTN service routine when calling CELQPIPI for the INIT_MAIN or INIT_SUB PreInit requests.

The user-supplied GETSTORE and FREESTORE service routines get control in AMODE64 with the usual register and parameter passing conventions described in this book. For almost all AMODE 64 calls to GETSTORE, the routine gets control on the current Language Environment stack by using the XPLINK linkage. For the few places where this is not true, Language Environment will temporarily obtain storage to use as a stack.

GETSTORE and FREESTORE service routines must be specified together. If one is zero (0), then the call to CELQPIPI(init_main) or CELQPIPI(init_sub) fails.

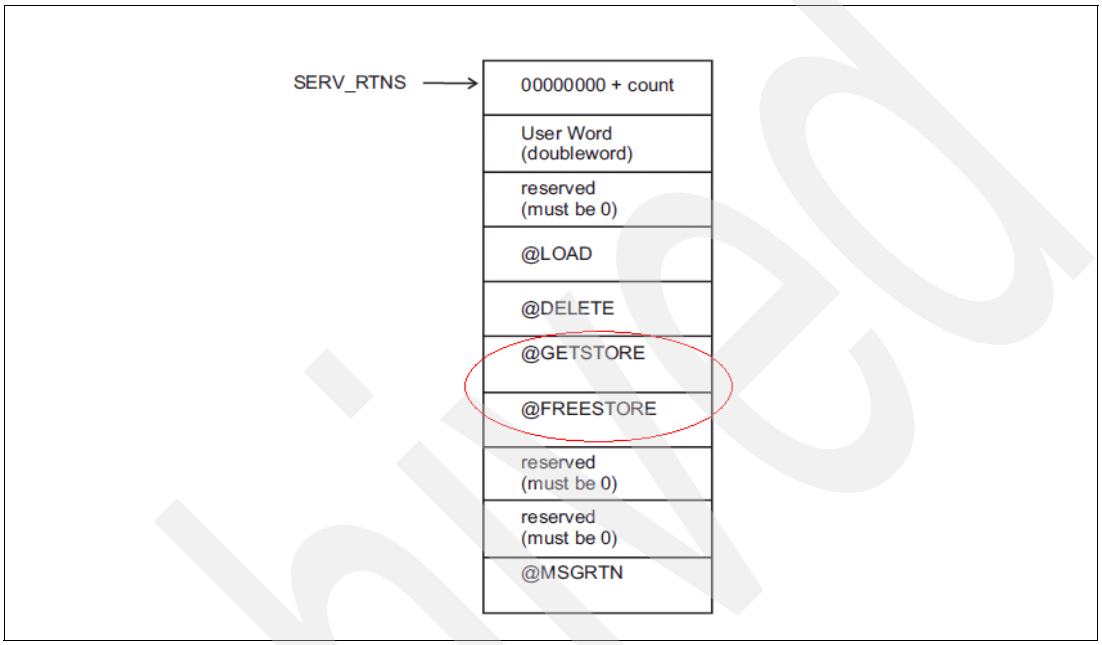


Figure 22-9 New storage support

Usage of GETSTORE

The GETSTORE service routine allocates storage on behalf of the Language Environment storage manager. A user can request three types of storage when invoking the GETSTORE service routine:

- Below the 16 MB line
- Below the 2 GB bar
- Above the 2 GB bar

Note: The obtained storage must be obtained in a key that is compatible with the application.

Table 22-1 and Table 22-2 on page 453 list the calling conventions of the new GETSTORE function.

Table 22-1 Input parameter for GETSTORE

Name	Description
PISA_Addr	A doubleword address of the PreInit storage attributes (PISA) control block
User word	A doubleword field containing the user word supplied in the Service Routine Vector

Table 22-2 Output parameter for GETSTORE

Name	Description
Stg_address	A doubleword address of the storage obtained or zero (0).
Obtained	An unsigned doubleword that contains the number of bytes (below the bar) or megabytes (above the bar) obtained.
Return code	The fullword return code from the @GETSTORE service
Reason code	The fullword reason code from the @GETSTORE service

Table 22-3 lists the return codes from the GETSTORE service.

Table 22-3 Return codes of GETSTORE

Return code	Reason code	Description
0	0	Successful.
8	0	The specified version of the PISA is not supported by this service routine.
16	0	Unsuccessful; an uncorrectable error occurred.

PreInit storage attributes control block

The PreInit storage attributes (PISA) control block, defined in macro CELQPIDF, contains the attributes that the GETSTORE routine is to use when obtaining the storage. The macro can be found in the SCEEMAC library in your system. Table 22-4 lists and describes the PISA fields and flags.

Table 22-4 PISA field descriptions

PISA field	Description
PisaVersion	A fullword field that contains the version number of this PISA. The only valid value for this field is currently 1, but other values may be added in the future when new storage attributes are added to this interface. The GETSTORE service routine checks this field to verify that it supports the specified version. If it cannot support the version, it returns a return code of 8.
PisaAmount	An unsigned doubleword that contains the amount of storage requested. For below the bar storage, the amount of storage to obtain is in the number of bytes. For above the bar storage, the size of the memory object to obtain is in the number of megabytes.

PISA field	Description
PisaFlags	A doubleword flag area. The flags are defined as follows: **PisaBelowtheLine** Bit zero in PisaFlags is ON if the requested storage is required to be below the 16 MB line. **PisaBelowtheBar** Bit one in PisaFlags is ON if the requested storage is required to be below the 2 GB bar. **PisaAbovetheBar** Bit two in PisaFlags is ON if the requested storage is required to be above the 2 GB bar. **PisaGuardLoc** Bit three in PisaFlags specifies whether the guard location is at the low virtual end or the high virtual end of the memory object. **PisaPageFrameSize1MEG** Bit four in PisaFlags is ON if the memory object is to be backed by 1 megabyte page frames (equivalent to IARV64 PageFrameSize= 1MEG). **PisaPageFrameSizeMAX** Bit five in PisaFlags is ON if the memory object is to be backed by the largest page frame size supported (equivalent to IARV64 PageFrameSize=MAX).
PisaSubpool_no	A fixed binary(31) subpool number 0-127.
PisaDumpPriority	A fullword dump priority for memory objects.
PisaGuardSize	A doubleword indicating the number of megabytes of guard area to be created at the high or low end of the memory object.
PisaUsertkn	A doubleword token to be associated with a group of memory objects.

PisaPageFrameSizeMAX bit

If you are using PisaPageFrameSizeMAX as 1 MEG, then both bits four and five are off and the memory object will be backed by 4 K page frames (equivalent to IARV64 PageFrameSize=4K). This flag is used for above the bar storage only. Failure to provide storage with this attribute may have a performance impact in your application.

If you use PisaPageFrameSizeMAX, then both bits four and five are off and the memory object will be backed by 4 K page frames (equivalent to IARV64 PageFrameSize=4K). This flag is used for above the bar storage only. Failure to provide storage with this attribute may have a performance impact in your application.

In the PisaGuardLoc field, bit three is OFF if the guard areas are created starting from the origin of the memory object (that is, from the low virtual end). Bit three is ON if the guard areas are created at the end of the memory object (that is, at the high virtual end). This flag is used for above the bar storage only, and only has meaning when PisaGuardSize is non-zero. Failure to guard the memory object as requested may cause Language Environment's stack management features to work incorrectly.

PisaSubpool_no

Language Environment allocates storage from the process-level storage pools. This field is used for below the bar storage only.

PisaDumpPriority

This field is used for above the bar storage only. Failure to provide storage with the requested dump priority may result in a dump that does not contain useful diagnostic information.

PisaGuardSize

This field is used for above the bar storage only. Failure to guard the memory object as requested may cause Language Environment's stack management features to work incorrectly.

PisaUsertkn

This can be used on a later @FREESTORE request to free all memory objects associated with this value. This field is used for above the bar storage only. Non-authorized callers will not be able to directly use the UserTkn provided by Language Environment because Language Environment uses the authorized word in the high half. Refer to the sample @GETSTORE routine to see one way of handling this.

Usage of FREESTORE

This function is the opposite of the GETSTORE function to free storage on behalf of the Language Environment storage manager. Table 22-5 lists and describes the input parameters for the FREESTORE function.

Table 22-5 Input parameters for FREESTORE

Name	Description
Stg_address	The doubleword address of the storage or memory object to free.
Amount	An unsigned doubleword that contains the amount of storage in bytes to free. For below the bar storage only.
Subpool_no	The fixed binary(31) subpool number 0-127. For below the bar storage only.
Flags	**MatchUsertkn** Bit zero in Flags indicates how the value specified in parameter UserTkn is to be used.
UserTkn	A doubleword token that identifies the memory object or a group of memory objects to be detached. For above the bar storage only.
UserWord	A doubleword field containing the user word supplied in the Service Routine Vector.

MatchUsertkn

If MatchUsertknis ON, then the @FREESTORE routine is expected to free all memory objects that are associated with this user token (this is equivalent to IARV64 MATCH=USERTKN). If MatchUsertkn is OFF, then @FREESTORE is to use this user token when freeing a single memory object with the origin in Stg_address (equivalent to MATCH=SINGLE). This is for above the bar storage only. Table 22-6 lists and describes details of the output fields for the FREESTORE function.

Table 22-6 Output parameter for FREESTORE

Name	Description
Stg_address	A doubleword address of the storage obtained or zero.

Name	Description
Obtained	An unsigned doubleword that contains the number of bytes (below the bar) or megabytes (above the bar) obtained.
Return code	The fullword return code from the @GETSTORE service.

The possible return codes are documented in Table 22-7.

Table 22-7 list of return codes of FREESTORE

Return Code	Reason Code	Description
0	0	Successful
16	0	Unsuccessful; uncorrectable error occurred

Usage of MSGRTN

MSGRTN allows error messages to be processed by the caller of the application. If the message pointer is zero (0), then your message routine is expected to return the size of the line to which messages are written (in the line_length field). This allows messages to be formatted correctly; that is, broken at places such as blanks. Table 22-8 lists and describes the calling conventions for the MSGRTN function.

Table 22-8 Input parameters for MSGRTN

Name	Description
Message	A pointer to the first byte of text that is printed, or zero (0)
Msg_len	The fixed binary (31) length of the message
UserWord	A doubleword user field

Table 22-9 lists the output parameter.

Table 22-9 Output parameter for MSGRTN

Name	Description
Line_length	The fixed binary (31) size of the output line length. This is used when Message is zero (0).

Table 22-10 lists and describes the possible return codes.

Table 22-10 List of return codes of MSGRTN

Return code	Reason code	Description
0	0	Successful
16	0	Unsuccessful; uncorrectable error occurred

Service routine samples can be found in SCEESAMP: a sample AMODE 64 Assembler GETSTORE and FREESTORE replacement service routine, CEEWQGST. The appropriate sample AMODE 64 Assembler MSGRTN replacement service routine is CEEWQMSG. Existing sample AMODE 64 PreInit driver program CEEWQPIP has been updated to use the replacement GETSTORE, FREESTORE, and MSGRTN routines.

The sample code is also provided in the Language Environment Programming Guide for 64-bit Virtual Addressing Mode.

Important: The new service routines will not be supported on releases prior to z/OS V1.11. If an application running on a prior release places addresses in the service routine vector for any of the new routines, it will fail as it does today.

22.3 CICS AFP support

In prior releases, Language Environment did not fully support binary floating point (BFP) or decimal floating point (DFP) operations in applications that run in the CICS TS environment. With this new support, binary and decimal floating point operations are fully supported in a CICS TS Version 4 or later environment.

The AFP(VOLATILE) compiler option is no longer required, and BFP/DFP error handling is similar to that in a non-CICS environment. Before this change, BFP and DFP 0C7 program checks were reported with the CEE3207 message when running in a CICS environment. With this change, these program checks will be reported with the usual CEE321x, CEE322x, and CEE323x messages that are used in non-CICS environments. The interface between Language Environment and CICS TS automatically activates the new support when CICS TS Version 4 or later is used.

The typical floating point 0C7 exceptions are now reported using the same Language Environment messages as in non-CICS environments. It is now possible to run many simultaneous programs in a CICS TS region that perform binary or decimal floating point operations with non-default rounding modes, with no interference between the applications. The AFP(VOLATILE) compiler option is no longer required, because Language Environment and CICS TS save and restore all floating point registers when switching between running transactions.

CEEDUMPs and formatted IPCS dumps will sometimes show additional registers after CICS program checks and ABENDs, as follows:

- Floating point registers 0-15 (before this change only 2, 4, 6, 8 were included)
- Floating point control register (FPC)
- High registers (and low registers, as before)
- Access registers

The FPC register and floating point registers 1, 3, 5, 7, and 8 to 15 appear if the application has used them previously. This is the same as in non-CICS environments. The access registers appear only after program checks in a CICS environment.

Note: The support is also available in Language Environment HLE7750 (z/OS 1.10) with APAR PK71900 and in Language Environment HLE7740 (z/OS 1.9) with APAR PK71900.

Certain binary and decimal floating point exceptions that were previously reported with a CEE3207 message are now reported with the following messages:

CEE3216, CEE3217, CEE3218, CEE3219, CEE3220,

CEE3221, CEE3222, CEE3223, CEE3224, CEE3225,

CEE3226, CEE3227, CEE3228, CEE3229, CEE3231,

CEE3232, CEE3233.

With this change, 0C7 exceptions are now reported with the same messages in both CICS and non-CICS environments. The sample CEEWUCHA (the USRHDLR program) has been modified to check for these CEE32xx messages in addition to CEE3207. Any customized copies of CEEWUCHA may need to be updated to properly handle binary and decimal floating point exceptions.

The floating point control register (FPC), floating point registers 1, 3, 5, 7, and 8-15, access registers (ARs), and high registers (HRs) may now saved and restored when applications are resumed after program checks and ABENDs when CICS TS Version 4 or higher is used. Any changes to these registers made by user condition handlers or signal catchers may be ignored when the registers are restored and the application is resumed.

22.4 Decimal floating point support

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 V1R11, as follows:

- New decimal floating point math functions are provided.
- New conversion specifiers for printf() family of functions are provided.

Table 22-11 lists and describes the new functions in the `math.h` header file.

Table 22-11 New added C/C++ functions

Functions	Description
cbrtd32(), cbrtd64(), cbrtd128()	Calculate the Cube Root
expm1d32(), expm1d64(), expm1d128()	Exponential Minus One
exp2d32(), exp2d64(), exp2d128()	Calculate the base-2 exponential
fmad32(), fmad64(), fmad128()	Multiply then add
fmodd32(), fmodd64(), fmodd128()	Calculate Floating-Point Remainder
hypotd32(), hypotd64(), hypotd128()	Calculate the square root of the squares of two arguments
log1pd32(), log1pd64(), log1pd128()	Natural Log of x + 1
log2d32(), log2d64(), log2d128()	Calculate the Base-2 Logarithm
quantexpd32(), quantexpd64(), quantexpd128()	Compute the quantum exponent
__remquod32(), __remquod64(), __remquod128()	Computes the remainder

The new formatting signs %a and %A are provided for the printf() family of functions to be used with Decimal Floating Point types.These new conversion specifiers determine which type of formatting style is to be used by printf(), either e/E or f/F style formatting.

The following requirements must met to use the new functions:

- Applications must be running on z/OS V1R11 or higher.
- APAR PK71899 rolls back the V1R11 decimal floating point support to V1R8. PTFs are available; see the following **Note:**.
- 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.

Note: The floating support is available with UK43813 for z/OS V1R8, UK43814 for z/OS V1R9, and UK43815 for z/OS V1R10 in the prior releases of z/OS.

22.5 Exploitation of large pages

In z/OS V1R11, IBM enhanced the Memory Object Services function - __moservices(). Introduced in V1R10, this function allows an application to create and free enclave-level memory objects with user-specified characteristics. For V1.11, the caller can request that the memory object be backed by large page frames. To use the large frame support, IBM has introduced two new flags to allocate the memory object, as listed and described in Table 22-12.

Table 22-12 New flags for __moservices()

Flag	Description
__MOPL_PAGEFRAMESIZE1MEG	The memory object will be backed by 1 Megabyte page frames. __moservices() returns -1 with errno EMVSERR if the current hardware does not have Large Page support or if large page frames are unavailable on the system.
__MOPL_PAGEFRAMESIZEMAX	The memory object will be backed by the largest page frame size supported, and backed by 4 K page frames if the current hardware does not have Large Page support or if large page frames are unavailable on the system. When 4 K page frames are used, the request will be successful, but __mopl_iarv64_rc and __mopl_iarv64_rsn will contain values to indicate the memory object is not backed by large page frames.

Notes:

- If none of these flags are set, then the memory object will be backed by 4 K page frames.
- The flags are mutually exclusive.

Figure 22-10 on page 460 shows a sample program using large page support. This program allows 100 MB of memory in a 1 MB segment to be backed up with large page frames by using __MOPL_PAGEFRAMESIZE1MEG. If no large page support is available, the function fails with RC=-1.

The __MOPL_PAGEFRAMESIZEMAX flag is more flexible. If no large page support is available on your system, then the memory is allocated in the traditional way of 4 k pages.

```
/*********************************************************************
 *                                                                   *
 * project      : ITSO redbook SG24-7729                             *
 *                                                                   *
 * function     : use of __moservices()                              *
 * requirements :                                                    *
 *                                                                   *
 * history      : 18.05.2009 Lutz       first creation               *
 *                                                                   *
 *********************************************************************/
 #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));

    /* try to get 500MB and enforce to backed 1M frames */
    mymopl.__mopldumppriority = __MO_DUMP_PRIORITY_STACK + 5;
    mymopl.__moplrequestsize  = 500;
    mymopl.__moplgetstorflags = __MOPL_PAGEFRAMESIZE1MEG;

    mos_rv = __moservices(__MO_GETSTOR,sizeof(mymopl),
                          &mymopl,&mymoptr);
    if (mos_rv == 0) {
     printf("moservices(GETSTOR) successful, MO addr: %p\n",
            mymoptr);
     mos_rv = __moservices(__MO_DETACH, 0, NULL, &mymoptr);
     }
    else
     printf("error getting storage \n");
    return(0);
 }
```

Figure 22-10 C Sample using __moservices

Figure 22-10 shows the compile with the LP64 option. Figure 22-11 on page 461 shows a sample job to compile and link.

```
//LUTZCP   JOB ,'COMPILE',NOTIFY=LUTZ,REGION=512M,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
/*JOBPARM S=SC74
//COMP     EXEC EDCQCB,
//        INFILE='LUTZ.SOURCE(MOSRV)',
//        OUTFILE='LUTZ.LOADLIB(MOSRV),DISP=SHR'
//BIND.SYSIN  DD *
  NAME MOSRV(R)
/*
//
```

Figure 22-11 Sample job to compile and link

After the program is compiled and executed, a `Successful execution` message similar to that shown in Figure 22-12 is displayed.

```
moservices(GETSTOR) successful, MO addr: 808800000
```

Figure 22-12 Successful execution

If something is wrong with your system installation, you will see instead the message shown in Figure 22-13. Unfortunately, there are nodifferent return codes for certain errors. Instead, RC=-1 is always returned, even in the following situations:

- z10 hardware is not being used.
- There is no LFAREA defined.
- You attempted to allocate more large frames than defined.

```
error getting storage -1
```

Figure 22-13 Failing program output

Using large page support

If large page support is not used, programs trying to use it will fail with the following message:

```
IARO20I THE LFAREA WAS SPECIFIED BUT LARGE PAGE SUPPORT IS NOT AVAILABLE
```

The use the large page support you have to check the following system settings:

- Place to LFAREA=xxxxM in your active IEASYSxx member
- Ensure that your z/OS have more than 4 GB main storage, LF support will be placed above
- You must running on a z10 EC hardware machine type 2097
- You have to have z/OS V1R11 installed.

22.6 File I/O tracing

C/C++ file I/O problems are often time-consuming to troubleshoot, because first an accurate and complete set of data must be gathered to describe the failure, and then someone needs

this data to recreate locally and debug the failure. There are typically three main questions that need to be answered when debugging a file I/O failure:

- How was the file opened?
- What is the file type and name?
- What I/O function calls lead to the failure?

Currently, as much of this information as possible is communicated through a user's description of the failure. However, it is time-consuming and difficult to accurately and completely gather this information, in particular the I/O calls leading to the failure. As a result, the accuracy and completeness of this data has a direct effect on the efforts required to recreate and debug the failure.

The goal of this line item is to provide a tracing mechanism that will allow a user to easily gather all of this information in one place, thus saving a significant amount of time for both users and troubleshooters.

In V1R11, IBM introduced a new environment variable:

```
_EDC_IO_TRACE=(Filter,Detail,Buffer Size)
```

If you set this variable before starting a program that you want to trace, then every file I/O will be recorded. Table 22-13 lists and describes the parameters for this variable.

Table 22-13 Parameter for _EDC_IO_TRACE

Parameter	Description
Filter	This indicates which files to trace. **//DD:filter** The trace will include the DD names matching the specified filter string. **//filter** The trace will include the MVS data sets matching the specified filter string. Member names of partitioned data sets cannot be matched without the use of a wildcard. **filter** The trace will include the z/OS UNIX System Services files matching the specified filter string. **Examples:** //DD:* The trace will include all DD names. //* The trace will include all MVS data sets. This is the default setting. /* The trace will include all z/OS UNIX files. * The trace will include all MVS data sets and z/OS UNIX files.
Detail	**0** No tracing will be performed. This is the default setting. **1** First-level tracing will be performed. Trace will include: The file being traced. The trace detail level and buffer size_EDC_IO_TRACE settings. Details describing how the file was opened (the function called and parameters passed). Formatted file data returned from fldata(). The file pointer address. The DD name (if applicable). The function flow details on entry to externally documented file I/O functions. **2** Second-level tracing will be performed. The trace will include everything that the first-level tracing includes, except that the function flow details will be for entry to externally documented and internal slot, exit, operating system, and open file I/O functions.

Parameter	Description
Buffer Size	**sizeK** This specifies the size of each file's trace buffer, where size is a number specified in kilobytes. The default buffer size is 16 K. **sizeM** This specifies the size of each file's trace buffer, where size is a number specified in megabytes.

Locating the file I/O trace

If the application is running under TSO or batch, and an EDCTRACE DD is not specified, Language Environment writes the trace to the batch log (SYSOUT=* by default). The SYSOUT class may be changed by specifying an EDCTRACE DD, or by setting the environment variable, _CEE_DMPTARG=SYSOUT(x), where x is the preferred SYSOUT class.

If the application runs under z/OS UNIX System Services and either runs in an address space after fork() was invoked, or is invoked by one of the exec family of functions, then the trace is written to the z/OS UNIX file system. Language Environment writes the trace to one of the following directories, in the specified order:

- The directory found in environment variable _CEE_DMPTARG, if found
- The current working directory, if the directory is writable and the EDCTRACE path name does not exceed 1024 characters
- The directory found in environment variable TMPDIR (an environment variable that indicates the location of a temporary directory if it is not `/tmp`)
- The `/tmp` directory

The name of this file uses the following format:

```
/path/EDCTRACE.Date.Time.Pid
```

Where:

Path This is the path determined from the preceding algorithm.

Date This is the date when the trace is taken, appearing in the format YYYYMMDD (such as 20071122 for November 22, 2007).

Time This is the time that the trace is taken, appearing in the format HHMMSS (such as 115601 for 11:56:01 a.m.).

Pid This the process ID that the application is running in when the trace is taken.

22.7 Heap pool improvements

z/OS V1R11 has several heap pool improvements. Heap storage is used to allocate storage that has a lifetime not related to the execution of the current routine; it remains allocated until you explicitly free it or until the enclave terminates.

Heap storage is typically controlled by the programmer through Language Environment run-time options and callable services. For z/OS Language Environment, heap storage is made up of one or more heap segments comprised of an initial heap segment, and, as needed, one or more heap increments, allocated as additional storage is required.

The initial heap may or may not be preallocated prior to the start of the application code, depending on the type of heap. Each heap segment is subdivided into individual heap elements. Heap elements are obtained by a call to one of the heap allocation functions, and

are allocated within the initial heap segment by the z/OS Language Environment storage management routines. When the initial heap segment becomes full, Language Environment gets another segment, or increment, from the operating system.

Storage management terminology

The following terminology is used when referencing common storage management services:

Stack This is an area of storage in which stack frames can be allocated.

Heap This is an area of storage used by Language Environment routines. The heap consists of the initial heap segment and zero or more increments.

Heap element This is a contiguous area of storage allocated by a call to the CEEGTST service. Heap elements are always allocated within a single heap segment.

Heap increment This refers to additional heap segments that are allocated when the initial heap segment does not have enough free storage to satisfy a request for heap storage.

Heap pool This is a storage pool that, when used by the storage manager, can be used to improve the performance of heap storage allocation. This can improve the performance of a multi-threaded application.

Heap segment This is a contiguous area of storage obtained directly from the operating system.

Note: If you use CEEDOPT, CEECOPT, or CELQDOPT ++USERMODs to set installation-wide default run-time options for Language Environment, then the ++USERMODs must be updated to include the four new suboptions of the HEAPCHK run-time option.

HEAPCHK statement

Figure 22-14 shows the syntax of the enhanced HEAPCHK statement for Language Environment runtime options.

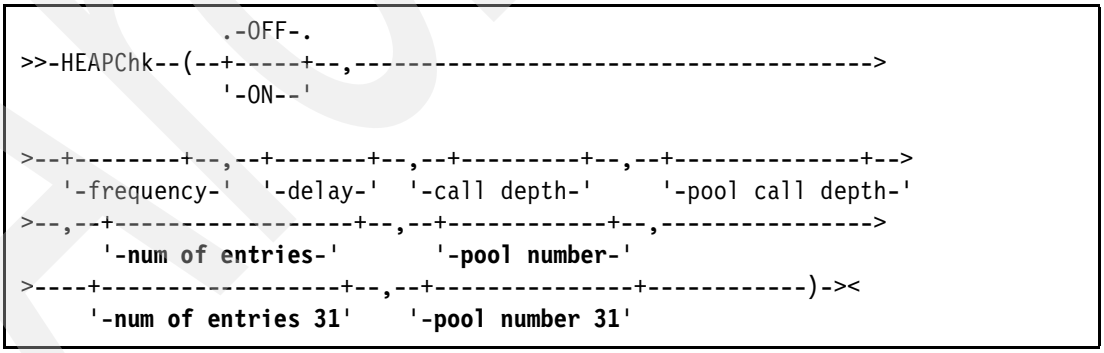


Figure 22-14 New HEAPCHP options

Number of entries

This option specifies the number of entries to be recorded in the heap pool trace table for the main user heap in the application. If the heap pool trace table is available and Number of Entries is 0, then the heap pool trace table is not generated.

Pool number

This option specifies to filter the entries of the heap pool trace table, recording only those entries of a specific pool ID for the main user heap in the application. The value is to be a valid pool number (1-12). If the heap pool trace table is available and Pool Number is 0, then the entries of all pools will be traced.

IPCS formatting the heap pools trace

This option specifies to format heap pool information and create a report that contains information about heap pools and extents, validating extents and cells. This report will be part of the current IPCS Verbexit LEDATA 'HEAP' option.

Expand the information available to heap pool activities by formatting information about:

- QPCB
- QPCB Entry for each pool
- Addresses
- Free chain validation (validating that pointer is within a valid extent and cell pool number is valid)
  - No need to print out each free cell in this phase
- Extent validation and display
  - Address and size of extent
  - Display each free and allocated cell and validate if cell pool number in header is correct

LEDATA support

To invoke the enhanced LEDATA support, use the following command:

```
IP VERBEXIT LEDATA [ 'parameter [,parameter]...' ]
IP VERBEXIT CEEERRIP [ 'parameter [,parameter]...' ]
```

You can shorten the command as follows:

```
IP VERBX LEDATA [ 'parameter [,parameter]...' ]
```

VERBX command

The parameters given to the VERBX command are covered here.

Report type parameters on the SUMMARY statement are as follows:

[SUMMARY]

[HEAP | STACK | SM]

[HPT(value) | **HPTTCB (value) | HPTCELL(value) | HPTLOC(value)]**

[CM]

[MH]

[CEEDUMP]

|[PTBL(value)]

[ALL]

Use these parameters to select the type of report. You can specify as many reports as you want. If you omit the parameters, the default is SUMMARY. The parameters are described as follows:

HEAP This specifies a report on Storage Management control blocks pertaining to HEAP storage. It also specifies a heap pool report with information useful to find potential damaged cells.

STACK This specifies a report on Storage Management control blocks pertaining to STACK storage.

SM This specifies a report on Storage Management control blocks. This is the same as specifying both HEAP and STACK.

HPT This requests that the heap pool trace (if available) be formatted. If the value is zero (0) or an asterisk (*), then the trace for every heap pool's pool ID is formatted. If the value is a single number (1-12), then trace for the specific heap pool s pool ID is formatted. If no filter is specified, then all entries are formatted for the specific pool ID.

Note: Filter options without specifying HPT implies HPT(*).

HPTTCB This filters the heap pool trace table (if available), printing only those entries for a given TCB address (value).

HPTCELL This filters the heap pool trace table (if available), printing only those entries for a given cell address (value).

HPTLOC This filters the heap pool trace table (if available), printing only those entries for a given virtual storage location (value). The valid values are the following:

- 31: - Display entries located on virtual storage below the bar.
- 64: - Display entries located on virtual storage above the bar.
- ALL: - Entries located on virtual storage below and above the bar.

Note: Users can specify multiple options together (like HPTTCB and HPTCELL). All pieces of information must match the trace entry for it to be formatted.

If location and cell contradict each other, then an error will be displayed (like HPTLOC(31) and HPTCELL(64bit addr)).

Heappool Report sections of the LEDATA output

The Heappool Report shown in Figure 22-15 on page 467 provides information about each cell pool. It provides detailed information about the free chain associated with every qpcb pool data area and all the free and allocated cells in the extent chain. It also contains information related to errors found when the cells are validated.

Within each cell pool, Language Environment keeps track of unallocated cells by chaining them together. The LEDATA HEAP option formats the free chain within each cell pool, and validates that:

- The cell pointer is within a valid extent
- The cell pool number is valid

If the formatter finds a problem, then it will place an error message describing the problem directly after the formatted line of the cell that failed validation.

The LEDATA HEAP option produces a report that lists all of the cells within each pool extent, and identifies the cells as either allocated or freed. For each allocated cell the contents of the first X'20' bytes of the area are displayed to help identify the reason for the storage allocation.

The formatter validates whether the cell pool number in the header is correct. If a problem is found, then an error message describing the problem is placed after the formatted line of the cell that failed validation.

```
Heappool Report

   QPCB:  00014D30
   EYE_CATCHER:QPCB

   Data for pool 1:

   qpcb pool data area:  00014D48
   FIRST Cell:25995000    LAST Cell:25995000

       [1] Free Chain Tree for Pool 1

        Free Cell Addresses
        259950B0

   Heappool Extent Mapping

      qpcb extent:  25995000
      EYE_CATCHER:EX31

      [2] Map of Heappool Extent for Pool 1

         To display entire pool extents: IP LIST 25995000 LEN(X'00008000')
         ASID(X'0021')

         25995020: Allocated storage cell, length=00000038. To display: IP LIST          25995020 LEN(X'00000038') ASID(X'0021')
         25995028: C3C4D3D3 00000000 40000000 00000000  24700F98 24703F70
         25993870 00000490 |CDLL.... ..........q.....r.......|

         25995058: Allocated storage cell, length=00000038. To display: IP LIST
         25995058 LEN(X'00000038') ASID(X'0021')25995060: C3C4D3D3 25995028
         80000000 00000000  247006F0 24700770 2471CEB0
         00000150 |CDLL.r&............0...........&|

         25995090: Allocated storage cell, length=00000010. To display: IP LIST
         25995090 LEN(X'00000010') ASID(X'0021')

         25995098: 259ADBB8 00000000                                                |........                       |

         259950A0: Allocated storage cell, length=00000010. To display: IP LIST          259950A0 LEN(X'00000010') ASID(X'0021')
         259950A8: 259ADBE0 00000000                                                |........                       |

         259950B0: Free storage cell, length=00007F50. To display: IP LIST
         259950B0 LEN(X'00007F50') ASID(X'0021')

   Summary of analysis for Heappool Extent for Pool 1:

   Amounts of identified Cells:  Free:00007F50  Allocated:00000090 Total:00007FE0
   Number  of identified Cells:  Free:       1  Allocated:       4 Total:       5
   00000000 Cells were not accounted for.
   No errors were found while processing this heappool Extent.
```

Figure 22-15 Sample heappool report

22.8 C/C++ runtime enhancements

With z/OS V1R11, the following minor changes were implemented in the c/C++ runtime environment.

XL C/C++ runtime EAV 2 support

Using this support, with an XL C/C++ application you can now process extended format sequential data sets that have extended attributes, including those data sets that reside in the extended addressing space.

First use fopen() against an existing extended format sequential data set that resides in the extended addressing space. Then use other FILE functions to process the stream.

Globalization White Paper 2007

A new locale was defined, because of the separation of Serbia from Montenegro. The main difference between the new Serbia local and the existing Serbia-Montenegro is the currency code. The new currency code for Serbia is RSD. To invoke this support, use the command shown in Figure 22-16.

```
setlocale(LC_ALL,"sr_RS.UTF-8");
```

Figure 22-16 Setlocale for Serbia

CCSID functions under CICS

This support provides the ability for XL C/C++ applications running under CICS to convert between code set names and their CCSID and also determine the type for a given code set name or CCSID (ASCII or EBCDIC). Now clients can use the following functions in CICS:

- __toCcsid()
- __CSNameType()
- __toCSName()
- __CcsidType()

Note: Be sure to update the CICS System Definition (CSD) file using the program definitions in the CEECCSD member (and CEECCSDX member for CICS TS 3.1) found in the SCEESAMP data set. A module was added to support the CCSID functions under CICS.

22.9 Changes to assembler macros

In prior releases, inconsistencies between the macros provided by the various Language Environment products (z/OS, z/VM, and z/VSE) presented challenges for vendors who wanted to write simpler code to be ported between operating systems. In z/OS V1R11, the following changes were made:

- A CEEGLOB assembler macro similar to IBM Language Environment for z/VSE was created.
- Support in CEEPPA was added for the SERVICE keyword option.
- Support in CEEENTRY was added for the RMODE and AMODE keyword options.
- Support in CEEFETCH was added to handle both Language Environment and non-Language Environment code. Support was added to perform a “Language Environment-load” if the module was previously loaded.

CEEGLOB macro

The CEEGLOB macro generates a set of global assembler variables. These global assembler variables can be tested and used at assembly time to verify the availability of services and functions that require specific levels of Language Environment or operating systems. Table 22-14 lists the CEEGLOB variables which were added.

Table 22-14 Added CEEGLOB variables

Variable	Description	Example
&CEEGVER (alias &GVER)	Product number	Set to 4 for z/OS V1R11M0
&CEEGREL (alias &GREL)	Product release	Set to 11 for z/OS V1R11M0
&CEEGMOD (alias &GMOD)	Product modification level	Set to 0 for z/OS V1R11M0
&CEEGENV (alias &GENV)	Operating system environment from which the macro has been invoked	Set to 3 for z/OS

CEEPPA for the SERVICE keyword

The SERVICE keyword can only be specified on the first CEEPPA macro in the assembler source; all other instances of the keyword are ignored. When the SERVICE keyword is in use, the time stamp is generated automatically.

The TSTAMP option is forced to YES even when the user specified TSTAMP=NO. If the TSTAMP option is forced to YES, then the following severity 4 MNOTE is generated: SERVICE PARAMETER SPECIFIED TSTAMP PARAMETER FORCED TO 'YES'.

```
Example setting the SERVICE string with a concatenation of the CEEGLOB variables:
      GBLC  &GVER,&GREL,&GMOD
      CEEGLOB
ASMTSTRC CEEENTRY PPA=MYPPA,BASE=R11,MAIN=YES
         LA    3,12
         ST    3,RETCODE
         LA    2,8
         LA    3,0
         ST    2,0(,3)
         CEETERM RC=RETCODE,MODIFIER=0
*
RETCODE  DS    F
R3       EQU   3
R11      EQU   11
         LTORG  ,
* The service level string is set to the concatenation of the CEEGLOB values for
* the Version, Release and Modification Level
MYPPA    CEEPPA SERVICE=&GVER.&GREL.&GMOD
         CEEDSA ,
         CEECAA ,
         CEEOCB ,
         END    ASMTSTRC
```

Figure 22-17 Usage of CEEGLOB

SERVICE keyword

A new SERVICE keyword is added to set the service level string for a routine.

```
Syntax: SERVICE=service_string
```

The service string length and contents are located following the time stamp and version information, as shown in Figure 22-18.

```
188+*************************************************************************
189+*                T I M E    S T A M P                                   *
190+*************************************************************************
191+*
192+*        Time Stamp
193+*,Time Stamp = 2009/02/02 15:16:00                                      01-CEEPP
194+*,Version 1 Release 1 Modification 0                                    01-CEEPP
195+CEETIMES DS    OF                                                      01-CEEPP
196+         DC    CL4'2009'          Year                                 01-CEEPP
197+         DC    CL2'02'            Month                                01-CEEPP
198+         DC    CL2'02'            Day                                  01-CEEPP
199+         DC    CL2'15'            Hours                                01-CEEPP
200+         DC    CL2'16'            Minutes                              01-CEEPP
201+         DC    CL2'00'            Seconds                              01-CEEPP
202+         DC    CL2'1'             Version                              01-CEEPP
203+         DC    CL2'1'             Release                              01-CEEPP
204+         DC    CL2'0'             Modification                         01-CEEPP
205+         DC    AL2(6)             Length of Service String      @D2A   01-CEEPP
206+         DC    C'011100'          Service parm                  @D2A   01-CEEPP
207          CEEDSA  ,
```

Figure 22-18 Time stamp example listing

Note that this field is not interrogated by Language Environment.

The SERVICE keyword can only be specified on the first CEEPPA macro in the assembler source; all other instances of the keyword are ignored. When the SERVICE keyword is in use, the time stamp is generated automatically.

The TSTAMP option is forced to YES even when the user specified TSTAMP=NO. If the TSTAMP option is forced to YES, then the following severity 4 MNOTE is generated:

```
SERVICE PARAMETER SPECIFIED TSTAMP PARAMETER FORCED TO 'YES'
```

Figure 22-19 on page 471 illustrates setting the SERVICE string with a concatenation of the CEEGLOG variables.

```
Example setting the SERVICE string with a concatenation of the CEEGLOB
variables:
      GBLC  &GVER,&GREL,&GMOD
      CEEGLOB
ASMTSTRC CEEENTRY PPA=MYPPA,BASE=R11,MAIN=YES
         LA    3,12
         ST    3,RETCODE
         LA    2,8
         LA    3,0
         ST    2,0(,3)
         CEETERM RC=RETCODE,MODIFIER=0
*
RETCODE  DS    F
R3       EQU   3
R11      EQU   11
         LTORG  ,
* The service level string is set to the concatenation of the CEEGLOB values
for
* the Version, Release and Modification Level
MYPPA    CEEPPA SERVICE=&GVER.&GREL.&GMOD
         CEEDSA ,
         CEECAA ,
         CEEOCB ,
         END    ASMTSTRC
```

Figure 22-19 Usage of CEEGLOB

Figure 22-20 shows an extract of a CEE-Dup taken on a z/OS 1.11 system. The service level string 011100 is a concatenation of the Version, Release, and Modification level values obtained from the CEEGLOB.

```
Output showing the usage of SERVICE on the CEEPPA set to the values obtained from the CEEGLOB macro:
CEE3204S The system detected a protection exception (System Completion Code=0C4).
         From compile unit ASMTSTRC at entry point ASMTSTRC at compile unit offset +0000008A at entry offset
+0000008A
         at address 0006D08A.
CEE3DMP V1 R10.0: Condition processing resulted in the unhandled condition.       02/11/09 2:27:40 PM
ASID: 01DB   Job ID: TSU01819   Job name: GORELIK    Step name: Y36PROC    UserID: GORELIK

CEE3845I CEEDUMP Processing started.

Information for enclave ASMTSTRC

  Information for thread 8000000000000000

  Traceback:
    DSA   Entry       E  Offset  Statement   Load Mod             Program Unit          Service  Status
    1     CEEHDSP     +00004B34              CEEPLPKA             CEEHDSP               HLE7750  Call
    2     ASMTSTRC    +0000008A              ASMRC01G             ASMTSTRC              011100 Exception

    DSA   DSA Addr   E  Addr    PU Addr    PU Offset  Comp Date  Compile Attributes
    1     2159C0B0   0D1BB3E0   0D1BB3E0   +00004B34  20080319   CEL
    2     2159C030   0006D000   0006D000   +0000008A  20080512   ASM

  Condition Information for Active Routines
    Condition Information for ASMTSTRC (DSA address 2159C030)
      CIB Address: 2159C9A8
        .
```

Figure 22-20 Sample part of CEEDUMP

22.10 Swap and make context

Earlier releases of z/OS were unable to use the C Runtime Library context functions makecontext() and swapcontext() to implement co-routines in an AMODE 64 application. In release 11 of z/OS, IBM has modified the following C Runtime functions:

- getcontext()
- setcontext()
- makecontext()
- swapcontext()

In earlier releases, swapcontext() can fail in AMODE 64 applications with the following message even when an equivalent AMODE 31 application runs successfully:

```
CEE0250S An unrecognized label variable was detected.  The stack frame address
could not be associated with an active stack frame.
```

For z/OS V1R11, the interfaces for the context functions remain unchanged. However, the internals have changed and a new C/C++ environment variable _EDC_CONTEXT_GUARD was added.

This variable allows the user to control the method used to handle the guard page for AMODE 64 user context stacks. Valid values are ACTIVE (the default) and INUSE. Table 22-15 lists and describes these values.

Table 22-15 Values for _EDC_CONTEXT_GUARD

_EDC_CONTEXT_GUARD	Description
ACTIVE	The guard page for a user context stack is guarded each time the context is given control, and it is unguarded each time the context gives up control. This is the default behavior when _EDC_CONTEXT_GUARD is not set.
INUSE	The guard page for a user context stack is guarded the first time the context is given control, and it is unguarded when the context has run to completion (that is, when the function specified on the call to makecontext() returns or exits).

The INUSE method improves the application performance by eliminating most of the guarding and unguarding. But if the application does not adhere to the restrictions, abends could occur in unexpected places due to the reuse of guarded storage. This setting comes with the following restrictions:

- The storage for user context stacks must be allocated from the heap.
- The storage for a user context stack cannot be reused or freed until the last context to use the stack runs to completion.

22.11 C/C++ Compiler dependencies

As part of ongoing enhancements being made to the z/OS XL C/C++ compiler, the Language Environment C/C++ runtime needs to make changes to support the z/OS XL C/C++ compiler direction.

In z/OS V1R11, the following Compiler changes were made:

- Inline versions of four wmem* functions
- Extended character types
- Detect accidental use of Language Environment C headers under Metal C

Now user applications may be able to gain performance improvements through the use of the inline versions of four wmem* functions instead of library calls.

Applications may be able to make use of the extended character types when needing to support various encoding forms, such as Unicode.

Metal C programmers can find build problems more quickly when accidentally using the Language Environment C headers during their compilations.

23

IBM z/OS Management Facility

IBM z/OS Management Facility V1R11 (z/OSMF V1R11) provides a framework for managing various aspects of a z/OS system through a Web browser interface. By streamlining various traditional tasks and automating others, z/OSMF can help to simplify areas of system management and reduce the level of expertise needed for managing a system.

z/OSMF provides a single platform for hosting the Web-based administrative console functions of IBM server, software, and storage products. With z/OSMF, you manage solutions rather than specific IBM products. Because z/OSMF provides system management solutions in a task-oriented, Web browser-based user interface with integrated user assistance, both new and experienced system programmers can more easily manage the day-to-day operations and administration of the mainframe z/OS systems.

z/OSMF provides you with a single point of control for:

- Performing common system administration tasks
- Defining and updating policies that affect system behavior
- Performing problem data management

  z/OSMF allows you to communicate with the z/OS system through a Web browser, so you can access and manage your z/OS system from anywhere.

This chapter describes IBM z/OS Management Facility (z/OSMF). This facility helps z/OS system programmers manage and configure the z/OS environment. z/OSMF provides the infrastructure, services, and user interfaces to support a modern, Web browser-based management console for z/OS. System programmers, administrators, and operators can manage and administer a z/OS system more easily. z/OS Management Facility V1.11.0 provides a set of tools to aid in problem data management.

23.1 z/OSMF implementation with z/OS V1R11

Today, IT departments are organized in multiple domains due to the complexity of the various environments and tasks that are involved, as illustrated in Figure 23-1. Systems require cross-domain collaboration as well as modern and integrated tools to enhance productivity. IBM z/OS Management Facility (z/OSMF) provides a single platform for hosting the Web-based administrative console functions of an IBM server, software, and storage products. With this product, you can manage *solutions* rather than specific IBM products.

z/OSMF is provided as a no fee, separately licensed and entitled z/OS program product. z/OS Management Facility consists of the following FMIDs:

- HSMA110
- HBBN700

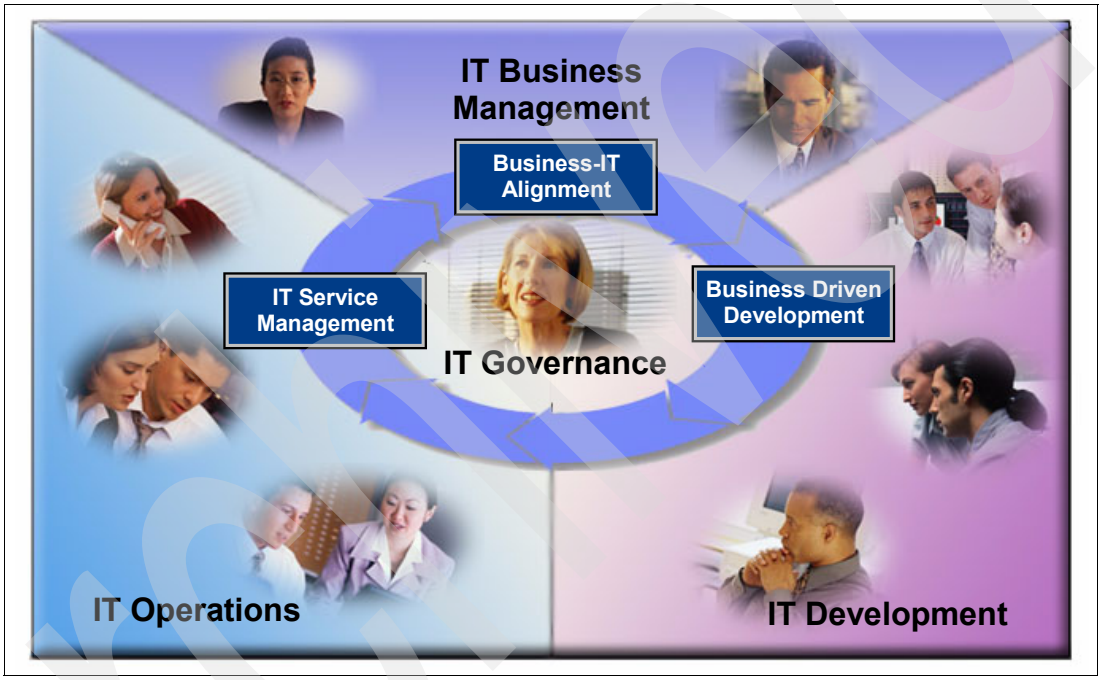


Figure 23-1 IT domains

23.2 z/OSMF overview

z/OSMF is a new, central system management function for z/OS. It is designed to provide better tools for managing systems and helping system programmers to be more productive. z/OSMF provides a framework for managing various aspects of z/OS systems through an intuitive Web 2.0 browser user interface and new enabling technologies on z/OS.

Note: Web 2.0 refers to what is generally understood as a second generation of Web development and design. z/OSMF as a Web 2.0-based solution incorporates a browser interface that communicates with the z/OS system.

z/OSMF applications

z/OSMF applications utilize the Dojo framework and Java script and run on a special version of WebSphere Application Server. The applications exploit functions provided by z/OS system

components. Everything is installed on the z/OS server; there are no client-side install requirements.

By streamlining various traditional tasks and automating others, z/OSMF can help you to simplify areas of system management and reduce the level of expertise needed for managing a sysplex. Daily operations and administration of the mainframe become easier to manage for both new and experienced system programmers. z/OSMF is a new zero-priced product.

Based on client engagements and feedback, the basic areas of system programming that need to be addressed are listed in Figure 23-2.

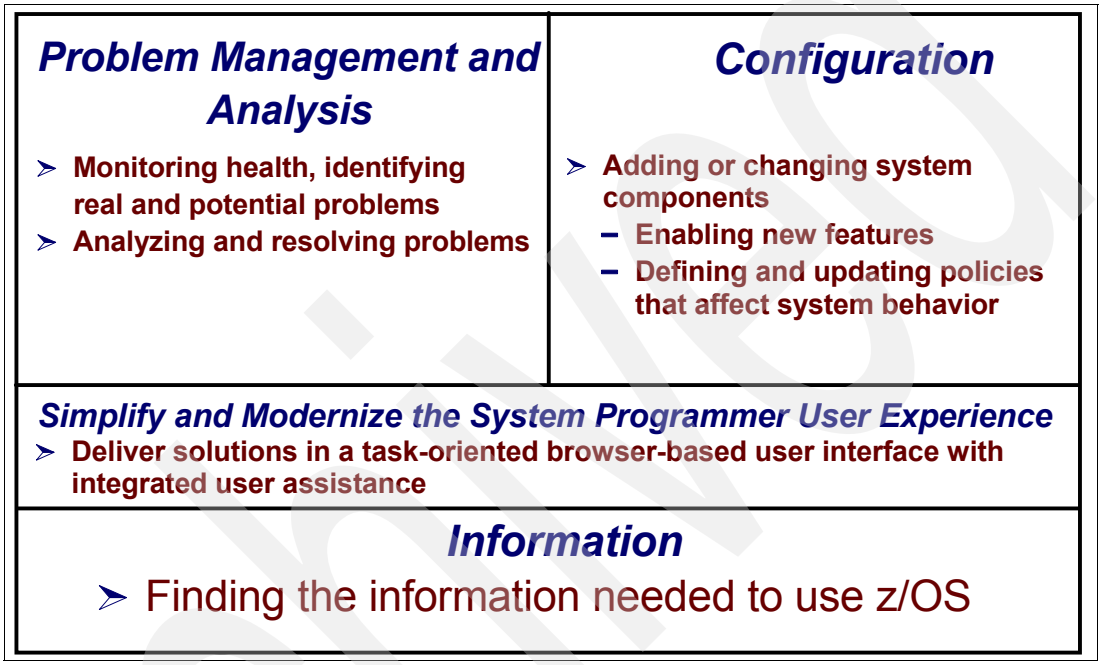


Figure 23-2 z/OSMF main focus

23.2.1 z/OSMF functional capabilities

The main focus of z/OSMF is simplification. It will help system programmers who are new to the z/OS environment to perform routine tasks more quickly by:

- Providing a modern, browser-based user interface that is more familiar to users who are new to the z/OS platform
- Automating tasks, thus reducing the learning curve
- Embedding active user assistance in the UI (for example, wizards that guide users through tasks; interactive troubleshooting aids)

System programmer tasks

z/OSMF can optimize system programmer tasks so that a mixed skilled workforce can maximize productivity and quality. z/OSMF allows you to perform the following system management functions:

- Problem data management by using an incident log, which helps to centralize problem data for your system and simplifies the process of sending diagnostic data to IBM
- Services for managing user access to the z/OSMF product
- Services for adding links for external tools to the z/OSMF navigation area

The main challenges faced by the system programmers are listed in Table 23-1.

Table 23-1 System programmer challenges

Novice	Experienced
Problem analysis and management	Too little time and too many tasks with fewer staff, requiring increased productivity
Getting the “Big Picture”	Ageing workforce (people retiring)
Gaining organizational knowledge	Responsibilities spanning many products and platforms
Product documentation	
Gaining enough (or appropriate) experience	
Unfamiliar concepts and tools	
Tasks that require detailed knowledge of command syntax and formats	
Gaining the trust of more experienced colleagues	

23.2.2 z/OSMF installation

z/OSMF with z/OS V1R11, the initial release of the product, provides the infrastructure, services, and user interfaces that allow you to perform the following functions:

- Configure TCP/IP policy-based networking functions on z/OS V1R11 systems and later.
- Manage user access to the z/OSMF product.
- Add links for external Web applications and Web sites to the z/OSMF navigation area.
- Perform problem data management tasks through the Incident Log, which centralizes problem data for your system and simplifies the process of sending diagnostic data to IBM

One instance of z/OSMF can manage only one local system or sysplex. Multiple users may log into the same instance of z/OSMF from separate workstations or browsers.

From one workstation the user can manage additional sysplexes by opening new browser windows (or tabs) and logging into the z/OSMF instance installed on those sysplexes (one browser per system or sysplex).

Only one active instance of z/OSMF is supported within a sysplex at any point in time. An additional instance can be created (for example, for test or service update or backup), but it is not to be actively managing the systems at the same time (for example, working on the same incident concurrently from two separate instances of z/OSMF) or using the same data repository.

23.2.3 z/OSMF and related system components

z/OSMF is offered by IBM as a separately licensed program product for z/OS.

Structurally, z/OSMF is comprised of a Web browser interface that communicates with the z/OSMF application running on the z/OS host system. Depending on the system management task to be performed, z/OSMF interfaces with other z/OS components to offer a

simplified interface for performing tasks. These components make up the environment necessary for using the functions available in z/OSMF. No separate client install is required.

z/OSMF is shipped with the following software:

- IBM WebSphere Application Server OEM Edition for z/OS Version 7.0, which provides a native Web services runtime environment for z/OSMF. It is new for z/OS V1R10.
- A set of administration and system management tasks that run on IBM WebSphere Application Server OEM Edition for z/OS.
- Technologies for serving the Web browser interface, such as Java Script and a Dojo framework.

Note: IBM WebSphere Application Server OEM Edition for z/OS Version 7.0 provides a native Web services runtime environment that is equivalent to the z/OS version of WebSphere Application Server Version 7.0.

This environment is used to deploy and host Java Platform, Enterprise Edition (Java EE)-based z/OS system applications. It is completely separate from any licensed copy of WebSphere Application Server that you might possess.

23.2.4 z/OSMF Web interface and z/OS V1R11

The goal of this architecture is to provide simplified systems management functionality through a common, easy-to-use interface. As an example, Figure 23-3 on page 480 shows the architecture flow for the Incident Log task, starting with z/OSMF and continuing through IBM WebSphere Application Server OEM Edition for z/OS, with information passed to the CIM server, through CEA, and finally System REXX.

z/OSMF is shipped with the following software:

Dojo A Dojo framework and JavaScript is an Open Source DHTML toolkit written in JavaScript. It builds on several contributed code bases (nWidgets, Burstlib, f(m)), which is why it is sometimes referred to as a “unified” toolkit.

Dojo aims to solve long-standing, historical problems with DHTML that prevented mass adoption of dynamic Web application development. Dojo allows you to easily build dynamic capabilities into Web pages and any other environment that supports JavaScript. Use the components that Dojo provides to make Web sites more usable, responsive, and functional.

WebSphere IBM WebSphere Application Server OEM Edition for z/OS provides the runtime environment for the systems management applications. It is new for the z/OS V1R10 environment, and is installed separately.

Included is a set of administration and system management tasks that run on IBM WebSphere Application Server OEM Edition for z/OS. When operational, z/OSMF uses services provided by the following z/OS components:

CIM The Common Information Model (CIM) server running on the host z/OS system provides the z/OS data and administrative capability.

CEA The Common Event Adapter (CEA) component enables CIM providers to identify, receive, and process selected z/OS events.

System REXX The System REXX component provides an infrastructure through which REXX execs can be run outside the normal TSO/E or batch environments by using a simple programming interface.

Incident Log task Depending on the system management task being performed by the user, z/OSMF makes use of new enabling technologies on z/OS. For example, the Incident Log task of z/OSMF uses services provided by the following z/OS components:

- CIM
- CEA
- System REXX (SYSREXX)

This software is illustrated in Figure 23-3.

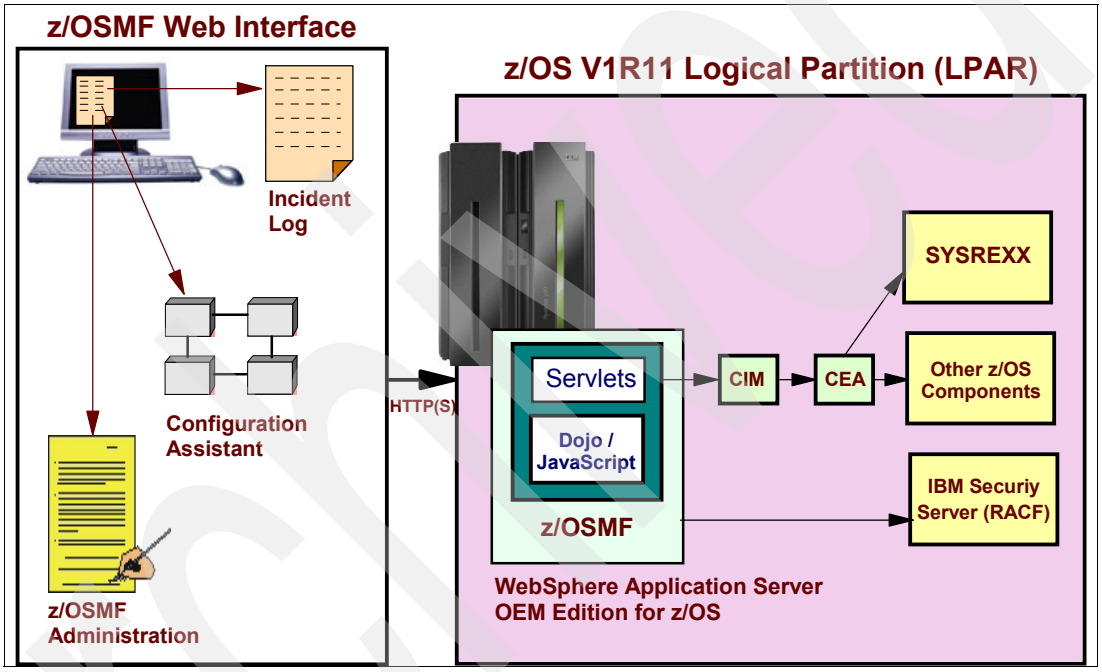


Figure 23-3 z/OSMF components interaction example

23.2.5 z/OSMF setup overview

To set up z/OSMF, the system programmer runs a series of z/OSMF configuration shell scripts. These scripts perform a number of setup tasks, including setting up the z/OSMF data file system, deploying z/OSMF into IBM WebSphere Application Server OEM Edition for z/OS, and creating command templates for authorizing users to system resources. During this processing the scripts issue messages and create log statements to indicate the actions performed, as well as any setup errors encountered.

The following shell scripts are provided with the product:

- To set up IBM WebSphere Application Server OEM Edition for z/OS, the system programmer runs a WebSphere configuration shell script that prompts for and saves configuration settings and creates the necessary jobs to run. Other WebSphere scripts run after the initial script to create the instance of IBM WebSphere Application Server OEM Edition for z/OS and complete the setup.

- To set up z/OSMF, the system programmer runs a series of z/OSMF configuration shell scripts. These scripts perform a number of setup tasks, including setting up the z/OSMF data file system, deploying z/OSMF into IBM WebSphere Application Server OEM Edition for z/OS, and creating command templates for authorizing users to system resources. During this processing the scripts issue messages and create log statements to indicate the actions performed, as well as any setup errors encountered.

The z/OSMF configuration scripts create RACF commands in a REXX exec that your security administrator verifies and runs. Although the scripts generate RACF commands by default, you can create a script with equivalent SAF commands if your installation uses another security management product.

The z/OSMF configuration process creates an initial administrator user ID. With this user ID, the administrator can log into z/OSMF and add more users and administrators as needed.

Note: For detailed configuration information about z/OSMF and WebSphere Application Server OEM Edition, see the following documentation:

- *IBM z/OS Management Facility User's Guide,* SA38-0652
- *Program Directory for z/OS Management Facility,* GI11-2886

  This book describes the material and procedures for installing z/OSMF and for applying the requisite service for this product.
- *IBM WebSphere Application Server OEM Edition for z/OS Configuration Guide,* GA32-0631.

  This book provides configuration information for setting up IBM WebSphere Application Server OEM Edition for z/OS, which is part of the configuration process for z/OSMF.

23.2.6 z/OSMF security

Establishing a secure environment for z/OSMF begins with your installation's security administrator. This person is responsible for ensuring that the security policies for users and resources on the z/OS system are followed with z/OSMF. Your installation's security administrator must grant the proper security product access to users before they can use z/OSMF to work with the z/OS system.

Using z/OSMF requires sufficient authority in both z/OS and z/OSMF, as follows:

- On the z/OS system to be managed, the resources to be accessed on behalf of z/OSMF users (data sets, operator commands, and so on) are secured through the security management product at your installation, for example, RACF.
- In z/OSMF, access to tasks is secured through the management of user roles.

If a z/OSMF user attempts to access a resource without sufficient authority on the z/OS host system, the user's request is rejected with an error message (insufficient authorization).

23.2.7 z/OSMF considerations for multiple instances on the same sysplex

You can have only one active instance of z/OSMF in a sysplex at any time. In certain situations, however, you might choose to create more instances of z/OSMF on the same system or other systems in your sysplex. You might do this, for example, to have a backup instance of z/OSMF available for failover (or takeover) or for testing purposes.

Observe the following restrictions for using additional instances of z/OSMF in your environment:

- You can have only one instance of z/OSMF per instance of IBM WebSphere Application Server OEM Edition for z/OS.
- The z/OSMF data file system can be used by only a single instance of z/OSMF in a sysplex at a time. To prevent the same z/OSMF data file system from being accessed by more than one z/OSMF at a time, z/OSMF locks the data file system through a global resource serialization ENQ with QNAME ZOSMF.

  If you start a second instance of z/OSMF using the same data file system, that z/OSMF will become active, but will not be usable. Users who attempt to access the second instance of z/OSMF will encounter a z/OSMF error Web page. Furthermore, all log messages from the second instance of z/OSMF are routed to the WebSphere log, rather than to the z/OSMF log in the z/OSMF data file system.
- Do not run multiple instances of z/OSMF simultaneously in a sysplex, even if using separate z/OSMF data file systems.

  Consider, for example, that the Incident Log task is sysplex-wide in scope; it manages dumps in the sysplex dump directory. If users attempt to access the Incident Log from separate instances of z/OSMF at the same time, significant delays and resource contentions can result.
- Do not list the QNAME ZOSMF ENQ in the resource name list (RNL) in your installation's GRSRNLxx member.

23.3 Using z/OSMF

z/OSMF provides a Web-based graphical user interface (GUI) that enables you to easily complete system management tasks. The user interface consists primarily of tables and property sheets. You can tailor the values shown with functions such as filtering and sorting, refreshing the data displayed with the latest information, and setting preferences to customize your user experience.

To make it easier for you to perform system management tasks, z/OSMF provides a modern, easy-to-use graphical user interface, as explained in the following sections.

z/OSMF panels

z/OSMF provides a visual framework surrounding a work area where various panels are displayed. This framework provide the basis for a common look and feel and serves as the launch point for the tasks that users need to perform.

The z/OSMF interface, as shown in Figure 23-4 on page 483, has various areas such as:

- Banner area
- Taskbar
- Navigation area
- Work area

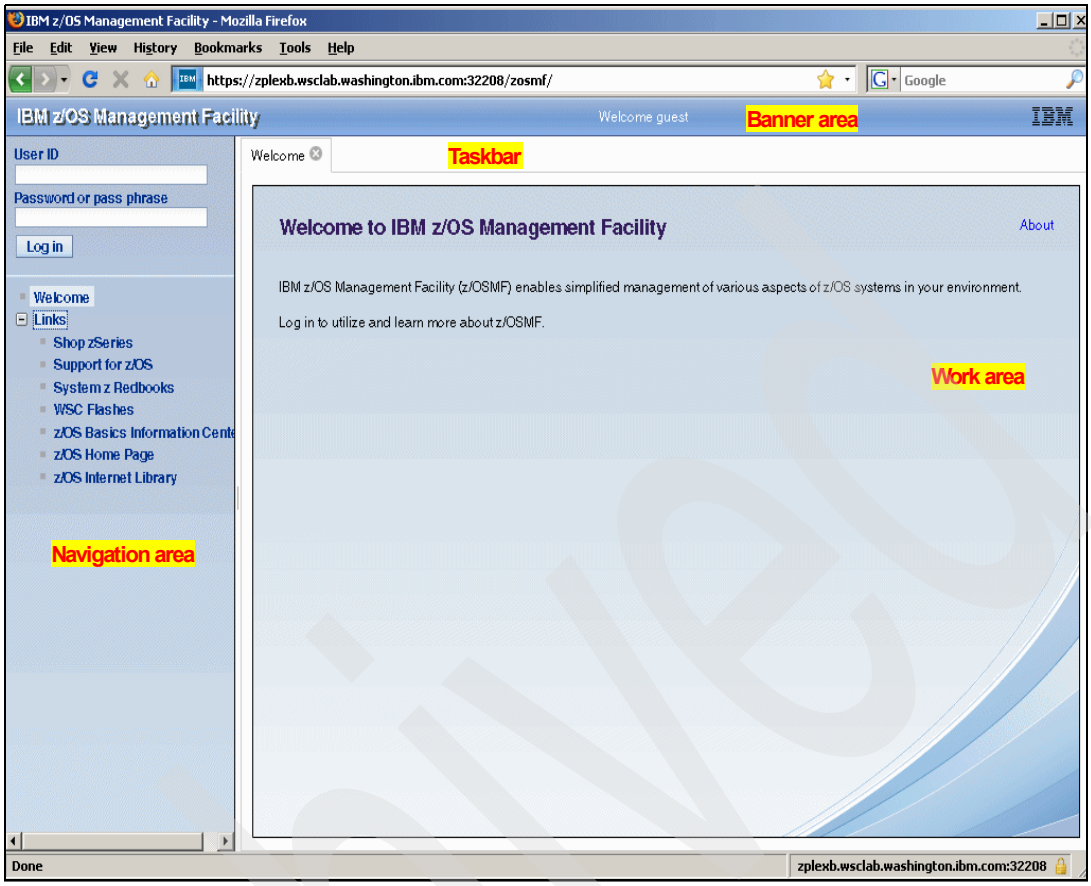


Figure 23-4 z/OSMF interface layout

z/OSMF interface layout

As shown in Figure 23-4, the Banner area has the title for z/OSMF, a welcome message, a Log out link (if you are logged in), and the IBM logo are displayed in the banner area. Note that the banner area is not resizable.

The Taskbar has one tab for each active task. To close a tab, click the X that is displayed next to the tab title.

The Navigation area is divided into two sections separated by a horizontal line:

- The log in section (top section) is displayed only when you are not logged into z/OSMF. It contains the user ID and password or passphrase fields that you can use to log in.
- The task section (bottom section) is always displayed. This area contains the tasks that you are authorized to access in z/OSMF. Most tasks are grouped by category. To display the tasks for a particular category, expand the category.

Note: To change the size of the navigation area, click and drag the divider left (to decrease the size) or right (to increase the size).

The Work area has the content (including tables, wizards, and property sheets) that you can browse or take an action against. When clicked, the Help link (located in the top right corner of the work area) opens a new browser window and displays help information about the panel.

23.3.1 z/OSMF welcome window

After logging into z/OSMF using a valid RACF user ID and password or passphrase, you can access the tasks you are allowed to perform; see Figure 23-5 on page 484.

You can launch multiple instances of z/OSMF using separate computers or separate browsers, or using multiple instances of the same browser. If you use multiple instances of the same browser (new window or tab) and your browser is configured to use the same browser session for each instance, when you log into or out of one z/OSMF instance, you are automatically logged into or out of each instance.

If you launched multiple z/OSMF instances using separate computers or separate browsers, or using multiple instances of a browser that is not configured to use the same browser session, you must log into and out of each z/OSMF instance.

Your z/OSMF session will expire after a period of time has elapsed. By default, this period is eight hours (480 minutes) from the time you log into z/OSMF. Your installation can choose to modify this setting during the configuration of z/OSMF.

If your z/OSMF session expires, you can reauthenticate using the reauthentication dialog box.

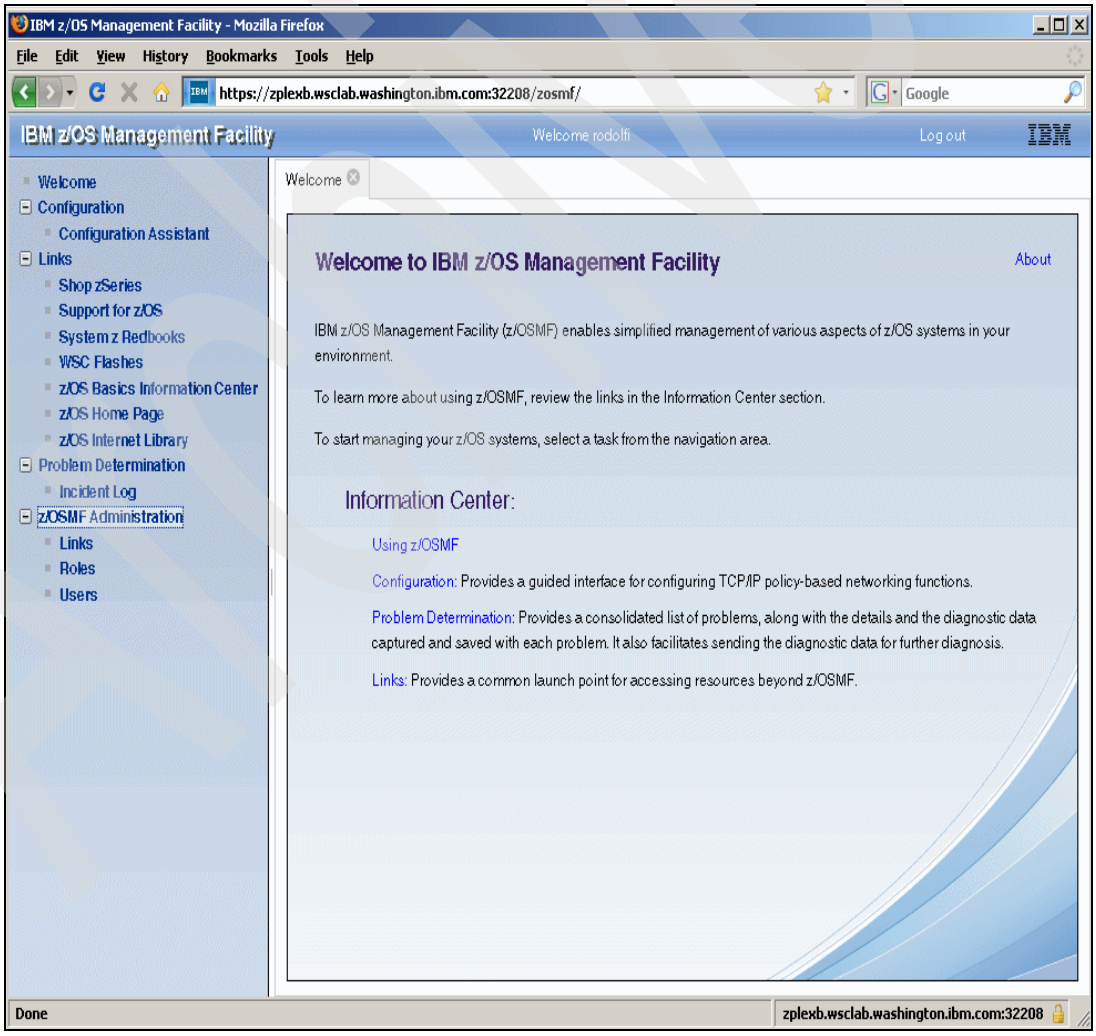


Figure 23-5 z/OSMF welcome window

From the navigation area you can select the task you want to perform. In this example, the user has an administrator role and therefore has all tasks available.

23.3.2 z/OSMF configuration options

In its first release, z/OSMF had only one option in the configuration section: the Configuration Assistance for the Policy agent in the z/OS Communication Server stack. This task is integrated now in z/OSMF and still can be performed using the IBM Configuration Assistant for z/OS Communication Server Java client. The main difference here is that you do not have to install anything in the client machine.

Clicking the Configuration Assistant option in the navigation area gives access to it, as shown in Figure 23-6 on page 485.

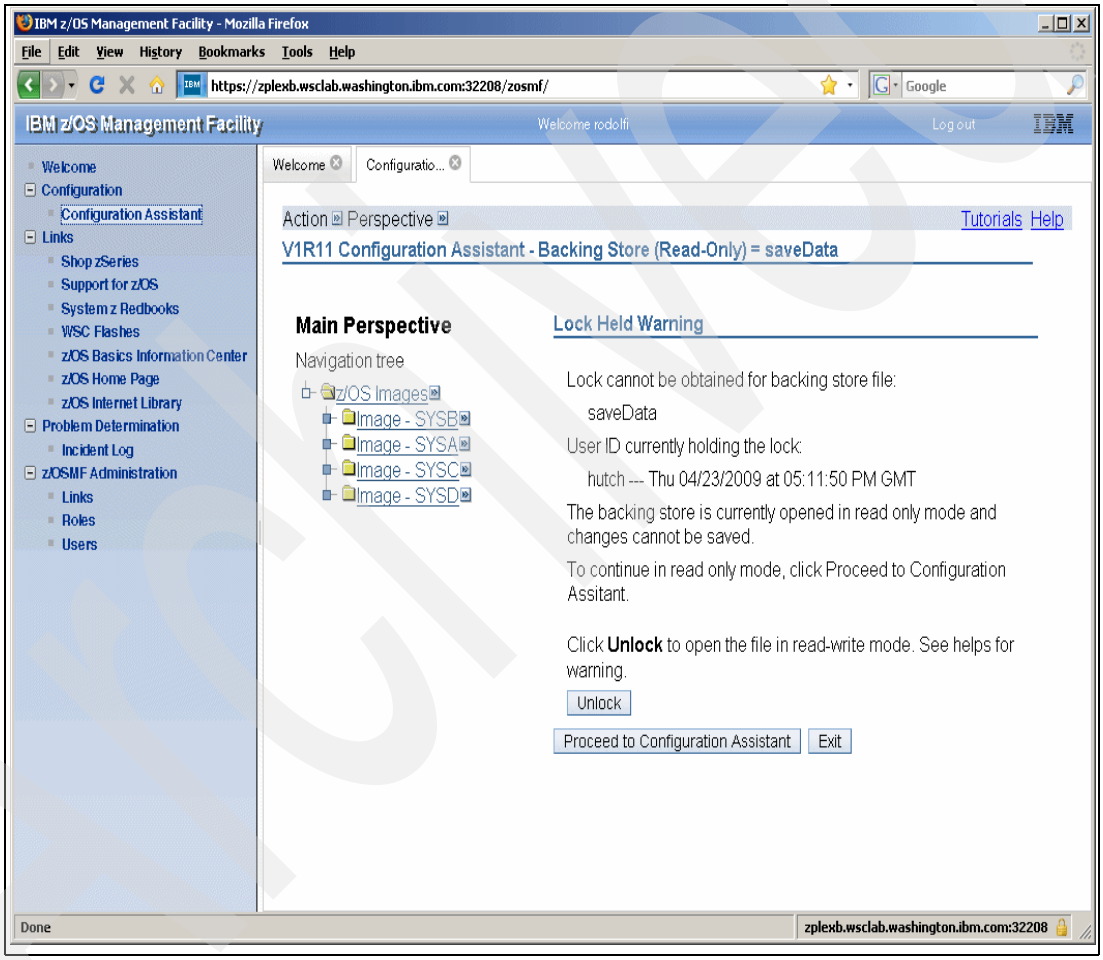


Figure 23-6 z/OSMF Configuration Assistant

If the IBM Configuration Assistant for z/OS Communication Server is being used and you already have policies defined, you have to transfer those policies to z/OSMF before you begin to use it. No change is necessary on the files.

All current policies supported by IBM Configuration Assistant for z/OS Communication Server are supported by z/OSMF, as shown in Figure 23-7 on page 486.

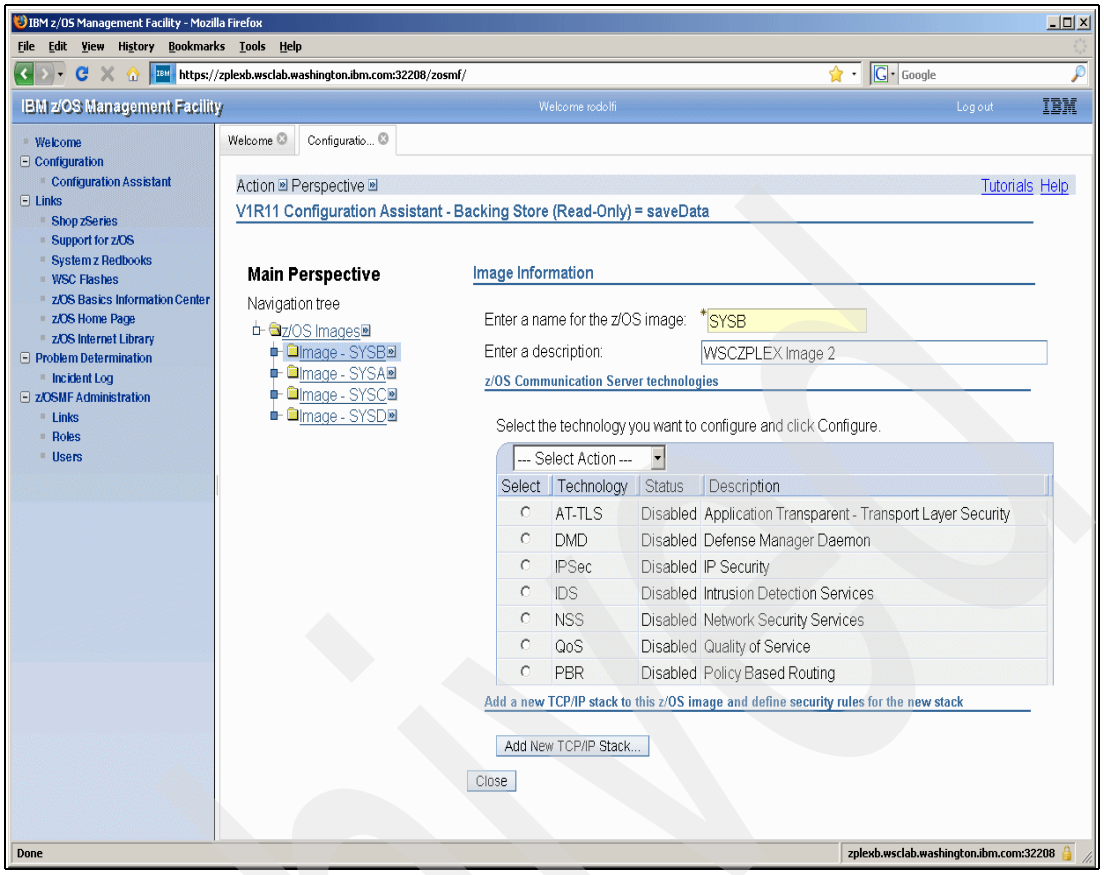


Figure 23-7 z/OSMF Configuration Assistant options

23.3.3 z/OSMF incident log

The Incident Log is a task in z/OSMF that simplifies the management of SVC dumps that have occurred on a system or in a sysplex. The Incident Log offers a browser-based user interface for viewing and managing system detected and user-initiated incidents and their associated diagnostic data.

Using this interface reduces the possibility of errors while obtaining, aggregating, and sending the collection of diagnostic data to IBM or an Independent Software Vendor (ISV). The Incident Log task is available for systems running z/OS V1R10 or z/OS V1R11.

An *incident* is an occurrence that is not part of the standard operation of a service. It can cause a disruption to, or a reduction in, the quality of service and productivity. Incidents are identified by the occurrence of a supervisor call (SVC) dump. SVC dumps that an authorized program or an operator initiates (using **DUMP** or **SLIP** commands) are related to user-initiated incidents. SVC dumps initiated by the system on behalf of abend recovery result in ABEND incidents.

When an incident occurs, the system creates diagnostic log snapshots of the operations log and error and error log summary, based on the settings specified in your installation's CEAPRMxx parmlib member.

The key functions available in the Incident Log task in z/OSMF V1R11 include:

- **Display list of incidents:** The Incident Log provides a consolidated list of problems along with the details and the diagnostic data captured and saved with each problem. By default,

only incidents that have occurred in the past three days are displayed. A maximum of 500 incidents are included in the log.

- **Set properties:** The Incident Log provides the ability to correlate an incident with a problem number or tracking ID. The problem number allows you to associate an incident with an IBM problem number (such as a PMR or an ETR number) or an ISV problem number. With the tracking ID, you can associate an incident with a problem record in your installation's problem management system.
- **Display properties**: The Incident Log allows you to view additional information about an incident, such as the symptom string, reason code, or list of collected diagnostic data. A separate tab shows the diagnostic data set associated with the incident (the SVC dump and diagnostic snapshot files). An incident can represent a multi-system SVC dump, which consists of a primary dump and multiple secondary dumps taken on other systems in the sysplex. In such cases, the Incident Log shows all of the SVC dumps associated with the incident, with a single set of diagnostic snapshot files.
- **Send diagnostic data:** The Incident Log task provides a Send Diagnostic Data wizard that you can use to send diagnostic data to IBM or another FTP destination. Here, z/OSMF gathers the diagnostic data, compresses it, and uses FTP to send it to an FTP destination. The FTP destination can be any destination you choose including the IBM Support Center or an ISV.
- **Manage FTP job status:** The Incident Log allows you to view or delete the status of an FTP job or to cancel an FTP job.
- **Allow next dump:** The Incident Log provides the ability to easily update the DAE data set, so that you can capture the next instance of an SVC dump that is being suppressed by DAE (with the same MVS symptom string).
- **Delete incident:** The Incident Log allows you to simultaneously delete an incident, all of its FTP job status information, and all associated diagnostic data (such as the operations log, error log, and SVC dumps).
- **Manage FTP destinations and profiles:** The Incident Log provides a shared location where you can specify the destinations to which you want to send diagnostic data. The Incident Log also provides a shared location where you can specify the settings needed to send diagnostic data across your firewall or proxy.

Figure 23-8 on page 488 shows an example of an Incident Log panel.

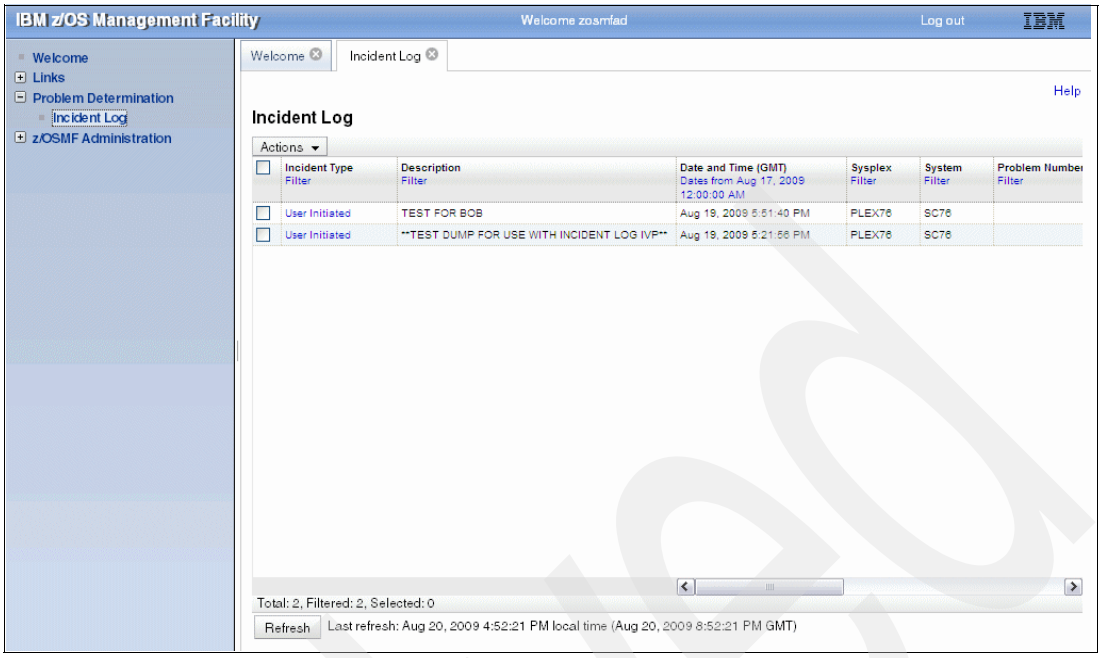


Figure 23-8 z/OSMF Incident Log

Figure 23-9 on page 489 shows an example of an Incident Log actions list.

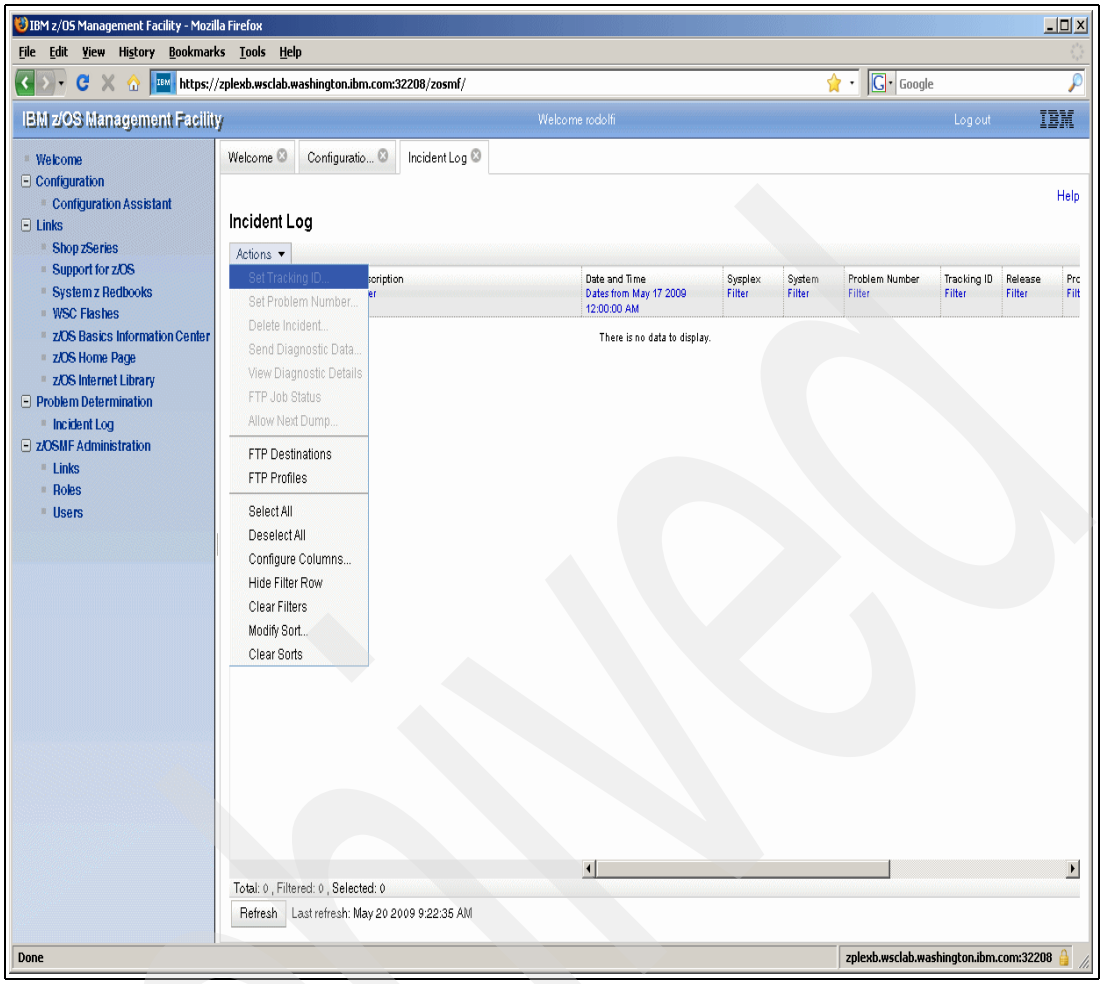


Figure 23-9 z/OSMF incident log actions list

23.3.4 z/OSMF links

Your installation can customize z/OSMF with links to external sites for system management tools and information.

The process of defining a link to z/OSMF includes specifying the link and its location (a URL) and selecting which z/OSMF roles can access the link. Depending on the user's Web browser settings, the link opens as either a new browser window or a new browser tab.

To define links for z/OSMF, your user ID must be assigned to a z/OSMF role that is permitted to define links. By default, only the z/OSMF Administrator role can define links.

To display the Links panel, expand the z/OSMF Administration category in the navigation area and select Links.This will begin a sequence of steps for defining links for z/OSMF; see Figure 23-10 on page 490.

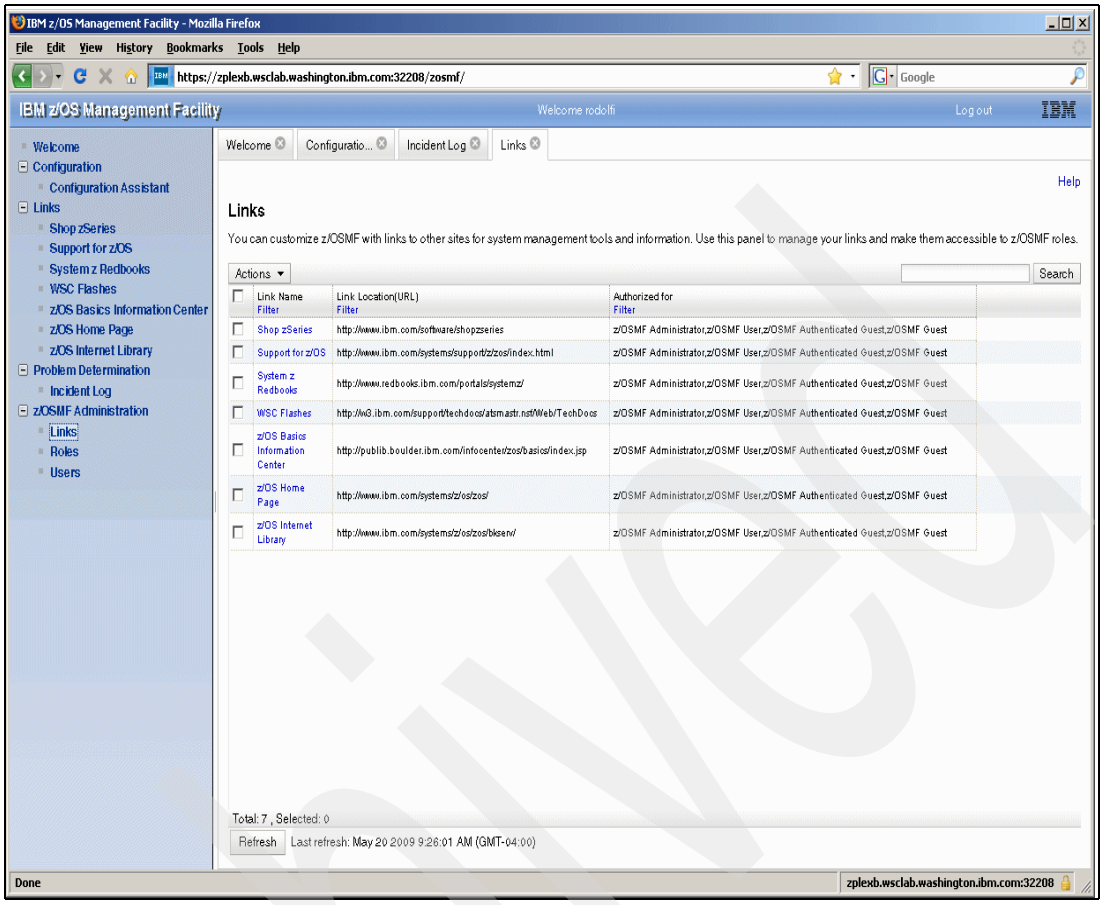


Figure 23-10 z/OSMF links

23.3.5 z/OSMF roles

In z/OSMF, a *role* represents the ability to perform one or more tasks. These tasks include both general administrative actions for z/OSMF and task-specific actions for the z/OS system to be managed. Your installation can modify the tasks for a role through the Roles task.

The process of defining a role in z/OSMF involves the separate actions of selecting tasks for the role (through the Roles task) and assigning users to the role (through the Users task).

z/OSMF filters the list of tasks shown in the navigation area to match the authorization of the assigned role for the current user. If a role is not authorized to work with certain tasks, those tasks are not displayed in the navigation area for users assigned to the role.

To work with roles for z/OSMF, your user ID must be assigned to a z/OSMF role that is permitted to the Roles task. By default, only the z/OSMF Administrator role can work with roles.

When defining roles in z/OSMF through the z/OSMF administration tasks, the z/OSMF administrator must ensure that a user's authority in z/OSMF is sufficient within the user's authority on the z/OS host system. If not, the user's authorization on the z/OS system takes precedence over the user's authorization in z/OSMF.

Be aware that z/OSMF does not prevent you from modifying the role or the user definition with which you are currently logged on to z/OSMF. For example, if you remove the Roles task from the role definition to which your user ID is assigned, you lose the authority to continue working

with the Roles task. z/OSMF performs the change and issues messages to confirm the modification of the role definition and indicate that you are not authorized to use the Roles task.

To work with roles in z/OSMF, expand the z/OSMF Administration category in the navigation area and select **Roles**. This will begin a sequence of steps for defining which tasks are assigned to a z/OSMF role; see Figure 23-11.

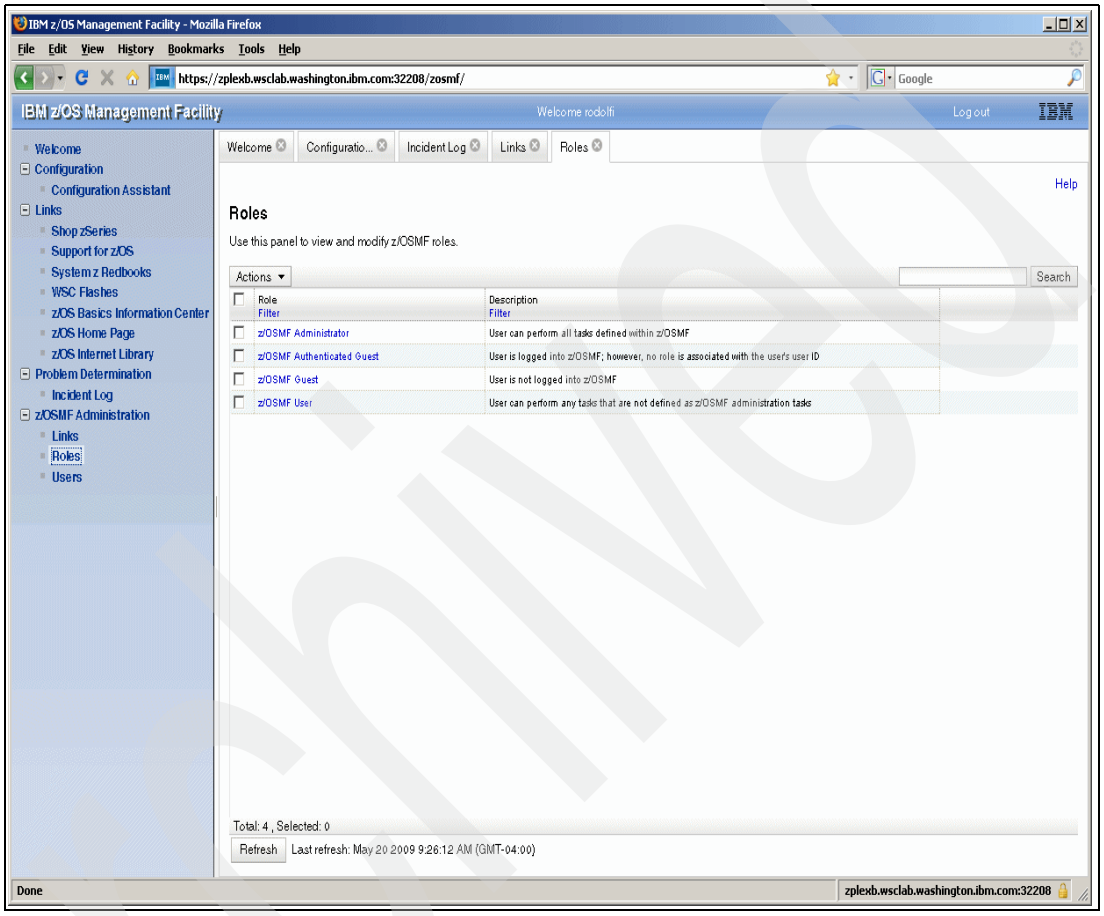


Figure 23-11 z/OSMF roles

23.3.6 z/OSMF users

To perform work in z/OSMF, a user must be defined to z/OSMF. With the appropriate authorization established in both z/OSMF and on the z/OS system to be managed, a z/OSMF user can select one or more tasks from the navigation area and advance through a guided sequence of panels for performing each task.

On the Users panel, you define a user by specifying the user ID and user name. You also select a role for the user: z/OSMF User or z/OSMF Administrator. By specifying a role for the user, you authorize this person to perform the tasks that are associated with that role.

To define users for z/OSMF, your user ID must be assigned to a z/OSMF role that is permitted to the Users task. By default, only the z/OSMF Administrator role can define users.

For any new users to be added, verify that your installation's security management product (for example, RACF) will permit the user to access the system resources needed to perform a

particular task. If necessary, contact your installation's security administrator to authorize the user.

For help with establishing the required security setup for the user, your installation's security administrator can refer to the configuration scripts that are provided with z/OSMF. The scripts contain sample RACF commands for authorizing users and groups to system resources.

The z/OSMF configuration process creates an initial administrator user ID. With this user ID, the administrator can log into z/OSMF and add more users and administrators as needed, through the Users panel.

To work with user definitions in z/OSMF, click the z/OSMF Administration category in the navigation area and select **Users**. This will begin a sequence of steps for defining one or more users to z/OSMF; see Figure 23-12.

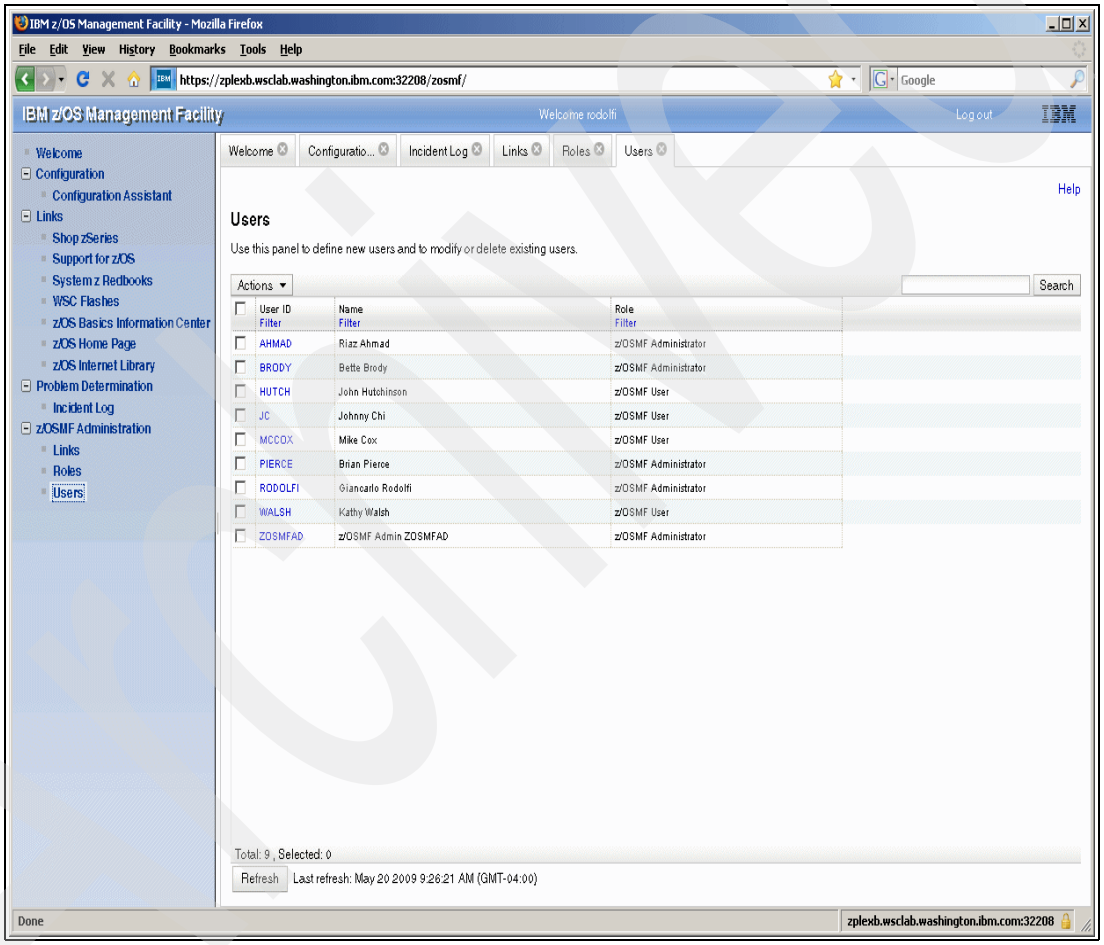


Figure 23-12 z/OSMF users

24

System REXX enhancements

System REXX is a z/OS component that allows REXX execs to be executed outside of conventional TSO/E and batch environments. REXX has long been considered one of the fastest development languages for system exit and utilities work on z/OS. The possibilities for exploiting REXX code through the use of System REXX are vast, whether to provide operator assists or to provide an easy way to process files and strings. The System REXX environment provides a function package that allows a REXX exec to invoke system commands and to return results back to the invoker in a variety of ways. System REXX execs may be initiated through an assembler macro interface called AXREXX or through an operator command.

The command interface in particular is very useful for running REXX directly from a console and providing operators with more specific information derived from multiple sources. This not only helps to pinpoint problems or potential problems faster, but can also create new trigger points to engage automation products.

This chapter describes the enhancements to System REXX as provided by z/OS V1R11:

- New REXXLIB statement in parmlib
- New SYSREXX REXXLIB command
- JES affinity support
- Using the AXREXX macro within an exit
  - SMF dump exit (IEFUJV)
    - Running IFASMFDP within the exec
- Implementing REXX health checks

24.1 System REXX overview

System REXX, SYSREXX, is a BCP component that can help you simplify and automate day-to-day operational tasks. It can also be used within an assembler program to call REXX execs. There are two interfaces, the **MODIFY AXR** command and an assembler macro AXREXX.

System REXX starts automatically during master scheduler initialization and is to run in the SYSSTC service class. There is a SYS1.PARMLIB member (CTIAXR00) for controlling Component Trace (errors are traced by default) and also a SYS1.SAMPLIB member (AXR00) that may be tailored and copied into SYS1.PARMLIB to override IBM-supplied defaults.

The command interface in particular is useful for running REXX directly from a console and providing operators with more specific information derived from multiple sources. This not only helps to pinpoint problems or potential problems faster, but can also create new trigger points to engage automation products.

24.1.1 System REXX with z/OS V1R9

The introduction of System REXX in z/OS is due to a requirement to provide an infrastructure to support Web-based interactions with z/OS components as part of the “New Face of z/OS” initiative for simplifying z/OS management. The cornerstone of this new infrastructure is SYSREXX, which allows execs to be run simply and independently from traditional TSO/E and batch environments. This implementation has two interfaces:

- A single program interface (AXREXX)
- Operator exploitation directly from a console

The expected benefits of this implementation include:

- Enabling rapid development and deployment of system programmer tools and operator assists
- Exploitation by new and old style applications
- Allowing health checks to be written in REXX

As a beginning in z/OS V1R9, SYSREXX is the required environment for CIM and Health Checker REXX execs to be written.

24.2 SYSREXX address space

The SYSREXX address space (AXR) is a subsystem that is started during master subsystem initialization. It reads the AXR00 parmlib member and allocates the REXXIN data set. When REXX work arrives through a PC directly into the appropriate server, SYSREXX or console-initiated REXX execs are detected by the AXR SSI listener, converted to **F AXR** command format, and queued to the command server’s CIB control block. They, in turn, are then selected and scheduled for processing.

Removing the AXR address space

The System REXX address space, AXR, is non-cancelable, but can be terminated by invoking the following command:

```
FORCE AXR,ARM
```

When the AXR address space terminates, an ENF signal 65 with a qualifier of 40000000x is issued. AXR can be restarted by starting the AXRPSTRT procedure, which can be found in SYS1.PROCLIB. When the AXR address space initializes, an ENF signal of 80000000x is issued. To restart the AXR address space, issue:

```
S AXRPSTRT
```

24.2.1 SYSREXX from consoles

When REXX work requests originate from an operator console, that is detected by an AXR SSI listener function (shown in Figure 24-1), or a program interface called using the AXREXX macro service (shown in Figure 24-1).

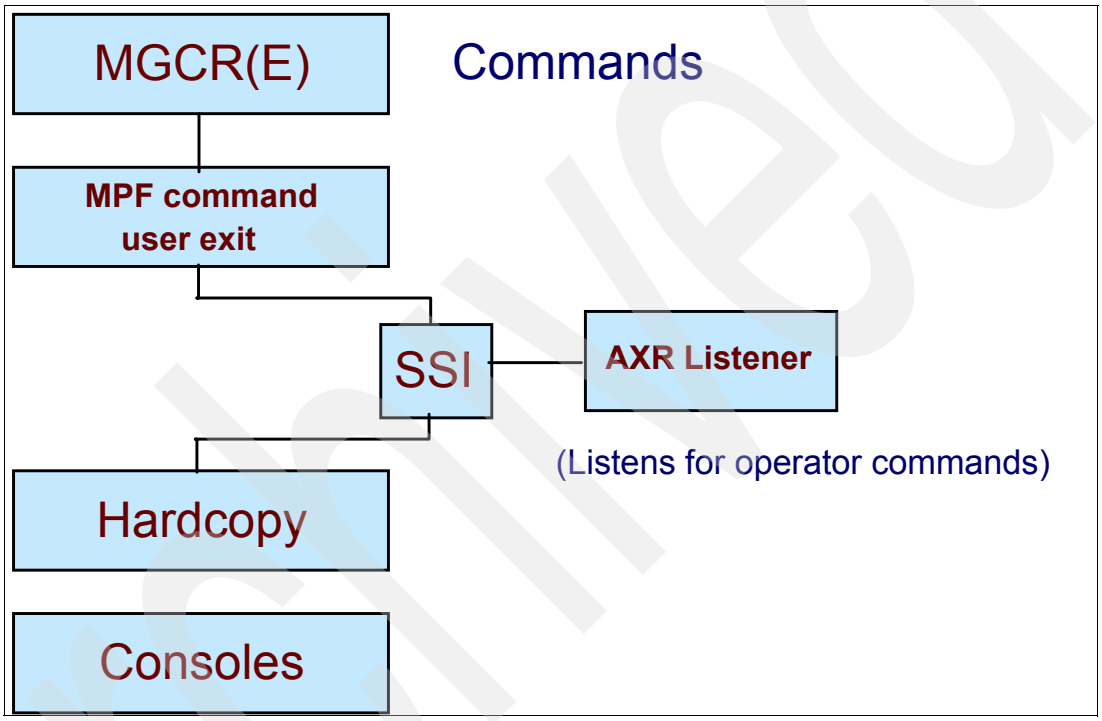


Figure 24-1 AXR listener function that intercepts commands for SYSREXX

24.2.2 AXREXX macro service

AXREXX provides a macro interface for System REXX services. Before issuing the AXREXX macro service, the caller does not have to place any information into any general purpose register (GPR) or access register (AR) unless the caller is using the input register in register notation for a particular parameter, or using it as a base register.

Note: See *z/OS MVS Programming: MVS Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*, SA22-7609, for a complete description of AXREXX macro parameters.

The AXREXX invoker can limit the amount of time that an exec can run by using the TIMELIMIT/TIMEINT keywords. When the time limit is reached, System REXX invokes HALT interpretation on the REXX environment where the exec is running. If the exec still does not complete after waiting for a period of time, then the task running the exec is detached. Invokers who specify a time limit need to realize that time out is an error condition and that for

SYNC=YES invokers, the final values of output arguments and variables will not be returned to the AXREXX invoker.

AXREXX supports an interface to cancel an exec. SYNC=NO AXREXX invokers can obtain the Request Token through the OREQTOKEN parameter for later input to the AXREXX **`CANCEL`** command. Cancel is processed as though the exec timed out.

Using the AXREXX macro service

The System REXX environment provides a functional package that allows a REXX exec to invoke system commands and to return the results back to the invoker in a variety of ways. When System REXX execs are initiated through an assembler macro interface called AXREXX or through an operator command, there are two execution environments that are supported:

TSO=NO When TSO=NO is specified on the AXREXX invocation, the exec is executed in an MVS host command environment, sharing the address space where it is executing with up to 63 other concurrently running TSO=NO execs. Data set allocation, other than provided by the AXREXX macro, is not supported in the TSO=NO environment.

TSO=YES The TSO=YES environment supports all of the host commands that TSO=NO supports, along with several of the host commands supported by TSO/E. If TSO=YES is specified on the AXREXX invocation, the exec will run isolated in a single address space, and can safely allocate data sets without concern of a DDNAME conflict with a concurrently running exec. If the exec were to exit with data sets allocated, then System REXX will free the allocations. The TSO environment is established by the dynamic TSO service (IKJTSOEV) and does not support all of the TSO functionality. Running under the MASTER subsystem further restricts what TSO host commands will work.

Applications that perform input and output to data sets other than those specified on the REXXINDSN and REXXOUTDSN AXREXX keywords are to use TSO=YES.

TSO environment

The TSO environment is established by the dynamic TSO service (IKJTSOEV) and does not support all of the TSO functionality. If an exec is initiated when the primary subsystem is not active, the exec runs under the MASTER subsystem, thus further restricting which TSO host commands will work. Only the TSO=YES environment supports SYSCALL (z/OS UNIX) host commands that are only available to requests associated with RACF user IDs that have OMVS segments defined to them, which establishes the level of z/OS UNIX authorization. In particular, if execs are initiated from the operator console, the operator must be logged on for many SYSCALL host commands to work.

As shown in Figure 24-2 on page 497, the REXX server controls a group of worker subtasks that attach daughter subtasks to process TSO=NO requests. Initially four are started, but up to 64 are started as required. The TSO server controls a group of worker subtasks that start between 1 to 8 address spaces to process TSO=YES requests.

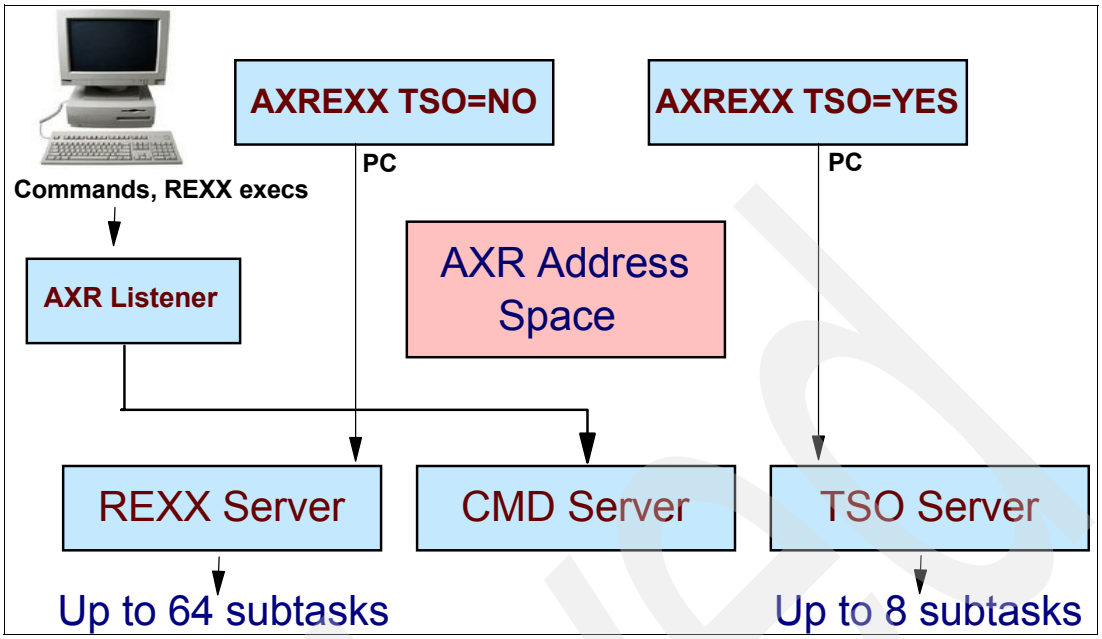


Figure 24-2 System REXX overview of REXX EXEC processing

24.3 Customizing System REXX

The customization necessary to use System REXX involves the AXRxx parmlib member and the user ID to be associated with System REXX and CPF command prefix for issuing console commands.

AXR00 parmlib member

During AXR address space initialization, the AXRxx parmlib member is read. A sample member is supplied in SYS1.SAMPLIB with member AXR00, as shown in Figure 24-3.

CPF defines a 1- to 8-character command prefix value for System REXX that can be used instead of specifying the **F AXR** command. This prefix may be defined as either SYSTEM or SYSPLEX in scope, as explained here:

SYSTEM System scope indicates that the CPF will only be recognized on the system on which it is defined.

SYSPLEX Sysplex scope indicates that it will be recognized throughout the SYSPLEX, and the command will be routed to the system on which it is defined.

Using the defaults can result in a very unusual prefix. For example, if the SYSCLONE value is 63, then the CPF prefix is REXX63.

```
CPF('REXX&SYSCLONE.',SYSPLEX) /* Defines REXXnn as a sysplex
                              wide cpf value */
AXRUSER(AXRUSER)        /* AXREXX security=axruser results in the
                           exec running in a security environment
                           defined by the userid AXRUSER          */
```

Figure 24-3 SYS1.SAMPLIB member AXR00

The following sample shows parameter setting in the AXR00 member in parmlib to set the CPF character to @ and having a sysplex scope:

```
CPF('@',SYSPLEX) /* DEFINES @AS A SYSPLEX WIDE CPF VALUE */
```

AXRUSER specification and security

The AXRUSER parameter in the AXRxx parmlib member specifies a 1- to 8-character user ID that is used to define the security environment that an exec initiated using the AXREXX macro when SECURITY=BYAXRUSER is specified. The exec will run with the level of authorization associated with the specified user ID. The installation needs to provide the user ID with SAF access to the resource SYSREXX.<userid>. If this parameter is omitted, then the default is AXRUSER.

The following security considerations are necessary for a SYSREXX environment.

- AXREXX is an authorized system service. Therefore, security controls are essential for:
  - Access to APF library
  - Permissions and standard security administration to determine what can be accessed or run
- EXECs by default use the invoker's security environment, but alternatively access can be given as follows:
  - Access authority of a third party
  - A special user ID assigned to AXRUSER
- EXECs use the invoker's enclave service class, so this can do the following:
  - Prevent CPU priority inversion and excessive resource usage
  - Allow resource usage to be charged back to the enclave service class

The SECURITY and UTOKEN parameters on the AXREXX macro service determine the security environment that the exec runs in. If omitted, the exec will run under the same security environment as its invoker. The security environment determines the data sets that may be accessed and the commands and programs that may be invoked.

- When SECURITY=BYUTOKEN is specified, the invoker can provide a UTOKEN to define the specific security environment that the exec is to run under (see *z/OS Security Server RACROUTE Macro Reference*, SA22-7692).
- When SECURITY=BYAXRUSER is specified, the exec will run under the security environment associated with the value (user ID) of the AXRUSER parameter specified in the AXR00 parmlib member. This can be useful if the installation wants to invoke AXREXX in an address space that does not have a security environment, such as the MASTER address space. The exec are not, however, to invoke any services that alter the security environment of the task running the exec.

In all cases, the REXX exec runs under a WLM enclave of the AXREXX invoker.

24.4 Design considerations

Introduced in z/OS V1R9, SYSREXX has triggered the imagination of many and driven, in consequence, new requirements such as:

- Introduced in z/OS V1R9, SYSREXX execs can only be processed from one PDS named SYS1.SAXREXEC. The SYS1.SAXREXEC data set contains execs that IBM has provided. These execs are *never* to be deleted or modified in any way. This data set is

where user-written execs are to be added. These user-written execs are *not* to start with the letters A through I, because those letters are reserved for use by IBM.

Important: The SYS1.SAXREXEC data set is unique in the system. Therefore, take regular backups of this SYS1.SAXREXEC data set to ensure the following outcomes:

- User-written execs can be easily restored in case of a restore of the resident volume.
- IBM execs can be easily restored in case of an human error while editing other members of this PDS.

The use of the SYS1.SAXREXEC data set is changed in z/OS V1R11; see 24.5.1, “REXXLIB concatenation” on page 499.

- Read-only storage, which is an important REXX capability, is currently disabled due to various TSO/E changes required.
- Currently, there is no JES affinity because SYSREXX starts under the master subsystem. For this reason, it cannot use spool-related functions such as INTRDR to submit batch.
- z/OS UNIX Syscall is not accessible. For this reason, a SYSREXX exec cannot take advantage of the most useful UNIX System Services functions.
- SYSREXX initializes using SYS1.PARMLIB(AXR00) and cannot use a separate member or concatenate members because it is dependent on an IPL NIP component change.
- New built-in functions are introduced, such as a wait function.

24.5 z/OS V1R11 System REXX enhancements

The following enhancements have been made to System REXX with z/OS V1R11:

- New concatenation of REXXLIB statements in parmlib
- New SYSREXX REXXLIB command
- JES affinity support
- Using the AXREXX macro within an exit
  - SMF Dump Exit (IEFUJV)
    - Run IFASMFDP within the exec
- Implementing REXX health checks

These enhancements are explained in more detail in the following sections.

24.5.1 REXXLIB concatenation

Introduced in z/OS V1R9, SYSREXX execs can only be processed from one PDS named SYS1.SAXREXEC. Consequently, client REXX execs must first be copied into a system data set to be processed. This caused a management problem that can become an issue if many people developed SYSREXX execs.

SYS1.PARMLIB definitions

With z/OS V1R11, you may now specify individual, user-specified AXRnn parmlib members instead of just AXR00. A new specification is introduced in the IEASYSxx parmlib member

through the AXR=keyword, specifying a list of parmlib members. This support allows statements such as shown in Figure 24-4. This provides more flexibility because this parameter does not need to be AXR00, which is the default value.

Additionally, the IEASYSnn parmlib member supports a concatenation of AXRnn members. The 2-character identifier (aa, bb, and so forth) is appended to AXR to identify the AXRxx member of SYS1.PARMLIB.

```
AXR=aa
AXR=(aa,bb..)
```

Figure 24-4 IEASYSxx parmlib member AXR support

Note: If AXR= is not provided in IEASYSxx, then AXR00 is used. The IBM-supplied default for AXRxx is AXR00, and the AXRxx member is optional.

If no alternative AXRnn parmlib member has been configured and AXR00 is not found, then REXX&SYSCLONE is the default value assigned to CPF, and SYS1.SAXREXEC is the default value assigned to REXXLIB.

REXXLIB statement

With z/OS V1R11, a new REXXLIB statement is introduced in the AXR00 parmlib member as shown in Figure 24-5. This introduces the ability to concatenate REXXLIB statements, to address the management problem related to the SYS1.SAXREXEC data set.

```
REXXLIB ADD DSN(<data set name>)
REXXLIB ADD DSN(<data set name>) VOL(<volser>)
```

Figure 24-5 New REXXLIB statement

The REXXLIB concatenation is an ordered list (by appearance in AXRxx) of data sets specified by REXXLIB ADD. When the AXREXX macro is called directly or indirectly from the operator console, System REXX searches the concatenated list, shown in Figure 24-8 on page 501, for the specified exec. If the cataloged SYS1.SAXREXEC is not among the data sets specified, it is appended to the end.

REXXLIB data sets

All data sets specified in the REXXLIB concatenation must have the same record type and record length as SYS1.SAXREXEC. The data set must either be a PDS or PDSE.

This statement has the following limits:

- It may specify up to 255 data sets.

  The number of data sets that you can add to form the REXXLIB concatenation is limited by the total number of DASD extents the data sets will occupy. A partitioned data set extended (PDSE) counts as one extent. The total number of extents must not exceed 255, which is the current system limit.

- It has a maximum concatenation of 255 extents.

  The system will concatenate as many of the REXXLIB data sets as possible until the system limit of 255 extents is reached. If each data set in the REXXLIB concatenation occupies just a single extent, then 255 data sets can be concatenated. The effective size of the concatenation is reduced if data sets in the concatenation occupy secondary

extents. Each secondary extent reduces the concatenation size by one. The total reduction is the sum of all secondary extents. When the system limit is exceeded, message AXR0115E is issued, and no further requests are processed by System REXX.

- The data sets must have the same attributes (RECFM and LRECL) as SYS1.SAXREXEC, and it can be a PDS or a PDSE.

  Concatenation is populated in the order the statements occur, and REXXLIB statements may be in multiple AXRnn parmlib members. The cataloged SYS1.SAXREXEC is included at the end of the concatenation if not specified in the concatenation, with the 255th data set being supplanted if necessary.

REXXLIB data set statement examples

In that context, system symbols are supported which can lead to the syntax shown in Figure 24-6.

```
CPF('@',SYSTEM)
AXRUSER(AXRUSER)
REXXLIB ADD DSN(WTSCPLX2.&SYSNAME..REXXEXEC)
REXXLIB ADD DSN(WTSCPLX2.REXXEXEC)
REXXLIB ADD DSN(SYS1.SAXREXEC) VOL(&SYSR1.)
```

Figure 24-6 REXXLIB statements with system symbols

24.6 New REXXLIB commands

To provide proper support for REXXLIB concatenation, a new SYSREXX REXXLIB command known as the **F AXR,SR R** command is introduced in z/OS V1R11; see Figure 24-7.

```
F AXR,SYSREXX REXXLIB
F AXR,SR R
```

Figure 24-7 New SYSREXX REXXLIB command

The **F AXR,SR R** command displays the REXXLIB concatenation as shown in Figure 24-8. You can use the **F AXR** command or you can use the prefix defined in the CPF parameter of the AXR00 parmlib member to replace **F AXR**, as shown.

```
@SR R
AXR0202I SYSREXX REXXLIB DISPLAY 136
  ENTRY VOLUME  DATA SET
    1   SBOX00  WTSCPLX2.SC65.REXXEXEC
    2   SBOX00  WTSCPLX2.REXXEXEC
    3   Z1BRB1  SYS1.SAXREXEC
```

Figure 24-8 Display the REXXLIB concatenation

SYSREXX status command

You can use the **F AXR** command to either obtain status about System REXX or to initiate the execution of a REXX exec. To obtain status information, enter the command shown in Figure 24-9.

```
F AXR,SR,STATUS
    AXR0200I SYSREXX STATUS DISPLAY 234
     SYSTEM REXX STARTED AT 19.08.56 ON 03/26/2009
     PARMLIB MEMBERS:      AXR00
     CPF:  @ (SYSPLEX)           AXRUSER:  IBMUSER
     TIMEINT:              30
     SUBSYSTEM:            AXR
     REQUESTS QUEUED:      0  ACCEPTING NEW WORK
     REXX WORKER TASKS:    ACTIVE:   0     TOTAL:   4
                           IDLE:     4     MAX:    64
                           ASYNC:    0     SYNC:    0
                           UNTIMED:  0
     TSO SERVER SPACES:    ACTIVE:   0     TOTAL:   0
                           IDLE:     0     MAX:     8
                           ASYNC:    0     SYNC:    0
                           UNTIMED:  0
```

Figure 24-9 SYSREXX STATUS display

To obtain detailed status information, enter the command shown in Figure 24-10, where:

STATUS | ST This specifies or indicates that general information about System REXX execs is to be returned to the invoker.

DETAIL | D This indicates that detailed information about execs that are currently running in System REXX is to be returned to the invoker.

```
F AXR,SR,STATUS,DETAIL
AXR0201I SYSREXX STATUS DETAIL
EXEC=WAITLOOP CJBN=AXR      CASID=0015 TSO=Y T/L=00.00.30
  REQTOKEN=00005200000000000BF3A704A6511A3B5
  EJBN=AXR02    EASID=0033 TCB=006FF098 CPU=000.004S TIME=005.739S
EXEC=INFINITE CJBN=AXR      CASID=0015 TSO=Y T/L=00.00.30
  REQTOKEN=0000540000000000BF3A704C2088405C
  EJBN=AXR03    EASID=0032 TCB=006FF098 CPU=000.006S TIME=003.925S
```

Figure 24-10 Detailed status information

AXREXX macro

Similarly, new parameters providing additional function are added to the AXREXX macro with z/OS V1R11:

REQUEST=GETREXXLIB This returns the REXXLIB concatenation. The concatenation itself cannot be dynamically modified within the scope of an IPL.

REXXLIB= This is the address of storage area where the REXXLIB concatenation details are returned. The details are mapped in AXRZARG.

REXXLIBLEN= This is the address of a field containing the length of the REXXLIB area

24.7 JES affinity

With z/OS V1R11, TSO=YES address spaces (AXR01-08) now run under the primary subsystem when available. Otherwise, they run under the MASTER scheduler address space (as in pre-V1R11 systems).

When running under the primary JES control, the following conditions are supported:

- The JES joblog is now available when running under JES.
- Additional host commands are supported, such as TSO TRANSMIT and RECEIVE.
- Although the SUBMIT host command is still not supported, an internal reader can be allocated to submit JCL, as follows:

  ```
  ALLOC FI(submit) SYSOUT WRITER(INTRDR) RECFM(F) LRECL(80) REUSE
  ```

  An example showing use of SYSREXX is shown in Figure 24-11.

Note: SYSOUT=(A,INTRDR) is not allowed for use in a dynamic TSO environment.

```
/* REXX */
dsn = jll.jcl
mem = bidon
jcl.0 = 0

"ALLOC  FI(jcntl) DA('"dsn"("mem")') SHR"
"EXECIO * DISKR jcntl (STEM jcl. FINIS"
"FREE   FI(jcntl)"

if jcl.0 /= 0 then
  do
    "ALLOC FI(submit) SYSOUT WRITER(INTRDR) RECFM(F) LRECL(80) REUSE"
    do i = 1 to jcl.0
       say 'STMT'i': 'jcl.i
end

"EXECIO * DISKW SUBMIT (STEM jcl. FINIS "
    "FREE   FI(submit)"
  end

exit rc
```

Figure 24-11 SYSREXX example of submitting to the INTRDR

24.8 z/OS UNIX SYSCALL

With z/OS V1R11, SYSCALL host commands are now supported for TSO=YES SYSREXX address spaces. More precisely, SAF identity is now at the address space level (it was previously at the execution task level). The userID function now returns the invoker's user ID

The SYSCALL commands available are determined by:

- Mode of z/OS UNIX customization (minimal versus full)

- The invoker's security authorization
  - the UID level as set in the user ID's OMVS segment
  - Any other special RACF privileges required

Note: For console-invoked execs using SYSCALL commands, an OMVS segmented user ID *must* be logged on.

Figure 24-12 shows an example of how to use this new capability.

```
SYSCALLS('ON')
   ADDRESS SYSCALL
   SYSCALLS('OFF')
```

Figure 24-12 UNIX System Services SYSCALL in SYSREXX

Starting AXR address spaces

This also provides the capability for specifying a list of parmlib members when starting the AXRPSTRT procedure.

For starting the AXR address space the new keyword AXR is allowed, with a default value of AXR00.

```
START AXRPSTRT,AXR=aa
START AXRPSTRT,AXR=(aa,bb..)
```

Figure 24-13 Starting the AXR address space

ENF 65 signal

An ENF signal is issued when the SYSREXX AXR address space starts and ends. The operator can restart AXR by using the AXRPSTRT procedure, found in SYS1.PROCLIB.

The syntax for restarting the AXR address space can be *one* of the following, where `aa` and `bb` are AXRnn parmlib members in SYS1.PARMLIB:

```
START AXRPSTRT

START AXRPSTRT,AXR=aa

START AXRPSTRT,AXR=(aa,bb,...)
```

If no parmlib members are specified, then values from AXR00 are applied (if it exists). Otherwise, default values are assigned.

24.9 SYSREXX built-in functions

When writing EXECs, SYSREXX provides specific functions.

Previously, three functions were provided:

AXRCMD Issue a console command and return command responses.

AXRCMD is used to issue a system command from within the exec and obtain one or more command responses. The arguments that can be specified are:

- Command text - the system command to be invoked

  This is an optional argument. If it is omitted, no command will be issued, but a response from the last command issuance will be returned if one exists.

- Msgstem - the stem of a list of variables into which AXRCMD places the command response message text.

  This is an optional argument. If it is omitted, the command text must be specified. To place the message text into compound variables which allows for indexing, msgstem is to end with a period (.) as in “messg.” for example.

  AXRCMD places each line of the retrieved message into successive variables. For example, if the command response is a 3-line message, then messg.1 contains line 1, messg.2 contains line 2, and messg.3 contains line 3. messg.0 will contain the number of lines.

  Note that msgstem does not end with a period; instead, the variable names are appended with consecutive numbers. For example, suppose you specify msgstem as “conmsg” (without a period). If AXRCMD retrieves a message that has two lines of message text, then AXRCMD places the text in the variables consmsg1 and consmsg2. The variable consmsg0 contains the number of lines in the message text, which is 2.

- Time - the amount of time in seconds that AXRCMD is to wait for a command response.

  This is an optional argument. If it is omitted, AXRCMD will not wait before attempting to determine whether a command response was returned. A value of 0 to 21474535 seconds may be specified.

AXRWTO Issue a single line message to a console.

AXRMLWTO Issue a multiline line message to a console.

Two new built-in functions are provided in z/OS V1R11:

AXRWAIT AXRWAIT is a function that provides the capability for a REXX exec to wait for a specified amount of time in seconds. A single parameter with a numeric value between 0 and 21474536 is required. If the input is not syntactically correct, AXRWAIT ends the exec with a syntax error; otherwise, it returns error codes.

AXRINFO AXRINFO returns information about the environment under which the exec is running. A supported parameter option is SUBSYSTEM, which returns the type of subsystem under which the REXX exec is running.

24.10 Migration and coexistence considerations

Quoted argument preprocessing has changed. It is applicable if quotes are currently used within the argument string of the MODIFY AXR,<rexxexecname> <args> command and the REXX exec expects and processes the quotes.

Arguments passed from a console-invoked exec are handled differently, as explained here.

Pre-V1R11

Arguments bounded by quotes are passed through to the REXX exec unaltered, including the quotes.

With V1R11

Bounding quotes are removed from single-quoted arguments. Two single consecutive quotes result in a single quote being passed as part of the argument. To preserve lower case argument strings, do the following:

- It is necessary to bound them in quotes.
- The bounding quotes will be removed as shown in Table 24-1. To preserve a single quote in an argument string, two consecutive quotes must occur within the command entered argument string.

Table 24-1 Quoted arguments processing

Input argument string	Resolved argument string
'www.lakeminnewaska.org'	www.lakeminnewaska.org
'Fire declared ' out	Fire declared OUT
'Minnewaska''s Scenic Beauty'	Minnewaska's Scenic Beauty
''' Shawangunk' Mountain 'Trails'	' Shawangunk MOUNTAIN Trails
'''Lake Awosting'''	'Lake Awosting'
''Ranger Guide”	RANGER GUIDE
''''Mossy Glenn Footpath''''	''Mossy Glenn Footpath''

Interactions and dependencies

There are no hardware or software dependencies.

Rollback APARs are available for z/OS V1R9 and z/OS V1R10, as follows:

- OA25908 - IPLNIP support for IEASYSxx / AXR=
- OA25932 - TSO/E support for read-only storage
- OA26802 - Other system REXX enhancements

Exploiters of this new support are:

- Health Checker.
- Security Server.
- Common Event Adapter.
- Any authorized assembler program can use the AXREXX macro.

Considerations with z/OS V1R11

z/OS V1R11 introduces a certain number of enhancements to SYSREXX, as follows:

- Data sets may be concatenated to SYS1.SAXREXEC which is achieved using the new REXXLIB statement in the AXRnn parmlib member.
- The TSO/E STORAGE function is supported in read-only mode.
- Parmlib members other than AXR00 may be specified using the new AXR= keyword in IEASYSxx.

- JES affinity is acquired for the AXR01-AXR08 address spaces after the primary JES subsystem starts. Every TSO=YES AXREXX request will have a JES job ID and joblog associated with it. This allows additional JES spool capabilities to be used within a REXX exec running under System REXX, including the ability to write JCL to an internal reader, retrieve output, and use the TSO/E TRANSMIT and RECEIVE commands.
- Support for z/OS UNIX callable services for TSO=YES AXREXX requests allows full use of available syscall commands.
- Two new built-in functions are provided:
  - AXRINFO will return the name of the primary subsystem under which the exec is running.
  - AXRWAIT provides a wait function to pause a REXX exec for a specified period of time.

25

Sysplex enhancements

IBM provides support within z/OS that allows authorized applications to query, change, and perform basic operational procedures against the installed System z hardware base. This support provides a set of high-level application program interfaces (APIs) for data exchange and command requests. The functionality, called base control program internal interface (BCPii), is delivered in the base of the operating system. The support not only allows control of the hardware that the APIs are executing on, but also extends to other System z processors within the attached process control network.

This support does not require communication on an IP network for connectivity to the support element (SE) and hardware management console (HMC). Calls using the BCPii APIs can be made from either C or Assembler programming languages.

This chapter discusses the following topics relative to sysplex availability:

- Introducing BCPii
- A new sysplex partitioning (SSD)
- New STP alerts messages
- How to automate these new facilities

25.1 From clustering toward closely coupling

With the various hardware systems and z/OS successive releases, over time an evolution has occurred within the Parallel Sysplex environment, as illustrated in Figure 25-1. The left side of the figure shows a Parallel Sysplex as first implemented in 1994. The right side of the figure shows a current configuration with the latest hardware (IBM z10 systems), as well as z/OS V1R11.

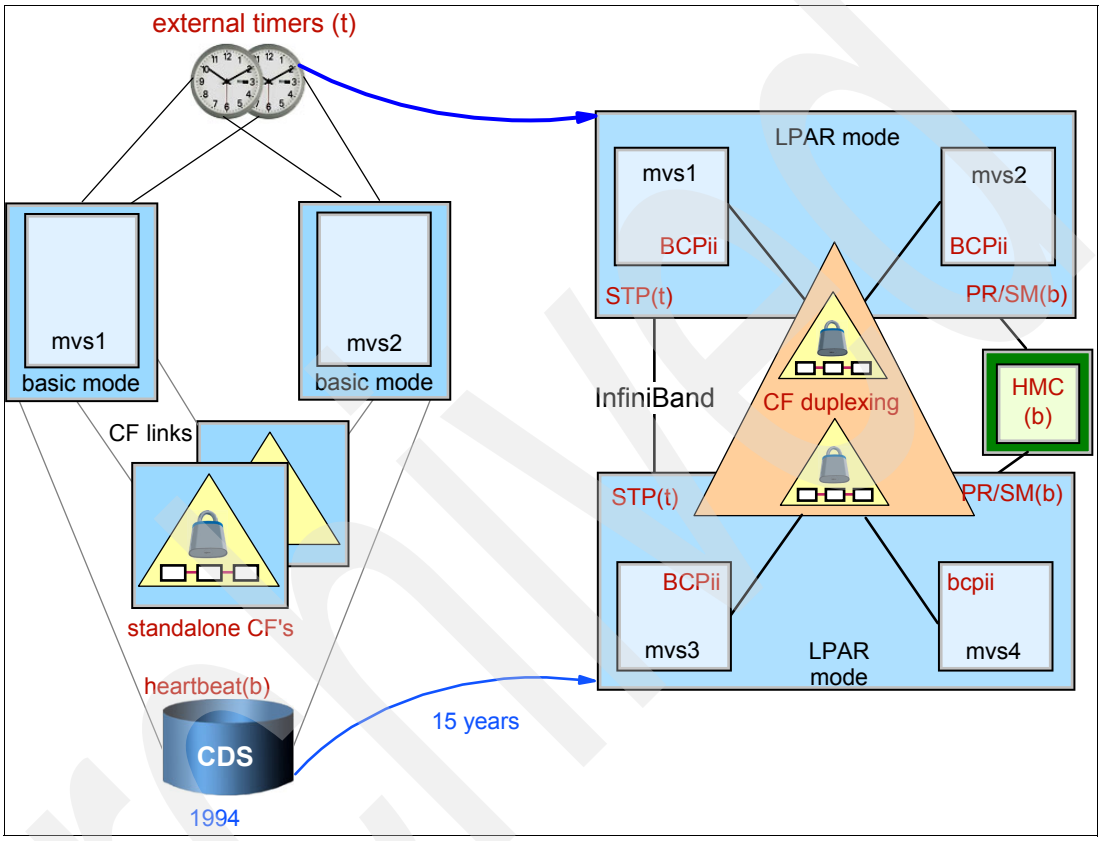


Figure 25-1 From clustering toward closely coupling

Basic mode

Originally, the operating systems were mainly running in basic mode. Today, operating systems run in LPAR mode along with the Coupling Facility, which provides duplexing for the required structures. You can think of the Coupling Facility (CF) as a piece of non-volatile electronic storage logically structured.

Parallel Sysplex

A Parallel Sysplex uses a kind of “heartbeat” on DASD (shown as function (b) in Figure 25-1) to verify that the other coupled systems are still functioning. With z/OS V1R11, a new way to verify the status of other systems has become available in certain cases through a set of new logical functions, namely BCPii, PR/SM, HMC, and the Support Element (SE). This is described in 25.2, “BCPii overview” on page 511 and in 25.3, “System partitioning using BCPii” on page 517.

Server Time Protocol

Timing synchronization (shown as function (t) in Figure 25-1 on page 510) was initially provided by external timers (necessarily redundant) sending signals to the coupled systems.

Today, Server Time Protocol (STP) exploits InfiniBand and uses messages to synchronize timers between the coupled systems.

STP is implemented within the hardware of the coupled systems. For communication, it uses the same physical link as those used by XCF and XES. The latest technology for those links is InfiniBand, which is available in z10 systems. With z/OS V1R11, the STP network begins to be visible at the operating system level. This opens up new opportunities for automatically managing the STP configuration.

25.2 BCPii overview

IBM provides support within z/OS that allows authorized applications to query, change, and perform operational procedures against the installed System z hardware base through a set of application program interfaces (APIs). These applications can access the System z hardware that the application is running on and extend their reach to other System z processors within the attached process control (Hardware Management Console) network.

By using the Base Control Program internal interface (BCPii), an authorized z/OS application can perform the following actions:

- Obtain the System z topology of the current interconnected Central Processor Complexes (CPCs), as well as the images, capacity records, and activation profiles on a particular CPC.
- Query various CPC, image (LPAR), capacity record, and activation profile information.
- Set various configuration values related to CPC, image, and activation profiles.
- Issue commands against both the CPC and image (LPAR) to perform minor or even significant hardware- and software-related functions.
- Listen for various hardware and software events that might take place on various CPCs and images throughout the HMC-connected network.

Communication to the SE and Hardware Management Console (HMC) using BCPii is performed completely within the base operating system and therefore does not require communication on an IP network for connectivity. This provides complete isolation of your System z hardware communication from any other network traffic within the intranet or Internet. Calls using the BCPii application programming interfaces can be made from either C or Assembler programming languages.

Note: As illustrated in Figure 25-2 on page 512, BCPii is a new address space. It provides a new API that allows authorized programs to communicate with the HMC.

It is totally unlike existing interfaces such as:

- Performing a topology function used by HiperDispatch to communicate with PR/SM.
- Diagnose (HVC or x'83') to communicate with the hardware or the Hypervisor.
- HWMxxxx service calls to communicate with the STP network.

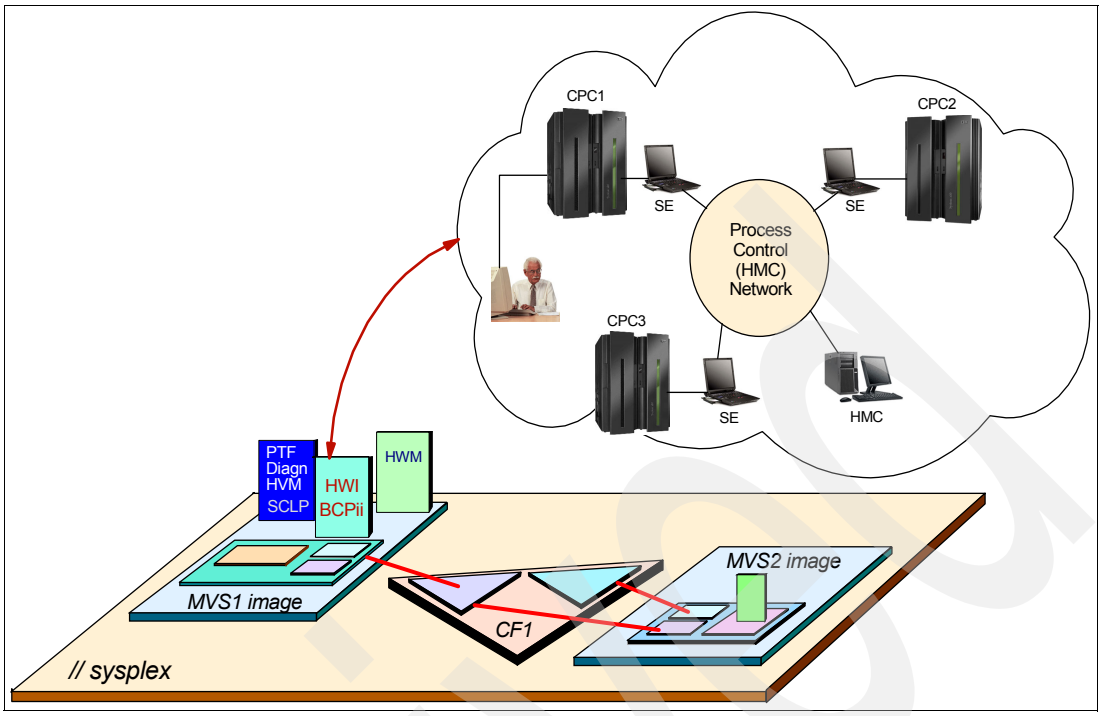


Figure 25-2 BCP internal interface

25.2.1 Using BCPii

As mentioned, BCPii provides an interface to authorized z/OS applications through the process control (HMC) network to implement the following:

- Monitor status or capacity changes.
- Obtain configuration data related to CPC or image.
- Re-IPL an image.
- Allow authorized z/OS applications to have HMC-like control over systems in the process control (HMC) network shown in Figure 25-3 on page 513.
- Complete communication isolation of existing networks (intranet/Internet) from the process control (HMC) network.
  - Communication to the SE is completely within the base z/OS.
- A new z/OS address space.

  The BCPii address space is the bridge between a z/OS application and the SE.
- A set of authorized APIs.

  The BCPii address space is mandatory for any BCPii API request. The system attempts to start the HWIBCPii address space during IPL.
- A communications transport between a z/OS application and the local SE.

  It also provides a communication transport between other SEs connected to other CPCs routed by the HMC. BCPii uses a low-level operating system connection to establish communication between an authorized application running on a z/OS image (LPAR) and the SE associated with the CPC that contains this z/OS image. You must configure the SE to permit these BCPii communications if BCPii services are required to be available by your installation.

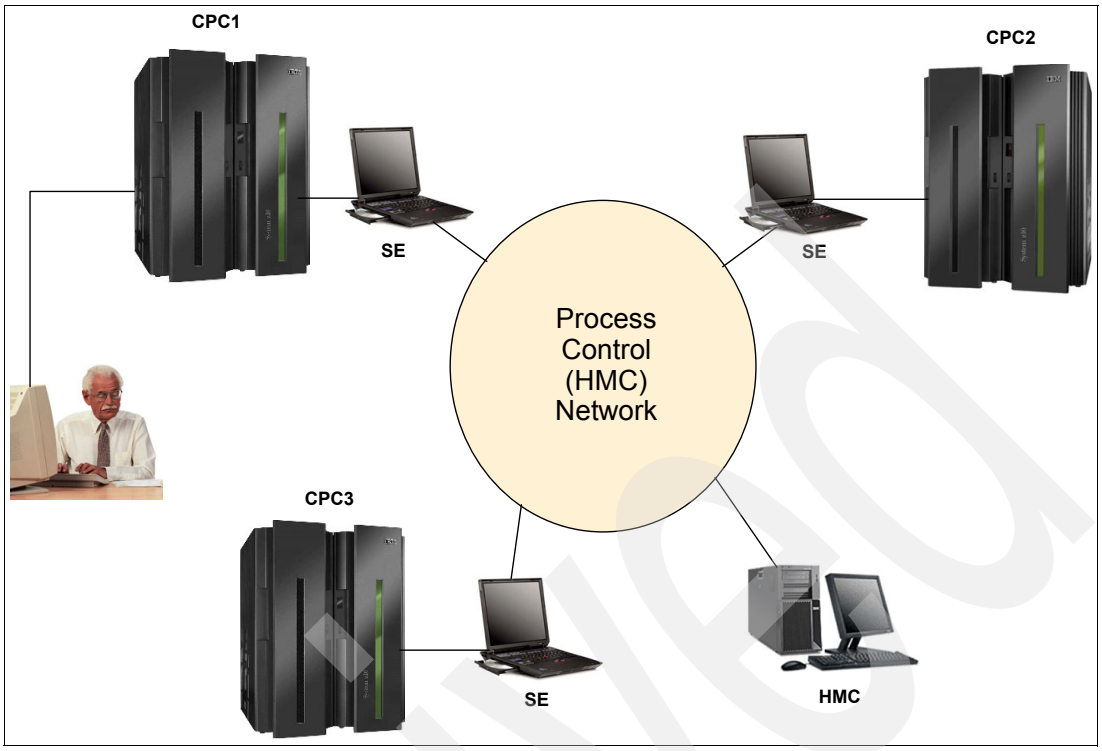


Figure 25-3 HMC network

25.2.2 BCPii address space

The BCPii address space, shown in Figure 25-4 on page 514, starts automatically at IPL. It requires initial setup before becoming active. It can be stopped if necessary and be restarted by using the HWISTART procedure.

The BCPii address space can perform the following tasks:

- Manage all application connections.
- Build and receive all internal communication requests to the SE.
- Provide an infrastructure for storage required by callers and by the transport communicating with the SE.
- Provide diagnostic capabilities to help with BCPii problem determination.
- Provide security authentication of requests.

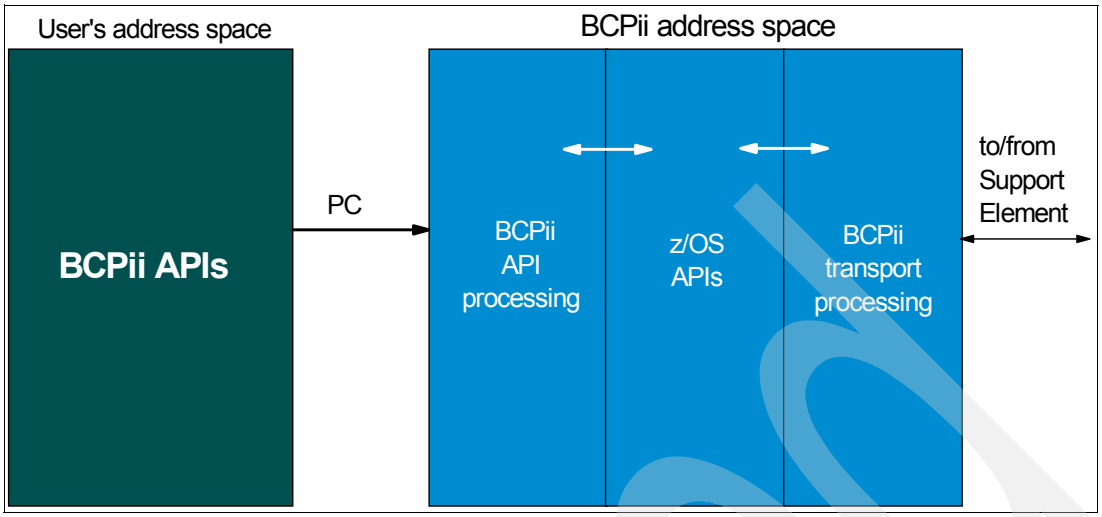


Figure 25-4 BCPii address space

Users of BCPii

BCPii is used by the following users:

- Vendor applications
  - Control center, systems management applications
- In-house applications for installations to manipulate System z resources, as follows:
  - Using a more automated, programmatic approach
  - From a z/OS environment rather than a hands-on HMC environment
  - Avoiding the need to have an intranet/Internet/internal network connect to the process control (HMC) network
- z/OS operating system components such as XCF

25.2.3 BCPii services

The following functions can be performed using BCPii APIs:

- Obtain the System z topology of the current interconnected CPCs, images (LPARs) and capacity records
- Query various CPC, image (LPAR) and capacity record information
- Issue commands against both the CPC and image to perform hardware and software-related functions
- Listen for various hardware and software events which can take place on various CPC and images.

BCPii service types

BCPii offers various types of services:

HWICMD Issue a BCPii Hardware Management Console command to perform a command against an HMC-managed object that is associated with central processor complexes (CPCs), CPC images (LPARs), and capacity records.

There can be CPC commands such as:

- Activate, deactivate an entire CPC

- CBU request (activate or undo)

- On/off capacity on demand request (activate or undo)

There can be image commands such as:

- Sysreset, load, start, stop, add, or remove temporary capacity, issue operating system command.

HWICONN Establish a BCPii logical connection between the application and a central processor complex (CPC), a CPC image (LPAR), a capacity record, or other types of activation profiles. This facilitates subsequent services to perform operations related to that CPC, image, capacity record, or activation profile.

Note: A connection remains active until a Disconnect service call (HWIDISC) is invoked; a loss of connectivity to the associated CPC has been detected by BCPii; or the address space of the caller is terminated. When the address space of the caller is terminated, the connection is disconnected implicitly by BCPii.

HWIDISC Release a BCPii logical connection between the application and the identified CPC, image, capacity record, or activation profile. If the connect token represents a CPC, any subordinate image, capacity record, or activation profile connection associated with the same CPC connection is also released.

HWIEVENT Register for BCPii events for the following purposes:

- Register an application and its connection to be notified for one or more hardware or software events occurring on the connected CPC or image, such as command completions, status changes, capacity changes, disabled waits, BCPii status changes.

- Delete the registration for notification of one or more previously registered events.

HWILIST Retrieve HMC and BCPii configuration-related information.

- List CPCs to list the CPCs interconnected with the local CPC.

- List images to list the images (LPARs) contained on an individual CPC.

- List capacity records to list the capacity records contained on an individual CPC.

- List events to list the events already registered on a particular BCPii connection.

Depending on what information is requested, the data returned by this service can be used on subsequent BCPii service calls to take the following actions:

- Connect to a central processor complex (CPC), image (LPAR), capacity record (CAPREC), reset activation profile, image

activation profile, or load activation profile using the HWICONN API.

- Register for the proper events (HWIEVENT) using the HWIEVENT API.

- Connect to the local CPC or Image.

HWIQUERY BCPii retrieval of SE/HMC-managed objects managed by the Support Element (SE) or hardware management console (HMC) related with central processor complexes (CPCs), CPC images (LPARs), capacity records, or other types of activation profiles.

HWISET BCPii change or set SE/HMC-managed objects data for Hardware Management Console (HMC)-managed objects associated with Central Processor Complexes (CPCs), CPC images (LPARs), or activation profiles and for a C application running in the UNIX System Services environment.

HWIBeginEventDelivery Begin delivery of BCPii event notifications to allow a C application running in the UNIX System Services environment to begin delivery of event notifications. This service must be issued before the HWIManageEvents service.

HWIEndEventDelivery End delivery of BCPii event notifications to allow a C application running in the UNIX System Services environment to end delivery of event notifications. This service unregisters the registration made by the HWIBeginEventDelivery service.

HWIManageEvents Manage the list of BCPii events to allow a C application running in the UNIX System Services environment to manage the list of events for which the application is to be notified. The HWIBeginEventDelivery service must have been called before the HWIManageEvents service being called because the appropriate delivery token returned from the HWIBeginEventDelivery service is required as input.

HWIGetEvent Retrieve outstanding BCPii event notifications to allow a C application running in the UNIX System Services environment to retrieve outstanding BCPii event notifications.

Security authorization

Proper authority is to be previously established in RACF, to allow callers to access those services or objects. To request BCPii services for programs from any address space, the environment must be as follows:

- Program-authorized.
- SAF-authorized.
- C and Assembler programming languages.
- UNIX System Services callers can receive event notifications through UNIX System Services-only services using the Common Event Adapter (CEA).

25.2.4 Information for query examples

The kinds of information that can be queried through BCPii are:

- CPC information
- General information, such as name, serial, machine type, ID, operating system information and networking information

- Status information
- Operating status and other status values
- Capacity information such as various CBU info, capacity on demand information, processor configuration, including IFA, IFL, ICF, and IIP
- Image information
- Capacity information such as defined capacity and processor weights
- Capacity record information, such as name, activation and expiration dates, activation days, record status, and the entire capacity record

25.3 System partitioning using BCPii

In a z/OS Parallel Sysplex, unresponsive systems must be partitioned from the sysplex in a timely manner, to avoid corruption of shared data and avoid causing “sympathy sickness” in the surviving systems of the sysplex.

XCF does not really know a system’s operational state, so it errs on the side of caution to avoid removing systems from the sysplex. However, the penalty for this caution is that the sysplex can suffer a workload processing interruption for a period of time dictated by the system’s failure detection interval.

To expedite and enhance the sysplex partitioning process, XCF now exploits the HMC remote query and remote reset functions (it actually goes down to the support element), accessed through the BCPii APIs. This use and exploitation of this new interface allows XCF to gain knowledge of the state of another image in the sysplex without operator intervention. This also shortens delays that are processed today when a system is in an indeterminate state.

Such expedient removal of unresponsive or failed systems from the sysplex is fundamental to the Parallel Sysplex approach to high availability. As a direct benefit, work can now be routed around the failure and processed on the surviving systems in the sysplex.

Partitioning prior to BCPii

z/OS systems occasionally experience critical situations where a system either enters a wait state or becomes “status update missing” and is generally non-responsive (hung, not processing work, or looping).

- The removal of unresponsive systems from the sysplex is the responsibility of the z/OS cross-system Coupling Facility (XCF) component of z/OS.
- The Sysplex Failure Management (SFM) sub-component of XCF extends the base XCF sysplex services in this area by providing a policy-based mechanism for improved automation of the removal of systems from the sysplex.

Prior to BCPii, XCF did not really know the operational state of a system that was in an unknown, “status-update missing” state. All that was known was that the system looked unresponsive because it was not performing system status updates.

XCF failure detection interval

Because of this lack of status information, XCF monitors the system for an interval of time (the failure detection interval) to allow the system a period in which to resume normal operation. The penalty of waiting this time interval, however, is that while XCF is waiting to see if the system resumes normal operation, the sysplex can suffer a workload processing interruption for a period of time dictated by the system’s failure detection interval.

In most sysplexes, this failure detection interval is set on the order of one and one-half to two minutes. There are limitations that constrain the XCF ability to remove an ailing system from the sysplex expeditiously. When operator intervention is required, the delay can extend to many minutes.

XCF cannot unambiguously determine that a system is truly dead. XCF must infer that it is appropriate to partition a system based on the amount of time elapsed since that system updated its heartbeat in the sysplex couple data set (CDS) or sent XCF signals.

XCF relies on the Coupling Facility to isolate systems by using commands to fence a target system from the channel subsystem. Isolation ensures data integrity when partitioning a system. Isolation can fail for many reasons, and when it does, XCF has no choice but to prompt the operator to perform a manual system reset.

25.3.1 BCPii as a new way of partitioning

This new support in z/OS V1R11 allows XCF to obtain certain knowledge of the state of another image in the sysplex to eliminate these constraints to the sysplex partitioning process

These enhancements to the partitioning protocol for removing a system from the system are called the system status detection (SSD) partitioning protocol. When this function is enabled, the monitoring system detects changes in target system status and can thereby bypass certain delays built into the status monitoring and partitioning processes. This support also enables the system to initiate actions, such as reset-normal, against the remote system to ensure that it is no longer actively processing.

This enhancement provides new support in XCF to exploit the HMC remote query and remote reset functions, accessed through the BCPii APIs, to expedite and enhance the sysplex partitioning process.

The BCPii support implemented in z/OS V1R11 provides mechanisms by which XCF can, in most cases, eliminate these constraints.

- The BCPii query capabilities allow XCF to determine whether a system is demised, without waiting for installation-specified intervals to elapse.
- A system is considered demised when:
  - The system enters a non-restartable wait state.
  - The system has been reset or reloaded.
  - The partition has been deactivated or check stopped.
- The BCPii remote command capability provides XCF an alternative to isolation, thereby allowing XCF to reset an image when fencing fails.

When SSD is enabled, XCF uses the BCPii APIs to expedite and enhance the sysplex partitioning process in the following areas:

- Bypass the failure detection interval (FDI), which is the current way of handling things
- Avoid IXC402D manual intervention
- Bypass the cleanup interval
- Avoid IXC102A manual intervention
- Validate a “DOWN” response
- Messages IXC102A and IXC402D

These topics are discussed in more detail in the following sections.

Bypass the failure detection interval (FDI)

FDI is used to indicate how long a system may appear unresponsive before other systems in the sysplex take disruptive actions against that system. Bypassing FDI means that:

- XCF instead uses the BCPii remote query capabilities to discover when another system has become demised.
- XCF immediately initiates sysplex partitioning without waiting for the FDI to elapse if the system can be determined to be demised. This typically speeds up partitioning by a few minutes.

Avoid IXC402D manual intervention

Message IXC402D is normally issued when a system has not updated its “heartbeat” in the sysplex CDS or has not sent XCF signals within the FDI and the SFM policy specifies an indeterminate status action of PROMPT. To avoid this:

- When XCF can establish through BCPii APIs that the unresponsive system is truly demised, it can bypass the FDI and immediately proceed with partitioning without issuing the WTOR.
- When XCF cannot establish that the unresponsive system is demised, it uses the BCPii remote reset capability to perform a system reset against the unresponsive system without issuing the WTOR unless the SFM Policy specified PROMPT.

Avoid IXC102A manual intervention

Message IXC102A is normally issued when SFM cannot successfully isolate the system image being removed and manual intervention is required. To avoid it means that XCF from now on uses the BCPii remote reset capability to perform a system reset against the system to be isolated without issuing the WTOR unless the SFM Policy specified PROMPT.

Bypass the cleanup interval

When varying a system out of the sysplex in response to operator command, XCF now uses the BCPii query capability to determine when the target system becomes demised. At that point, it is no longer necessary to wait for the cleanup interval to elapse because no further cleanup is taking place on the target. XCF can immediately continue with partitioning processing.

This is expected to speed up partitioning by whatever portion of the cleanup interval is not actually required for cleanup, perhaps 10 to 20 seconds.

Validate a “DOWN” response

Messages IXC102A and IXC402D invite the operator to reset an unresponsive system and to reply “DOWN” when the reset is complete. It is critical that the operator actually perform the reset before responding to the WTOR, because as soon as the response is received, XCF proceeds with partitioning, thereby freeing other systems to take over resources previously owned by the outgoing system.

If the target system has not in fact been reset, and is still attempting to manipulate those resources, data integrity problems can result. With this support, XCF uses BCPii query services to verify that the target system has in fact been reset before proceeding with partitioning, and re-prompts the operator if it has not. If this verification is not possible (that is, if BCPii services are not available or they fail for any reason), XCF acts as it does today, trusting the operator response and proceeding with partitioning.

25.4 Implementation considerations

To be able to exploit sysplex partitioning using BCPii, first ensure that BCPii is properly up and running.

Activate BCPii

BCPii attempts to auto-start during the IPL of z/OS.

- If the proper configuration has not been made for BCPii, it will graciously terminate.
- BCPii can be restarted by using the **START** command after BCPii has been configured properly.

To properly set up the BCPii configuration, follow these steps:

1. Configure the local SE to support BCPii.
2. Authorize an application to use BCPii.
3. Configure the BCPii address space.
4. Set up the event notification mechanism for UNIX System Services callers (if required).

The following sections explain these steps in more detail.

Configure the local SE to support BCPii

BCPii uses a low-level operating system connection to establish communication between an authorized application running on a z/OS image (LPAR) and the Support Element (SE) associated with the Central Processor Complex (CPC) that contains this z/OS image. You must configure the SE to permit these BCPii communications if BCPii services are required to be available by your installation.

Levels of hardware BCPii supports

The HWIBCPII address space, which supports the issuing of BCPii APIs from a z/OS image, runs on any level of hardware that supports the level of the z/OS operating system in which BCPii is included.

BCPii applications might need to perform hardware or software functions on CPCs other than the CPC on which the application is running. Such requests can be targeted to other System z hardware at a lower or higher hardware level than the local CPC, provided that these hardware levels are supported to coexist with the local CPC level.

The HWICMD service is only allowed to be targeted to at least a System z9 hardware level running on a particular microcode level. BCPii rejects the targeting of this service to any System z hardware level lower than System z9. Table 25-1 lists the minimum levels of microcode that are required on various hardware levels to run BCPii.

Table 25-1 Minimum levels of microcode for BCPii

Hardware level	Minimum microcode level
z9	G40965.133
z10	F85906.116

Each version of hardware has subtle (or sometimes, significant) changes in the way information is displayed and saved in the SE element. The examples serve as a guide only to where the actual definitions that need to be modified are located within the SE configuration windows.

Enable BCPii communications on the Support Element

You must enable cross-partition authority on the Support Element to allow the SE to accept the BCPii APIs flowing from the user application through the HWIBCPII address space. This setting controls whether a logical partition can issue a subset of control program instructions to other logical partitions activated on the same CPC.

Note: This setting must be selected on the local SE associated with the CPC of the image that the z/OS BCPii application is running on. It must also be selected for any other system for which BCPii communication is required.

Steps on the HMC

To change this setting, perform the following steps on the HMC:

1. Select the CPC that is required.
2. Open Single Object Operations.
3. Highlight the CPC icon.
4. Open the CPC Operational Customization task list.
5. Open the Change LPAR Security task, and the Change Logical Partition Security window displays.
6. Check the cross-partition authority check box for each image (LPAR) that you want to grant BCPii access. See Figure 25-5 on page 521, where we checked partition A01 (circled in red). At a minimum, the image (LPAR) where the BCPii address space is running must have this authority activated.

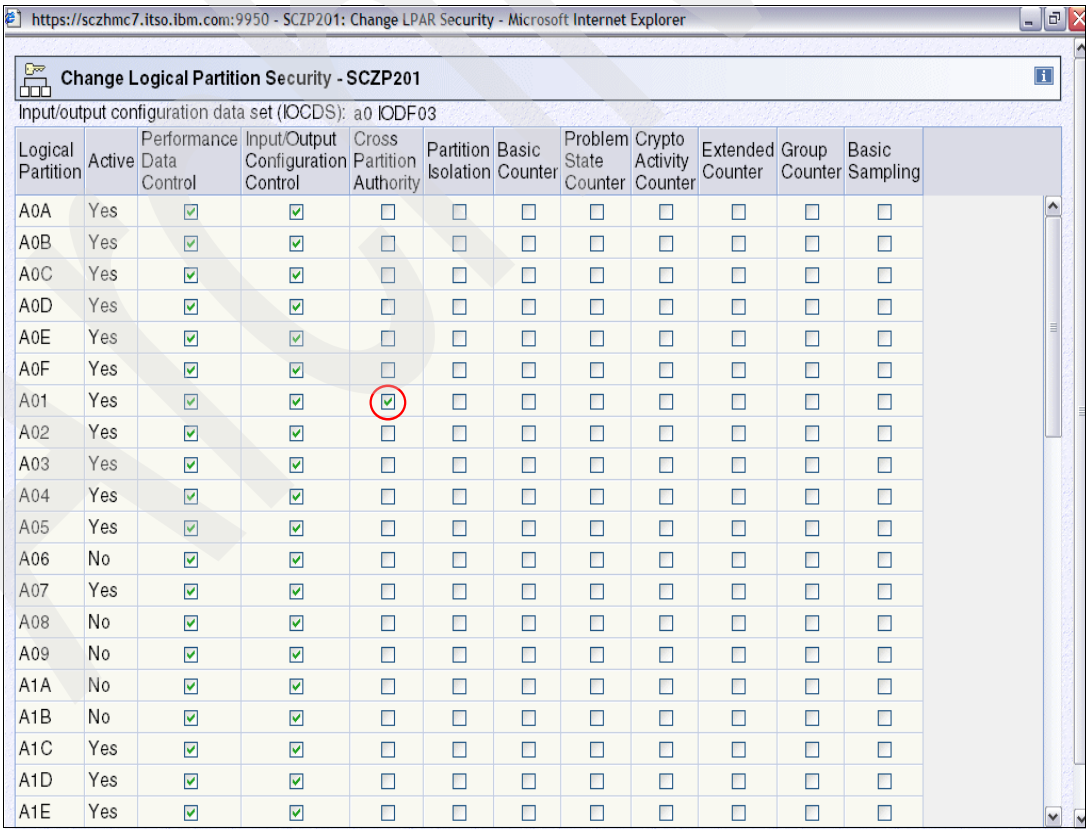


Figure 25-5 Change Logical Partition Security panel

7. When Save and Change is clicked, it might take time to update the various profiles. During this time the panel shown in Figure 25-6 is displayed.

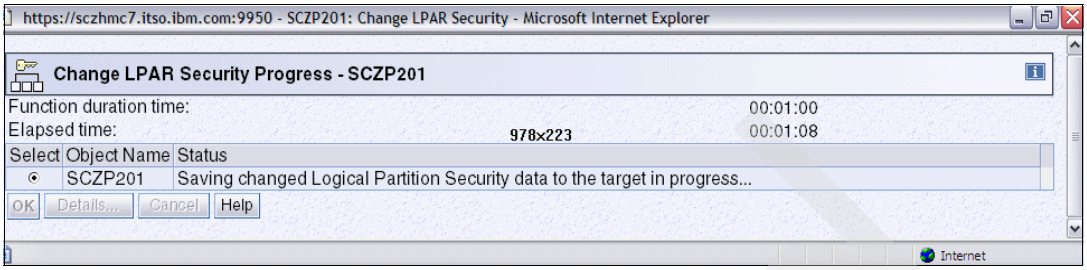


Figure 25-6 Change LPAR Security - save in progress

Failure to set this properly on the local SE associated with the image of z/OS that is running BCPii results in a severe BCPii address space initialization failure. You cannot start the address space, and will receive communications error X'101' with a reason code of X'D4'. Failure to set this up properly on remote SEs to which you want to connect results in the same return code and reason code on the HWICONN service call.

Note: Make the same updates to all CPCs that you want BCPii to communicate with and not just to the CPC from which the BCPii application is going to run on.

Define the BCPii Community Name on the Support Element

BCPii uses an SNMP community name to provide a level of security between the z/OS image that is executing the BCPii service and the SE itself. An *SNMP community* is a logical relationship between an SNMP agent and an SNMP manager. The community has a name, and all members of a community have the same access privileges: they are either read-only (members can view configuration and performance information) or read-write (members can view configuration and performance information, and also change the configuration).

To add the BCPii community name definition to the SE configuration, perform the following steps on the HMC:

1. Select the CPC that is required.
2. Open Single Object Operations.
3. Select the **Console Actions** view.
4. Select **Support Element Settings**.
5. Open the Customize API Settings.

Note: The Customize API Settings button appears in the Support Element Settings panel only if you are logged on the ACSADMIN HMC user ID.

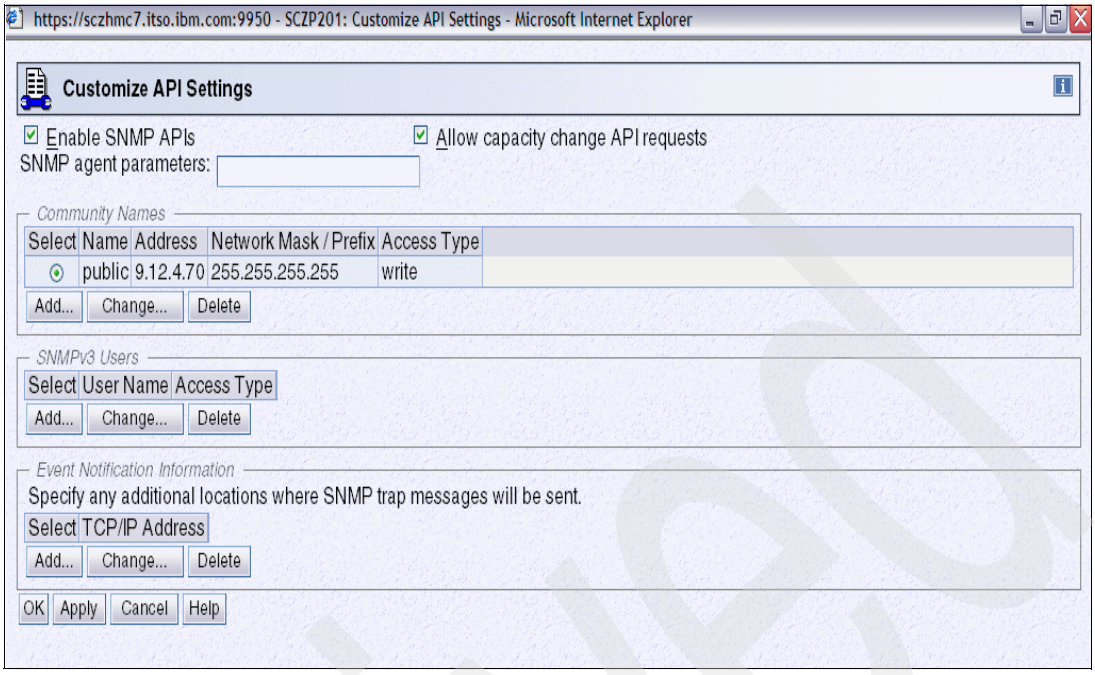


Figure 25-7 Customize API Settings

6. Check the **Enable SNMP APIs** check box as shown in Figure 25-7.
7. Consider checking the **Allow capacity change API requests** check box on a z10 or higher system (as shown in Figure 25-7) if the installation is to allow a BCPii application to perform temporary capacity upgrades.

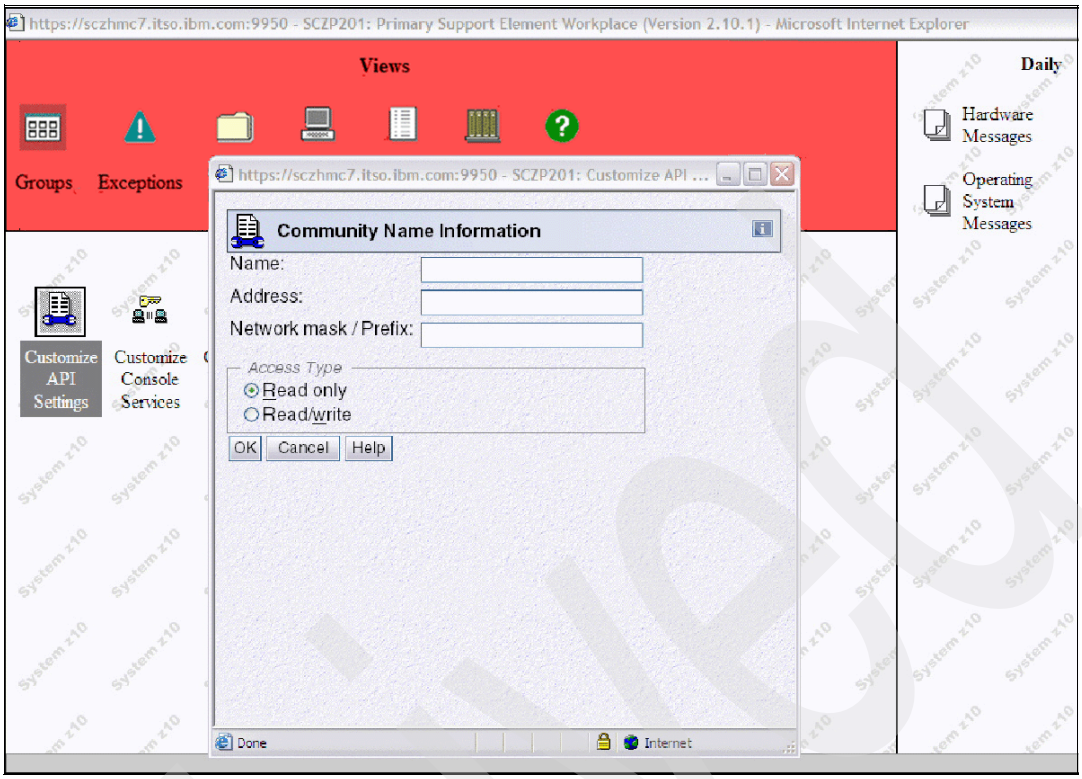


Figure 25-8 Community name information

On a Customize API Settings panel, you can add a Community Name as shown in Figure 25-8.

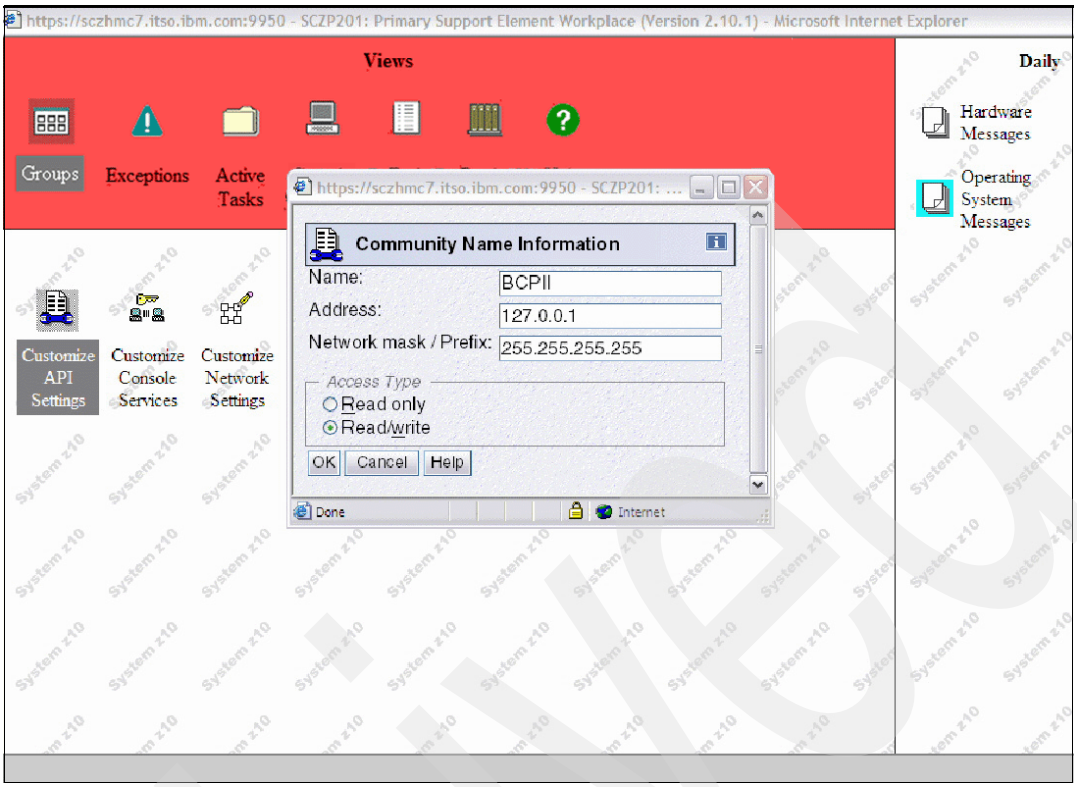


Figure 25-9 Community name in Customize API Settings

8. Make sure that the SNMP agent parameters are blank.
9. Add a BCPii community name, as shown in Figure 25-9. Click **Add**. When a window is prompted, fill in the following fields:

   Name: The actual SNMP community name. This value is a 1- to 16-character alphanumeric field. Because of restrictions with the security products on z/OS, the BCPii SNMP community name must not contain any lowercase characters.

   Address: For BCPii, the address, which is referred as a loop-back address, must be 127.0.0.1.

   Network mask/Prefix: This must be 255.255.255.255.

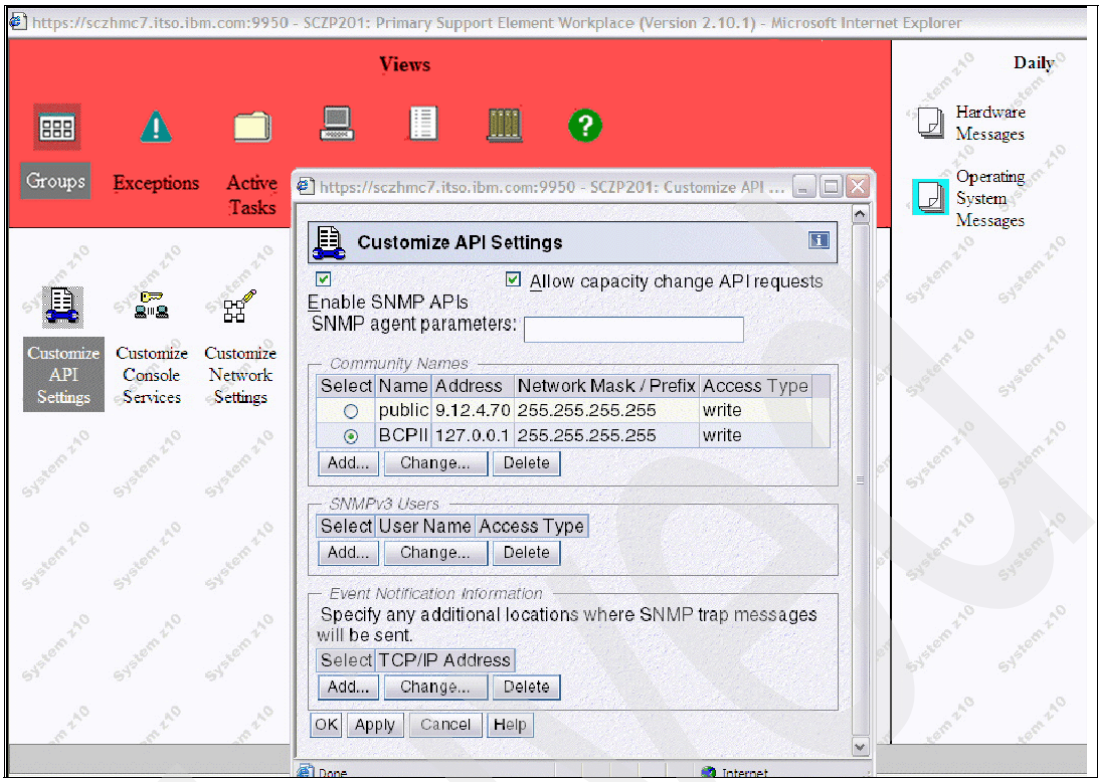


Figure 25-10 Custom API settings after apply

When you select **Apply**, the new Community Name is reflected in the Customize API Settings as shown in Figure 25-10. Failing to set this properly on the local SE associated with the image of z/OS that is running BCPii results in a severe BCPii failure and you cannot start the address space. Message HWI014I is issued if the community name defined on the SE for the local CPC does not match the definition in the security product for the local CPC.

Note: Make the same updates to all CPCs that you want BCPii to communicate with.

Setting up authority to use

Given the nature of the BCPii APIs and the capabilities of a BCPii application to potentially modify vital hardware resources:

- A number of authority validations are performed for each BCPii requestor.
- The API is not virtualizable.

A BCPii application needs to have program authority; general security product authority to be able to issue BCPii commands; authority to the particular resource that the application is trying to access; and a community name defined in the security product for each CPC to which communication is required.

Program authority

BCPii applications must be program-authorized, meaning that one of the following must be true of the application:

- It is running in supervisor state.
- It is running in an authorized key with PSW key mask (PKM) between 0 and 7.
- It resides in an APF-authorized library.

General security product authority

A BCPii application must have general authority to use BCPii. The profile HWI.APPLNAME.HWISERV in the FACILITY resource class controls which applications can use BCPii services. The security administrator must give at least read authority to this resource, in addition to granting authority to any specific resource that the application is attempting to access. In addition, BCPii requires that the FACILITY class to be RACLIST-specified.

Figure 25-11 shows a RACF example allowing user IBMUSER to use BCPii services in general.

```
RDEFINE FACILITY HWI.APPLNAME.HWISERV UACC(NONE)
PERMIT HWI.APPLNAME.HWISERV CLASS(FACILITY) ID(IBMUSER) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 25-11 Authority granting to BCPii resources

User IBMUSER is a user ID assigned by RACF to the HWISTART started task. Generic definitions may be created instead of specific users if the installation does not have specific definitions for every user.

Authorize an application to use BCPii specific resource

A BCPii application needs to have authority to the particular resource that it is trying to access. That particular resource can be the CPC itself, an image (LPAR) on a particular CPC, or a particular capacity record on a particular CPC. BCPii needs a profile defined in the FACILITY resource class that represents the target of the particular BCPii request. The profile name required to be defined depends on the type of the particular resource required.

Table 25-2 Specific resources to be authorized for

Request type	FACILITY class profile required
CPC	HWI.TARGET.netid.nau where netid.nau represents the 3- to 17-character SNA name of the particular CPC.
Image	HWI.TARGET.netid.nau.imagename where netid.nau represents the 3- to 17-character SNA name of the particular CPC and imagename represents the 1- to 8-character LPAR name.
Capacity record	HWI.CAPREC.netid.nau.caprec where netid.nau represents the 3- to 17-character SNA name of the particular CPC and caprec represents an 8-character capacity record name.
Activation profiles	HWI.TARGET.netid.nau where netid.nau represents the 3- to 17-character SNA name of the particular CPC the activation profile is defined.

The SNA name is comprised of the Netid.name defined at the SE. Netid is found in the panel Customize Network Settings, as shown in Figure 25-12 on page 528.

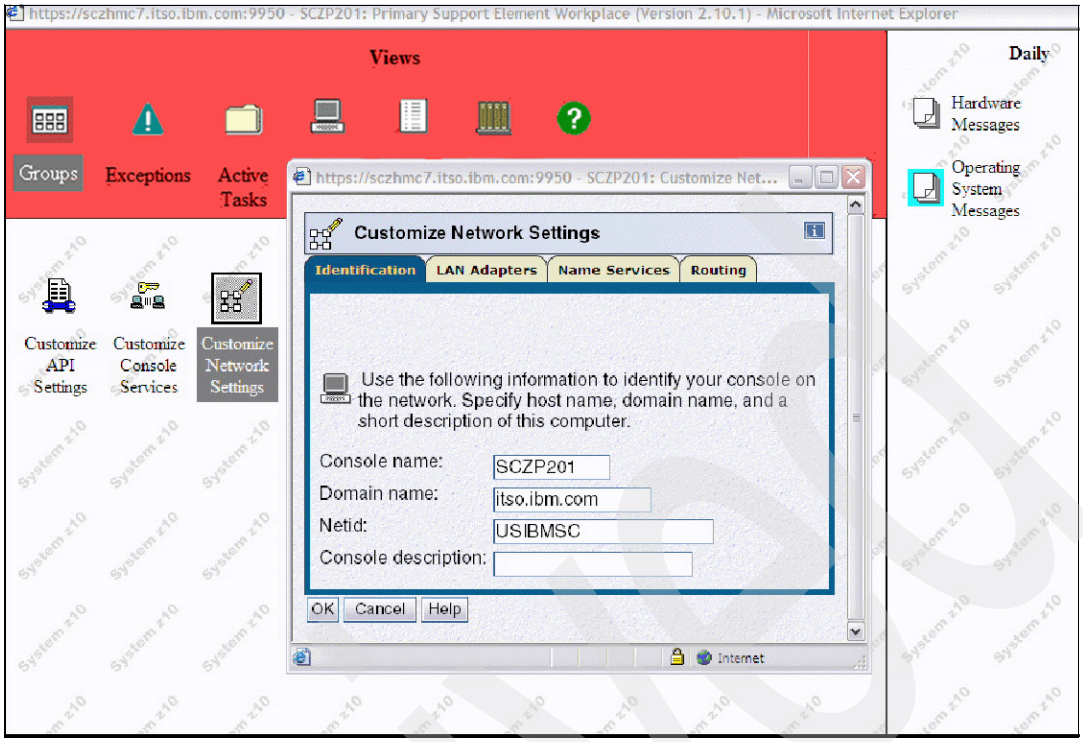


Figure 25-12 Netid on SE panel

The value of name in the SNA name is the SNA name of the CPC from which Single Object Operations has been logged on, as highlighted in Figure 25-13.

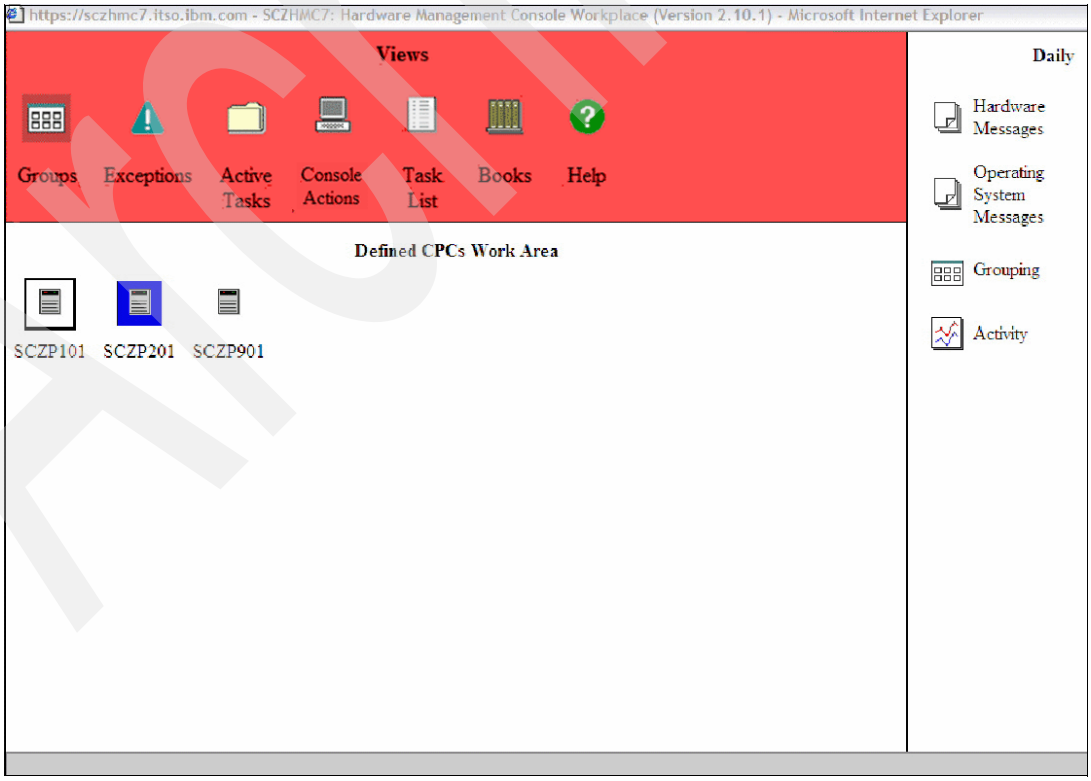


Figure 25-13 CPC name

The access level required for the particular profile depends on the service that the BCPii application attempts to issue. See the BCPii API documentation for specific information about the minimum access level required for each BCPii API service.

Figure 25-14 shows a RACF example allowing a user to have Connect, Event, List, and Query access to the SNA name of the CPC (USIBMSC.SCZP201, in our case) using the community name BCPII.

```
RDEFINE FACILITY HWI.TARGET.USIBMSC.SCZP201 UACC(NONE) APPLDATA('BCPII')
PERMIT HWI.TARGET.USIBMSC.SCZP201 CLASS(FACILITY) ID(IBMUSER) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 25-14 Authorizing access to CPC resource

Figure 25-15 shows a RACF example granting a user with Command, Connect, Event, List, Query, and Set access to any image (LPAR) on USIBMSC.SCZP201.

```
RDEFINE FACILITY HWI.TARGET.USIBMSC.SCZP201.* UACC(NONE)
PERMIT HWI.TARGET.USIBMSC.SCZP201.* CLASS(FACILITY) ID(IBMUSER) ACCESS(ALTER)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 25-15 Authorizing access to LPAR resource

Community name defined in the security product for each CPC

BCPii uses an SNMP community name to provide a minimal level of security between the z/OS image executing the BCPii service and the Support Element. An SNMP community name is associated with a particular CPC. The same SNMP community name that was defined in the SE configuration for a particular CPC also must be defined in the security product for each CPC to which communication is required. This community name definition is extracted from the security product by BCPii and propagated to the SE. The SE validates that the community name passed by BCPii is correct before proceeding with the request.

To define the BCPii community name in the security product, use the **APPLDATA** field with the CPC profile definition to associate a community name with a particular CPC. The **APPLDATA** field for the BCPii community name contains a 1- to 16-character alphanumeric field. Because of restrictions with the security products on z/OS, the BCPii SNMP community name must not contain any lowercase characters.

Figure 25-16 shows a RACF example assigning a BCPii community name of `BCPII` to an existing CPC definition for a SNA name of CPC, USIBMSC.SCZP201.

```
RALTER FACILITY HWI.TARGET.USIBMSC.SCZP201 APPLDATA('BCPII')
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 25-16 Assigning a BCPii community name

Configure the BCPii address space

As previously mentioned, the BCPii address space is the bridge between a z/OS application and the SE. The address space can perform the following tasks:

- Manage all application connections.
- Build and receive all internal communication requests to the SE.

- Provide an infrastructure for storage required by callers and by the transport communicating with the SE.
- Provide diagnostic capabilities to help with BCPii problem determination.
- Provide security authentication of requests.

The BCPii address space is mandatory for any BCPii API request. The system attempts to start the HWIBCPii address space during IPL. BCPii requires the *high-level qualifier*.SCEERUN2 and *high-levelqualifier*.SCEERUN data sets to be in the link list concatenation. IBM specifies these data sets in the default link list members (PROGxx) in z/OS V1R10 and higher. Failure to have these two data sets in the link list means that BCPii cannot be started, and such failure is accompanied by error message HWI009I indicating that BCPii was unable to load a required Language Environment part.

Set up the event notification mechanism for UNIX System Services callers if required

Applications running in a started procedure, batch, TSO or other non-UNIX System Services environment can use the HWIEVENT service and provide their own ENF exit that receives control when the application-requested events occur on the target CPC or image.

Applications running in a UNIX System Services environment do not have normal ENF exit processing capabilities available and cannot readily listen for ENF signals. The Common Event Adapter (CEA) address space allows UNIX System Services applications to be able to receive such event notifications. BCPii provides several services that use the CEA functionality to deliver these same events to UNIX System Services callers. See the documentation for the UNIX System Services-only services of BCPii in previous 25.2.3, “BCPii services” on page 514.

CEA address space setup

The use of the CEA address space by BCPii requires minor CEA setup steps before UNIX System Services-only services of BCPii can work properly. The Common Event Adapter (CEA) address space must be active to allow the UNIX System Services-only services of BCPii to operate. CEA has two modes of operation, namely minimum mode and full-function mode. If the UNIX System Services-only services of BCPii are required to be available, then CEA must be running in full-function mode. CEA, like BCPii, starts as part of a system IPL.

CEA ENF security configuration

A UNIX System Services BCPii application must be granted authority to listen to ENF68 events. With the CEA ENF controls, it is also possible to fine-tune which BCPii events that a user is allowed to listen to.

Figure 25-17 shows a RACF example giving e generic authority to the user ID associated with a UNIX System Services application authority to listen to any BCPii event.

```
AU user_id OMVS(Uid(n))
SETROPTS GENERIC(SERVAUTH)
RDEFINE SERVAUTH CEA.CONNECT UACC(NONE)
RDEFINE SERVAUTH CEA.SUBSCRIBE.ENF_0068* UACC(NONE)
PERMIT CEA.CONNECT CLASS(SERVAUTH) ID(user_id) ACCESS(READ)
PERMIT CEA.SUBSCRIBE.ENF_0068* CLASS(SERVAUTH) ID(user_id) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Figure 25-17 Authorize ENF 68 access

To give specific authority to only certain BCPii events, use the event qualifier as part of the profile name. The event qualifier maps to the event mask for ENF68 in the ENFREQ documentation in *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*. Hardware events are in the form '03xx00yy' where xx is the event source ('01'x = CPC, and '02'x =image) and yy denotes the particular event.

Figure 25-18 shows a RACF example allowing a user authority to only receive events related to CPC command responses (CmdResp = '01'x).

```
AU JOE OMVS(Uid(5))
RDEFINE SERVAUTH CEA.CONNECT UACC(NONE)
RDEFINE SERVAUTH CEA.SUBSCRIBE.ENF_006803010001 UACC(NONE)
PERMIT CEA.CONNECT CLASS(SERVAUTH) ID(JOE) ACCESS(READ)
PERMIT CEA.SUBSCRIBE.ENF_006803010001 CLASS(SERVAUTH) ID(JOE) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Figure 25-18 Allowing a user to receive CPC events

BCPii startup and shutdown

The BCPii address space normally does not need to be started or shut down. BCPii initialization occurs during system IPL. If the configuration is correct, no further action is required. The address space remains active and ready to handle BCPii requests.

BCPii address space does not start up at IPL

If the HWIBCPii address space is not active after an IPL, look for HWI* messages in the system log. Most of the time, these messages pinpoint the reason for the failure of BCPii to become active.

In most cases, the address space did not start for one of two main reasons:

- The Support Element that controls the CPC that contains the image of z/OS on which BCPii is being started has the improper configuration. Make sure all the steps have been followed in "Configure the local SE to support BCPii" on page 520.
- The community name of this local CPC is either not defined in the security product or contains an incorrect value. This is accompanied by message HWI014I. See "Community name defined in the security product for each CPC" on page 529.

After these problems have been corrected, restart the BCPii address space, as shown in Figure 25-19.

```
000280  S HWISTART
000080  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 001
000080          TO START HWISTART WITH JOBNAME HWISTART.
000080  $HASP100 HWISTART ON STCINRDR
000280  IEF695I START HWISTART WITH JOBNAME HWISTART IS ASSIGNED TO USER
        IBMUSER , GROUP SYS1
000080  $HASP373 HWISTART STARTED
000080  $HASP395 HWISTART ENDED
000080  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 006
000080          TO START HWIBCPII WITH JOBNAME HWIBCPII.
```

Figure 25-19 Start HWISTART

After all the security-related definitions have been corrected the BCPii becomes active, as shown in Figure 25-20 on page 532.

```
000280  IEE252I MEMBER CTIHWI00 FOUND IN SYS1.IBM.PARMLIB
000280  IEF196I IEF285I    SYS1.PARMLIB                                  KEPT
000280  IEF196I IEF285I    VOL SER NOS= BH8CAT.
000280  IEF196I IEF285I    CPAC.PARMLIB                                  KEPT
000280  IEF196I IEF285I    VOL SER NOS= Z19CAT.
000280  IEF196I IEF285I    SYS1.IBM.PARMLIB                              KEPT
000280  IEF196I IEF285I    VOL SER NOS= Z1BRC1.
000080  HWI016I THE BCPII COMMUNICATION RECOVERY ENVIRONMENT IS 031
000080  NOW ESTABLISHED.
000280  HWI007I BCPII IS ATTEMPTING COMMUNICATION WITH THE LOCAL CENTRAL 032
000280  PROCESSOR COMPLEX (CPC).
000080  HWI001I BCPII IS ACTIVE.
000080  IXC104I SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY: 034
000080    SYSTEM CANNOT TARGET OTHER SYSTEMS.
000080      REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL
000080    SYSTEM IS NOT ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
000080      REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL
```

Figure 25-20 BCPii is active

As shown in Figure 25-19 on page 531, messages IXC104I can then be issued when the sysplex CDS is not properly formatted for system status definition to be eligible. See 25.4.1, “Sysplex partitioning using BCPii” on page 532 to correctly set up this CDS.

Ending the HWIBCPii address space

The application of certain kinds of code maintenance or other unusual circumstances might require that the BCPii address space be stopped. To stop the BCPii address space, issue the **STOP** command for the BCPii address space P HWIBCPII. If not enough, you can use **CANCEL** and then **FORCE**.

BCPii issues an ENF 68 broadcast to notify interested ENF listeners that BCPii services are no longer available.

Restarting the HWIBCPII address space

After the BCPii address space has ended, it can be restarted. A procedure supplied by IBM in SYS1.PROCLIB allows the BCPii address space to be restarted. Issue the **S HWISTART** command to restart the HWIBCPII address space. When message HWI001I appears, confirming that BCPii is active, then BCPii requests continues.

BCPii issues an ENF 68 broadcast when the address space has completely initialized to notify interested ENF listeners that BCPii services are now available.

25.4.1 Sysplex partitioning using BCPii

The setting of the sysplex partitioning switch on a given system governs both that system’s use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems.

During XCF initialization an SSD monitor task is attached and a control block is updated with system information including an IPL token, the CPC name, the image name, and BCPii connector tokens. An ENF listener requests notification of ENF 68 signals from BCPii and the local system’s SSD eligibility status is updated. This determines whether a system can use BCPii to take action against another system, as well as whether other systems can take action against the local system.

Every system updates its heartbeat in the sysplex CDS every three seconds and looks at the status of the other systems. If the SSD protocol detects that a system has not updated its heartbeat for at least six seconds, it will issue a BCPii query to determine the status of that system. If the response to the BCPii query shows that the system is demised, the SSD protocol will take action to remove the system from the sysplex.

New functions, particularly those that are potentially disruptive, are shipped with switches to permit an installation to disable the function if desired. This sysplex partitioning enhancements includes also an enabling switch. Installation control can be by means of the COUPLExx OPTIONS statement in SYS1.PARMLIB(COUPLExx)" or **SETXCF OPTIONS** command.

Enabling the new protocol with this command might not be sufficient to cause the system to begin using it. Other environmental conditions must also be established, such as the availability of a sysplex couple data set formatted to support the larger records required by this protocol.

Enabling system status detection partitioning protocol

The following tasks and prerequisite hardware and software are required to enable the system status detection (SSD) partitioning protocol:

- The XCF address space must have adequate authorization through the z/OS Security Server (RACF) or an equivalent security product to access BCPii-defined FACILITY class resources. See "Assigning the RACF TRUSTED attribute" in *MVS Initialization and Tuning Reference* for information about using RACF to assign the TRUSTED attribute to the XCF address space.
- BCPii is configured to invoke z/Series Hardware APIs. See "Activate BCPii" on page 520 for more information about this topic.
- The system must operate on a z10 EC GA2 at Driver-76 or newer hardware and be at z/OS V1R11 to take full advantage of the functionality associated with the system status detection partitioning protocol.
  - A system running on z/OS V1R11 and downlevel hardware is only eligible to target other systems that are enabled to exploit the full functionality of the SSD partitioning protocol.
  - A system not running on the requisite hardware cannot be the target of SSD partitioning protocol functions.
- A sysplex couple data set formatted using the ITEM NAME(SSTATDET) NUMBER(1) control statement must be the primary couple data set in service in the sysplex.
- Install toleration PTFs for OA26037 on z/OS V1R10 and V1R9 systems in the sysplex to use the newly formatted sysplex couple data set required by the protocol.
- Ensure that the optional SYSSTATDETECT function is ENABLED on z/OS V1R10 and V1R9 systems. The SYSSTATDETECT function is enabled by default in z/OS V1R11.

The setting of the SYSSTATDETECT function on a given system will govern both that system's use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems.

Whenever a system detects that any of the environmental factors preventing the enablement of the SSD partitioning protocol have changed, it will attempt to enable the protocol. In practice, this means detecting either the availability of a version 5 sysplex CDS or a change in the installation controls, because the other environmental factors will not change without an IPL.

- When either of these factors changes, the system will report the status of the protocol with message IXC104I.
- The current setting of the SYSSTATDETECT function can be determined by issuing a **DISPLAY XCF,COUPLE** command.

Enabling the SYSSTATDETECT function

The SYSSTATDETECT function can be enabled by using the COUPLExx parmlib member and dynamically by using a command.

New PARMLIB changes

The FUNCTIONS statement allows you to enable and disable the SYSSTATDETECT function:

- FUNCTIONS ENABLE(SYSSTATDETECT)
- FUNCTIONS DISABLE(SYSSTATDETECT)

The default is ENABLE.

Dynamic enablement of system status detection protocol

Whenever a system detects that any of the environmental factors preventing the enablement of the SSD partitioning protocol have changed, it attempts to enable the protocol. In practice, this means detecting either the availability of a version 5 sysplex CDS or a change in the installation controls, because the other environmental factors cannot change without an IPL. When either of these factors changes, the system reports the status of the protocol with message IXC104I.

The setting of the switch on a given system governs both that system's use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems, as follows:

- The setting controls whether this system can employ the system status detection partitioning protocol when removing other systems from the sysplex.
- The setting controls whether other systems can employ the system status detection partitioning protocol when removing this system from the sysplex.

Parameters ENABLE| DISABLE of the command shown in Figure 25-21 indicates whether the options specified in the associated list are to be enabled or disabled. This is a required keyword. (opt1[,opt2 …]) identifies the options that are to be enabled or disabled.

Optional functions can also be enabled or disabled through the COUPLExx parmlib member.

```
SETXCF FUNCTIONS,{ENABLE|DISABLE}=SYSSTATDETECT
```

Figure 25-21 SETXCF FUNCTIONS

Message IXC104I is issued to indicate this system's eligibility to participate in the system status detection partitioning protocol.

```
FUNCTIONS ENABLE (SYSSTATDETECT)

                         DISABLE (SYSSTATDETECT)
```

Figure 25-22 Options in COUPLExx

This support introduces the new optional FUNCTIONS parameter SYSSTATDETECT, corresponding to the system status detection partitioning protocol. Use this parameter to enable the new protocol, or to disable it and revert to the existing partitioning protocol. The default setting is ENABLED.

Sysplex CDS

The enhanced partitioning depends on the availability of each system's addressing information to every other system in the sysplex. No system can apply the SSD partitioning algorithms (bypass FDI or cleanup interval, initiate remote reset, and so on) to any other system unless the sysplex is operating with sysplex CDSs containing the expanded system record. The addressing information is also necessary for correlating systems with the CPC on which each resides, so even the case in which a CPC is observed to have checkstopped requires the larger CDS.

Because the expansion of the system record is an incompatible change, support is required to accommodate it. Systems running z/OS V1R11 understand the new larger record. A toleration PTF is provided at lower releases (down to z/OS V1R9) to enable downlevel systems to process the larger record and higher version number.

Formatting a sysplex couple data set to support this protocol increments the version number associated with the couple data set. Such a couple data set cannot be brought into service if there are any systems active in the sysplex that do not have the code (either z/OS V1R11 or the toleration PTF) to accommodate the higher version number. Similarly, after such a couple data set is in use, no system without the code to accommodate it can join the sysplex.

SYSSTATDETECT protocol requires a SYSPLEX CDS formatted with SSTATDET, as follows:

- Formatting a sysplex couple data set to support the SSD protocol increments the version number associated with the couple data set and expands the System Record.
- The expanded system record contains information for the local system as well as information for each of the other remote systems in the sysplex that enables the SSD partitioning algorithms to function.

DISPLAY XCF,COUPLE

As depicted in Figure 25-23 on page 536, you can enter a **`DISPLAY XCF,COUPLE`** command to display summary information about the couple data sets:

- Information about the primary and alternate data sets, including Sysplex Failure Management-related information.
- Information about the system parameters set by the COUPLE statement in the COUPLExx parmlib member.
- Information related to the system status detection partitioning protocol.

Status of optional functions are"

- SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS
  - This indicates whether this system can employ the system status detection partitioning protocol when removing other systems from the sysplex.
  - If the system status detection partitioning protocol is not enabled, and enablement is desired, refer to the description for message IXC104I which lists the required action to correct the limiting factor.
- SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS:
  - This indicates whether other systems can employ the system status detection partitioning protocol when removing this system from the sysplex.
  - If the system status detection partitioning protocol is not enabled, and enablement is desired, refer to the description for message IXC104I which lists the required actions to correct the limiting factor.

```
------------------------- LIST MCS COMMAND OUTPUT --------- Row 1 to 24 of 107
C => _                                                              S => HALF
Mode: BOTH   SC65     LAFITTE
Display =>  Time     Y  System  Y  Job     Y  Hold => N  (Y or N)
                                          14:56 09/05/06 Enter '?' for HELP
------------------------------------------------------------------------------
 IXC357I  14.56.51  DISPLAY XCF 109
 SYSTEM SC65 DATA
    INTERVAL    OPNOTIFY     MAXMSG    CLEANUP      RETRY   CLASSLEN
         165         168       2000         15         10        956

    SSUM ACTION  SSUM INTERVAL  SSUM LIMIT     WEIGHT  MEMSTALLTIME
        ISOLATE              0          60         10            NO

    DEFAULT USER INTERVAL:     165
    DERIVED SPIN INTERVAL:     165
    DEFAULT USER OPNOTIFY: +     3

    MAX SUPPORTED CFLEVEL: 16

    MAX SUPPORTED SYSTEM-MANAGED PROCESS LEVEL: 16

    CF REQUEST TIME ORDERING FUNCTION: INSTALLED

    SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY:
      SYSTEM CANNOT TARGET OTHER SYSTEMS.
        REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL
      SYSTEM IS NOT ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
        REASON: SYSPLEX COUPLE DATA SET NOT FORMATTED FOR THE PROTOCOL

 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 25-23 DISPLAY XCF,COUPLE

The general syntax for this command is depicted in Figure 25-24.

```
DISPLAY XCF,COUPLE
IXC357I  hh.mm.ss  DISPLAY XCF
SYSTEM sysname DATA
SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY:
SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS.
  [REASON: targetotherrsn]
SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
  [REASON: othertargetrsn]
SYSTEM NODE DESCRIPTOR: type.mfg.plant.sequence
             PARTITION: partition side CPCID: cpcid
 SYSTEM IDENTIFIER: sysid
 NETWORK ADDRESS: snaaddr
PARTITION IMAGE NAME: image
 IPL TOKEN: ipltoken
```

Figure 25-24 DISPLAY XCF COUPLE syntax

25.4.2 Enhancements to the partitioning protocol

When this function is enabled, the partition monitoring system detects changes in target system status and can bypass certain delays built into the partitioning process. This support also enables the system to initiate actions, such as reset-normal, against the remote system to ensure that it is no longer actively processing.

As depicted in Figure 25-25 on page 537, there is a new set of lines, starting with the `ADDITIONAL INFORMATION:` header.

```
COUPLEXX PARMLIB MEMBER USED AT IPL: COUPLExx OPTIONAL FUNCTION STATUS:
FUNCTION NAME         STATUS               DEFAULT
SYSSTATDETECT      ENABLED             ENABLED
SYSPLEX COUPLE DATA SETS
PRIMARY   DSN: dsname
          VOLSER: prisysvol     DEVN: prisysdev
          FORMAT TOD            MAXSYSTEMMAXGROUP(PEAK)     MAXMEMBER(PEAK)
          mm/dd/yyy hh:mm:ss    maxsys  maxgroup(peakgrp) maxmember(peakmem)
ADDITIONAL INFORMATION:
        SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED
```

Figure 25-25 Additional Information display

The lines following this header are supplied by the component owning the couple data set, for example:

- SYSPLEX-ONLY COUPLE DATA SET SUPPORTED
  - The couple data set supports only the use of the sysplex couple data set itself.
- ALL TYPES OF COUPLE DATA SETS ARE SUPPORTED
  - The couple data set supports the use of additional couple data sets such as CFRM, SFM, and so on.

One or more of the following lines can also appear:

- GRS STAR MODE IS SUPPORTED
  - The couple data set supports GRS star mode operations.
- CLUSTER RESOURCE MANAGEMENT IS SUPPORTED
  - The couple data set supports cluster resource management operations.
- SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED
  - The couple data set supports the system status detection partitioning protocol.

25.4.3 New messages

A set of new messages comes along with this new support in z/OS V1R11:

```
IXC103I SYSTEM IDENTIFICATION INFORMATION information
```

The message displays the identification information about system status detection partitioning protocol associated with this system. This message is issued when the system performs initialization processing to become enabled to target other systems and be targeted by other systems using the system status detection partitioning protocol; see Figure 25-26 on page 538.

As part of the initialization process, the system establishes a logical application connection to BCPii to issue remote hardware management console API commands against other systems that are eligible targets of the system status detection partitioning protocol.

```
CONNECTION STATUS: {CONNECTED|NOT CONNECTED}
SYSTEM NAME: sysname
SYSTEM NUMBER: sysnum
IMAGE NAME: image
NODE DESCRIPTOR: type.mfg.plant.sequence
PARTITION NUMBER: partition
CPC ID: cpcid
NETWORK ADDRESS: netid.nau
IPL TOKEN: ipltoken
```

Figure 25-26 IXC103I SYSTEM IDENTIFICATION INFORMATION information

The following message displays the identification information about system status detection partitioning protocol associated with this system

```
IXC104I SSD PARTITIONING PROTOCOL ELIGIBILITY:information
```

In the message which indicates whether the SSD protocol is enabled on this system and its state, information is as shown in Figure 25-27

```
SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS.
[REASON: targetotherrsn]
SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
[REASON: othertargetrsn]
```

Figure 25-27 IXC104I SSD PARTITIONING PROTOCOL ELIGIBILITY information

```
SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS
```

This specifies whether this system can employ the system status detection partitioning protocol when removing other systems from the sysplex.

```
targetotherrsn
```

This indicates the reason why this system cannot use the system status detection partitioning protocol to aid in removing other systems from the sysplex. One of the reasons shown in Figure 25-28 applies.

```
SYSPLEX COUPLE DATA SET NOT FORMATTED TO SUPPORT PROTOCOL
NOT ENABLED BY INSTALLATION
OPERATING AS VM GUEST
BCPII SERVICES NOT AVAILABLE
SYSTEM OR HARDWARE ERROR
INSUFFICIENT SAF RESOURCE ACCESS AUTHORITY
UNEXPECTED SYSTEM SERVICE ERROR
PROTOCOL NOT APPLICABLE IN MONOPLEX MODE
PROTOCOL NOT APPLICABLE IN XCF-LOCAL MODE
```

Figure 25-28 Reason codes for targetotherrsn

```
SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS:
```

This specifies whether other systems can employ the system status detection partitioning protocol when removing this system from the sysplex. The `othertargetrsn` reason codes are shown in Figure 25-29 on page 539.

```
SYSPLEX COUPLE DATA SET NOT FORMATTED TO SUPPORT PROTOCOL
NOT ENABLED BY INSTALLATION
NOT SUPPORTED BY HARDWARE
OPERATING AS VM GUEST
SYSTEM OR HARDWARE ERROR
BCPII SERVICES NOT AVAILABLE
INSUFFICIENT SAF RESOURCE ACCESS AUTHORITY
UNEXPECTED SYSTEM SERVICE ERROR
PROTOCOL NOT APPLICABLE IN MONOPLEX MODE
PROTOCOL NOT APPLICABLE IN XCF-LOCAL MODE
```

Figure 25-29 Reason codes for system status detection partitioning protocol

```
IXC106I SYSTEM sysname status
```

This message indicates the system initiates partitioning to remove the named system from the sysplex. Partitioning might be able to bypass portions of the partitioning protocol because the named system is already known to have failed.

The status in the message can be one of the following:

- ENTERED WAIT STATE
  - The system was observed to have entered a non-restartable disabled wait state.
- RESET OR NEW IMAGE LOADED
  - The partition containing the system has been reset or a new system image has been loaded in the partition it formerly occupied.
- CPC FAILED
  - The central processing complex (CPC) on which the system was running has failed.
- PARTITION DEACTIVATED
  - The central processing complex (CPC) LPAR on which the system was running has been deactivated.

```
IXC107E SSD PARTITIONING PROTOCOL CONFIG IS NOT COMPLETE
```

Message IXC107E is issued if the system status detection partitioning protocol cannot be used. The system processes partitioning requests using a partitioning protocol based on the sysplex failure management (SFM) policy or default indeterminate status behavior. This message is DOMed when the exceptions causing the configuration to be incomplete are corrected.

One of the following conditions can exist:

- The BCPii address space and BCPii services are not available.
- The SYSSTATDETECT function is not enabled on the local system.
- The local system is not able to connect to the local CPC and local CPC image or a remote CPC and remote CPC image in the sysplex, although the local system is eligible and enabled to use the system status detection partitioning protocol.

An exception condition has been detected on the local system that prevents the system status partitioning protocol from being used to its fullest capability in the sysplex by the local system.

```
IXC108I SYSPLEX PARTITIONING INITIATING {FENCE|SYSTEM RESET}
```

When the IXC108I message is issued, as shown in Figure 25-30, it means that an action is being taken to ensure that a system being partitioned from the sysplex no longer has the capability to perform I/O to devices that can be used by another active system in the sysplex.

```
SYSTEM NAME: sysname
SYSTEM NUMBER: sysnum
[SYSTEM IDENTIFIER: sysid]
[IMAGE NAME: image]
[NETWORK ADDRESS: netid.nau]
[IPL TOKEN: ipltoken]
```

Figure 25-30 IXC108I SYSPLEX PARTITIONING INITIATING message

Message IXC109I

Message IXC1091 has the following text:

```
IXC109I {FENCE | SYSTEM RESET} OF SYSTEM sysname ...
```

The actual message is lengthy, as shown in Figure 25-31.

```
IXC109I {FENCE | SYSTEM RESET} OF SYSTEM sysname {SUCCESSFUL. | RESULTS: |
TIMED OUT.} [{HWICMD|HWMCA_EVENT_COMMAND_RESPONSE}
RETURN CODE: retcode] text
```

Figure 25-31 IXC109I FENCE or SYSTEM RESET OF SYSTEM sysname

IXC109I message explanation

Action was taken to ensure that a system being partitioned from the sysplex no longer has the capability to perform I/O to devices that can be used by another active system in the sysplex. Message IXC108I was issued when the action was initiated. This IXC109I message is issued when the results of the action are available or the action times out.

In the message text:

- FENCE - The action is system isolation by a system fence through Coupling Facility fencing services.
- SYSTEM RESET - The action is a SYSTEM RESET-NORMAL through the HWICMD BCPii callable service. sysname The name of the system targeted by the action.
- SUCCESSFUL. - The action completed successfully.
- RESULTS: - The action resulted in the indicated return code.
  - TIMED OUT. The action did not complete within the allotted time. Processing continues without the result of the action.
  - HWICMD The action resulted in the indicated HWICMD BCPii callable service return code.
  - HWMCA_EVENT_COMMAND_RESPONSE The action resulted in the indicated ENF68 command response return code. retcode The hexadecimal return code from the indicated source. text One of the following: blank No additional information is provided.
- (XCFAS DOES NOT HAVE SAF AUTHORIZATION) XCFAS does not have SAF authorization to RESET-NORMAL the target system.

System action

When the action is successful, partitioning can complete and message IXC105I is issued. When the action is not successful, operator intervention might be required to complete partitioning as indicated by message IXC102A being issued, or another action can be attempted as indicated by message IXC108I being issued. For example, a SYSTEM RESET can be attempted after a FENCE times out.

Message IXC111I

This message, shown in Figure 25-32, displays identification information associated with a system and the local system connection status to that system for the purposes of employing the system status detection partitioning protocol.

```
IXC111I LOGICAL PARTITION REMOTE CONNECTION INFO information
```

```
CONNECTION STATUS: {CONNECTED | NOT CONNECTED}
SYSTEM NAME: sysname
SYSTEM NUMBER: sysnum
IMAGE NAME: image
NETWORK ADDRESS: netid.nau
IPL TOKEN: ipltoken
DIAG INFO: diaginfo
```

Figure 25-32 IXC111I LOGICAL PARTITION REMOTE CONNECTION INFO information

In the message text:

- CONNECTED - The local system is connected to remote CPC image through BCPii callable services.
- NOT CONNECTED - The local system is not connected to remote CPC image through BCPii callable services. The local system cannot target remote CPC image with the system status detection partitioning protocol.
- diaginfo - The diagnostic information applicable when the connection status indicates that the local system is not connected to the remote system sysname and CPC image through BCPii callable services. The diagnostic data returned by the BCPii callable service can help determine the cause of the failed connection request. The diagnostic information contains the following data:
  - The name of the BCPii callable service that failed when the local system attempted to establish a connection to another CPC or image in the sysplex.
  - An error return code that was returned by the BCPii callable service that failed while the local system was attempting to establish a connection to a CPC or image. See *z/OS MVS Programming: Callable Services for HLL* for more information about BCPii services return codes and actions to take in the event of a specific return code.
  - A diagnostic data area returned by the BCPii callable service, which contains information related to the BCPii service that failed.

Message IXC112I

This message, shown in Figure 25-33 on page 542, displays identification information associated with a system and the local system connection status to that system for the purpose of employing the system status detection partitioning protocol.

```
IXC112I BCPII SYSTEM ERROR. BCPII CALLABLE SERVICE bcpiiservice
```

A BCPii callable service was issued on the local system by XCF in support of the system status detection partitioning protocol, but the BCPii callable service returned a failing return code.

```
REQUEST: request
RETURN CODE: retcode
DIAG CODE: diagcode
```

Figure 25-33 IXC112I BCPII SYSTEM ERROR. BCPII CALLABLE SERVICE bcpiiservice

In the message text:

- bcpiiservice is the name of the BCPii callable service that failed. bcpiiservice is one of the following services:
  - HWICONN BCPii HWICONN callable service
  - HWILIST BCPii HWILIST callable service
  - HWIQUERY BCPii HWIQUERY callable service
  - HWIEVENT BCPii HWIEVENT callable service
  - HWICMD BCPii HWICMD callable service
  - HWIDISC BCPii HWIDISC callable service request. This is the internal process that was in control at the time of the service failure.
- Request is one of the following processes:
  - LOCAL INITIALIZATION - Local system initialization to connect to the local CPC and CPC image through BCPii.
  - SYSTEM RESET - System reset processing against a remote target system.
  - DISCONNECT - Local system disconnect processing from the local CPC and CPC image and remote CPC and CPC images through BCPii.retcode The return code from the BCPii callable service that failed.
  - diagcode - Diagnostic data that was returned by the bcpiiservice to help determine the cause of the service failure.

Message IXC113I

This message, shown in Figure 25-34, indicates that the local system has released its connection to remote system image because either the system image has been removed from the sysplex or the SSD protocol has been disabled on the local system.

The local system can no longer target remote system image with system status detection partitioning protocol commands through BCPii.

```
IXC113I BCPII CONNECTION TO SYSTEM sysname RELEASED text
```

```
DISCONNECT REASON: discreason
IMAGE NAME: image
NETWORK ADDRESS: netaddr
SYSTEM NUMBER: sysnum
IPL TOKEN: ipltoken
```

Figure 25-34 IXC113I BCPII CONNECTION TO SYSTEM sysname RELEASED

As described, the local system has released its connection to remote system image using the BCPii HWIDISC callable service for reason *discreason*. The local system can no longer target remote system image with system status detection partitioning protocol commands through BCPii.

In the message text:

- sysname - This is the name of the remote system in the sysplex that the local system has released its virtual connection to.
- *discreason* - This is the reason why the local system released its BCPii connection to remote system sysname. One of the following lines is displayed:
  - SYSTEM IMAGE REMOVED FROM SYSPLEX - The system image has been removed from the sysplex because of a partition action taken against the system sysname.
  - SYSSTATDETECT DISABLED ON LOCAL SYSTEM - The system status detection partition protocol has been disabled on the local system by a SETXCF FUNCTIONS command. When the function is reenabled on the local system, it attempts to connect to other remote systems in the sysplex that are eligible targets of the system status detection partitioning protocol.

Message IXC114I

This message is shown in Figure 25-35. When the SSD partition protocol is enabled, XCF obtains status information for CPC and Image Name through BCPii APIs in response to a DOWN reply to IXC102A or IXC402D to determine if a SYSTEM RESET, LOAD or other acceptable alternative action was performed that results in the reset of system name.

```
IXC114I LOGICAL PARTITION REMOTE STATUS INFORMATION

SYSTEM NAME:  sysname
SYSTEM NUMBER: sysnum
CPC:  cpc
STATUS: OPERATING  |  NOT OPERATING |  NO POWER |  OTHER |  N/A
OPERSTAT: cpcstat
IMAGE NAME:  image  STATUS: OPERATING  |  NOT OPERATING |  NO POWER |  OTHER |
N/A   OPERSTAT:  imagestat
STORED IPL TOKEN:  ipltoken
RETURNED IPL TOKEN: ipltoken
DIAG INFO:
```

Figure 25-35 IXC114I LOGICAL PARTITION REMOTE STATUS INFORMATION message

The system continues processing. If it is determined that system name has not been RESET, then message IXC208I is issued stating that a DOWN reply was entered without system name being RESET.

If the returned status for Image is OPERATING, take appropriate action to system reset the system name and reply DOWN again to message IXC102A or IXC402D. See message IXC102A or IXC402D for acceptable actions to reset a system.

Message IXC373I

A COUPLExx parmlib member FUNCTIONS statement or a SETXCF FUNCTIONS command specified that one or more installation-controllable optional functions are to be enabled or disabled. This message reports the results of that request.

```
IXC373I XCF/XES OPT FUNCTIONS EN | DIS SYSSTATDETECT
```

25.4.4 Sysplex partitioning and system isolation considerations

The sysplex failure management (SFM) policy allows you to specify how a system is to respond to a Status Update Missing (SUM) condition. The system default is used when no action is specified for the system or when no SFM policy is active.

When the SSD protocol is enabled, the steps taken during sysplex partitioning and system isolation are affected and automatic action can be taken regardless of whether an SFM policy is active.

In this section we discuss the actions taken during sysplex partition and system isolation and the effect of having SSD protocol enabled.

Sysplex partitioning triggers system isolation

When a system is partitioned, for example as a result of an **V XCF,systemname,OFFLINE** command, the following processing occurs:

1. Sysplex partitioning is initiated - IXC101I.
2. The CLEANUP interval elapses.
3. Isolation is attempted - IXC108I and IXC109I.
4. If isolation is incomplete, then the IXC102A response is needed.
5. Sysplex partitioning is complete- IXC105I.

The system isolation will generate messages IXC108I and IXC109I. If isolation is successful, then sysplex partitioning will complete without issuing IXC102I. If isolation fails, then IXC102I will be issued and a response is required.

SSUM triggers system isolation

The following processing occurs:

1. Status updates stop.
2. FDI elapses and the system is declared SSUM.
3. ISI elapses.
4. A PR/SM SSUM policy action is attempted, if applicable.
5. Sysplex partitioning is initiated - IXC101I.
6. Isolation is attempted, if needed - IXC108I and IXC109I.
7. If isolation is incomplete, then the IXC102A response is needed.
8. Sysplex partitioning is complete- IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The partitioning reason in this case is SFM STARTED DUE TO STATUS UPDATE MISSING or POLICY INITIATED REQUEST if PR/SM SSUM action was successful.

Note: Sysplex partitioning does not wait the CLEANUP interval because it does not expect that the system being partitioned can perform cleanup when it is SSUM. The only time isolation is not needed for SSUM-triggered partitioning is when a PR/SM SSUM policy action is successful.

The operator must ensure that the system has been reset before responding to IXC102A.

SSUM triggers PROMPT

The following processing occurs:

1. Status updates stop.
2. FDI elapses and the system is declared SSUM.
3. OPNOTIFY elapses and the IXC402D response is needed.
4. Sysplex partitioning is initiated - IXC101I.
5. Sysplex partitioning is complete - IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The OPNOTIFY “timer” starts when the FDI “timer” starts, so IXC402D is issued after any OPNOTIFY time in excess of the FDI. The operator must ensure that the system has been reset before responding to IXC402D. The partitioning reason in this case is SYSTEM STATUS UPDATE MISSING.

Note: Sysplex partitioning does not wait the CLEANUP interval or the ISI, and isolation is not performed. This is because the operator indicated that the system is already reset.

SSD protocol triggers system isolation

The following processing occurs:

1. Status updates stop.
2. About 6 seconds elapse.
3. SSD protocol detects the system is demised and issues IXC106I.
4. A PR/SM SSUM policy action is attempted, if applicable.
5. Sysplex partitioning is initiated - IXC101I.
6. Isolation is attempted, if needed - IXC108I and IXC109I.
7. If isolation is incomplete, then the IXC102A response is needed.
8. Sysplex partitioning is complete - IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The SSD protocol checks the system’s heartbeat every 3 seconds until status updates resume, the system is reset, or a third system is monitoring partitioning.

With SSD enabled:

- Sysplex partitioning does not wait the FDI because a “failure” has definitely been detected.
- Sysplex partitioning does not wait the ISI because status has been determined.
- Sysplex partitioning does not wait the CLEANUP interval because a system that is demised cannot perform cleanup.

The operator must ensure the system has been reset before responding to IXC102A. System isolation is only needed when the system is demised due to a wait state and PR/SM SSUM action did not request RESET/DEACTIVATE.

ISOLATETIME(0) default

There are two types of default for ISOLATETIME:

- The default for ISOLATETIME taken *against* the local system by other systems.
- The default for ISOLATETIME taken *by* the local system against other systems.

Without the SSD protocol enabled, both defaults for ISOLATETIME are PROMPT. With the SSD protocol enabled, both are calculated. The default is always PROMPT for systems without the SSD protocol enabled.

If a PR/SM SSUM policy action is being used and fails, XCF will then try the default action. Without this support, OPNOTIFY processing still goes on in the background when a PR/SM SSUM policy action is taking place. With SSD protocol enabled, if the fallback default action is the new default of ISOLATETIME(0), then IXC402D will not be issued and partitioning will be initiated if the PR/SM SSUM policy action fails. The result is equivalent to ISOLATETIME(0) being used instead of RESETTIME/DEACTTIME.

Note: When an SFM policy is not defined and SSD is enabled, the default action is equivalent to ISOLATE(0) rather than PROMPT.

A specification of SSUMLIMIT for a system without specifying ISOLATETIME, RESETTIME, or DEACTTIME previously resulted in an error. It is now allowed and results in a default of ISOLATETIME(0).

- If a downlevel system uses a policy defined by an IXCMIAPU with the SSD protocol enabled, then PROMPT will be taken as the default instead of ISOLATETIME(0).

  A specification of SSUMLIMIT without specifying ISOLATETIME, RESETTIME, or DEACTTIME will be ignored.
- If a system with the SSD protocol enabled uses a policy defined by a downlevel IXCMIAPU, then ISOLATETIME(0) SSUMLIMIT(NONE) can be taken as the default instead of PROMPT.

Note: Each system decides what default to use. Thus, an R11 system with the SSD protocol enabled will determine a default for ISOLATETIME that is different from an R10 system.

Displaying the SSUM values

The **D XCF,C** command output indicates the effective values for SSUM ACTION, SSUM INTERVAL, and SSUM LIMIT when an SFM policy is active on all systems.

- Without the SSD protocol enabled, **D XCF,C** output indicates N/A for these values when an SFM policy is not active on all systems
- With the SSD protocol enabled, **D XCF,C** output more often indicates the effective values expected to be used.

Only the following conditions result in an N/A display:

- There is no other active system in the sysplex.
- A mixture of z/OS V1R11 and lower level systems are active in the sysplex.
- A PR/SM policy is active on a system.

In effect, the SSUM values are displayed when the expected effective values apply and can be reasonably determined. N/A is displayed for mixed systems because an R11 can isolate but an R10 can PROMPT.

System isolation for sysplex partitioning

System isolation is performed when XCF has not been ensured that the system being partitioned has stopped I/O to all shared devices. Without the SSD protocol enabled, XCF only attempts to isolate a system (using fencing services) when an SFM policy is active.

With the SSD protocol enabled, regardless of the SFM policy, XCF attempts to fence whenever the system being partitioned stored its fence authority in a CF. System isolation is no longer tied to an SFM policy, so a system can be isolated regardless of whether there is an SFM policy.

XCF attempts to perform a SYSTEM RESET using SSD Protocol (BCPii) whenever fencing did not isolate and the system being partitioned self-identified its image, CPC, or ipltoken and the SSD protocol is enabled for systems.

System isolation is not performed or required when a system is demised due to one of the following conditions:

- SYSTEM RESET OR NEW IMAGE LOADED
- PARTITION DEACTIVATED
- CPC FAILURE

System Status Detection Partitioning Protocol and BCPii

The System Status Detection (SSD) Partitioning Protocol takes advantage of BCPii interfaces to use z/Series hardware services to discover the status of failed systems in a sysplex during sysplex recovery processing. Using BCPii, XCF can bypass specified intervals such as the failure detection interval (FDI), the SSUM ACTION interval, and the cleanup interval. It also can avoid fencing processing and manual intervention requirements to shorten sysplex recovery processing time.

To enable XCF to use the SSD partitioning protocol and BCPii in the sysplex, meet the following requirements:

- The XCF address space must have adequate authorization through the z/OS Security Server (RACF) or an equivalent security product to access BCPii defined FACILITY class resources. See “Assigning the RACF TRUSTED attribute” in *z/OS MVS Initialization and Tuning Reference* for information about using RACF to assign the TRUSTED attribute to the XCF address space.
- The BCPii interfaces to invoke z/Series Hardware APIs must be enabled. See “BCPii Setup and Installation” in *z/OS MVS Programming: Callable Services for High-Level Languages* for information about installation and configuration steps and SAF authorization requirements to enable BCPii to invoke z/Series Hardware APIs.
- All systems in the sysplex must operate on z10 EC GA2, z10 BC GA1, or newer hardware. They must also be at z/OS V1R11 or higher to take full advantage of the functionality associated with the system status detection partitioning protocol. At a minimum, all systems in the sysplex must be at z/OS V1R11 or have installed toleration PTFs for OA26037 on V1R10 and V1R9 systems in the sysplex to use a sysplex couple data set formatted using the ITEM NAME(SSTATDET) NUMBER(1) control statement, which is required by the protocol.
- A system running on z/OS V1R11 (or higher) and on the requisite level of hardware and enabled to use the SSD partitioning protocol can exploit the full functionality of the SSD partitioning protocol. “Full functional exploitation” means that a system can be targeted by other systems in the sysplex enabled to use the SSD partitioning protocol for sysplex recovery, and that the system can target other systems running on z/OS V1R11 (or higher) and the requisite level of hardware in the sysplex to expedite sysplex recovery when the targeted system of recovery processing becomes demised.
- A system running on z/OS V1R11 (or higher) and downlevel hardware is only eligible to target other systems that are enabled to exploit the full functionality of the SSD partitioning protocol. A system not running on the requisite hardware cannot be the target of SSD partitioning protocol functions. A system running on downlevel hardware, a downlevel

version of z/OS, or running as a VM Guest cannot be enabled to use the SSD partitioning protocol.

- The sysplex couple data set must be formatted using the ITEM NAME(SSTATDET) NUMBER(1) control statement.
- The SYSSTATDETECT function must be enabled. By default, the SYSSTATDETECT function is enabled. The current setting of theSYSSTATDETECT function can be determined by issuing a DISPLAYXCF,COUPLE command.

System fencing messages

New messages IXC108I and ICXC109I are written to syslog by the partitioning monitor system. After IXC108I is issued for fencing, all systems in the sysplex attempt to fence the system. If a system is successful, the fence will be successful. Otherwise the fence times out after 32 seconds. Each system cuts a symptom record to log the results of the fence attempt. If fencing times out, RESET can then be attempted or IXC102A will be issued.

System reset messages

New messages IXC108I and IXC109I are written to syslog by the partitioning monitor system. After IXC108I is issued for system reset, only the system that issued the IXC108I (the partitioning monitor) attempts to reset the system. If results are provided, then IXC109I is issued with the results. Otherwise, the reset times out after 30 seconds. A successful reset returns HWMCA_EVENT_COMMAND_RESPONSE RETURN CODE 0.

For details about return codes from a BCPii callable service, see *z/OS MVS Programming: Callable Services for HLL*. For details about return codes from an ENF 68 command response, see *System z Application Programming Interfaces,* SB10-7030.

If the reset is not successful, IXC102A will be issued. Only one system attempts the reset; however, every system attempts the isolate.

Important: The HWICMD return code is '00000F02'X (3842) if XCFAS does not have the required SAF authorization.

Verification when operator indicates system reset

Operators are responsible for verifying that a system can no longer perform I/O to shared devices before replying DOWN to IXC402D or IXC102A:

```
IXC402D sysname LAST OPERATIVE AT hh:mm:ss. REPLY DOWN AFTER SYSTEM RESET, OR
INTERVAL=sec TO SET A REPROMPT TIME.
IXC102A XCF IS WAITING FOR SYSTEM sysname DEACTIVATION. REPLY DOWN WHEN MVS ON
sysname HAS BEEN SYSTEM RESET.
```

Operator response

Replying SYSNAME=sysname1,DOWN to IXC409D:

```
IXC409D SIGNAL PATHS BETWEEN sysname1 AND sysname2 ARE LOST. REPLY RETRY OR
SYSNAME=SYSNAME OF THE SYSTEM TO BE REMOVED.
```

A system must also be reset prior to issuing a **VARY XCF,sysname,OFFLINE,FORCE** command. On many occasions, an operator has not verified that the system was reset before performing these actions whose prerequisite is a system reset. If the verification is inconclusive, XCF has no choice but to trust the operator as it has always done.

With the SSD protocol enabled, XCF can verify that the system has been reset and thereby avoid operator errors that can cause data integrity problems. A response to IXC102A,

IXC402D, or IXC409D is not required if SSD Protocol is working. This means that getting an XC102A, IXC402D, or IXC409D with the SSD protocol enabled indicates that there was a problem resetting the system.

Replying DOWN to IXC102A, IXC402D, or IXC409D with the SSD protocol enabled can result in one form of the message ICXC208I being displayed; see Figure 25-36.

```
IXC208I THE RESPONSE TO MESSAGE msgnum IS INCORRECT: DOWN REPLY ENTERED WITHOUT
SYSTEM RESET

IXC208I THE RESPONSE TO MESSAGE IXC409D IS INCORRECT: SYSNAME=sysname1,DOWN
REPLY ENTERED WITHOUT SYSTEM RESET
```

Figure 25-36 IXC208I issued with the SSD protocol enabled

The SSD protocol can result in the **VARY XCF,sysname,OFFLINE,FORCE** command being rejected because the system was not reset. Message IXC370I is issued as shown in Figure 25-37.

```
IXC370I THE VARY XCF COMMAND COULD NOT BE PROCESSED: FORCE OPTION USED WITHOUT
SYSTEM RESET
```

Figure 25-37 IXC370I issued with the SSD protocol enabled

Tip: When IXC 208I or IXC370I is issued, you can bypass SSD to allow a reply of DOWN to be accepted by IXC102A, IXC402D, or IXC409D, or to allow the FORCE action to be taken.

SSD can be bypassed by using the following command:

```
SETXCF FUNCTIONS,DISABLE=SYSSTATDETECT
```

IXCL1DSU couple data set format utility

To have a CDS properly set for this new function, add the ITEM NAME(SSTATDET) NUMBER(1) line to the utility input, as shown in Figure 25-38 on page 549, to format a sysplex couple data set with the larger system records required by the system status detection partitioning protocol.

```
DEFINEDS
SYSPLEX(sysplexname)
MAXSYSTEM(maxsystems)
DSN(dsname) [VOLSER(volser)] [UNIT(unit)]
[CATALOG | NOCATALOG]
[STORCLAS(storclass)] [MGMTCLAS(mgmtclass)]
DATA TYPE(SYSPLEX)
[ITEM NAME(GROUP) NUMBER(groupnum)]
[ITEM NAME(MEMBER) NUMBER(memnum)]
[ITEM NAME(GRS) NUMBER(1)]
[ITEM NAME(CLUSTER) NUMBER(1)]
[ITEM NAME(SSTATDET) NUMBER(1)]
```

Figure 25-38 ITEM NAME(SSTATDET) NUMBER(1) line added

25.4.5 Interactions and dependencies

Because the enhanced partitioning depends on the availability of each system's addressing information to every other system in the sysplex, no system can apply the SSD partitioning algorithms (bypass FDI or cleanup interval, initiate remote reset, and so on) to any other system unless the sysplex is operating with sysplex CDSes containing the expanded system record. The addressing information is also necessary for correlating systems with the CPC on which each resides, so even the case in which a CPC is observed to have checkstopped requires the larger CDS.

The expansion of the system record is an incompatible change, therefore support is required to accommodate it. Systems running z/OS V1R11 understand the new, larger record. A toleration PTF is provided at lower releases (down to z/OS V1R9) to enable downlevel systems to process the larger record and higher version number.

Formatting a sysplex couple data set to support this protocol increments the version number associated with the couple data set. Such a couple data set cannot be brought into service if there are any systems active in the sysplex that do not have the code (either z/OS V1R11 or the toleration PTF) to accommodate the higher version number. Similarly, after such a couple data set is in use, no system without the code to accommodate it can join the sysplex. (As always, after a couple data set formatted with larger records is brought into service, it requires a sysplex-wide IPL to revert to a less-capable couple data set. However, there is no reason to do so, because the installation controls permit disabling the SSD partitioning protocol if necessary, without reverting to the older CDS format.)

Even though those downlevel systems are not able to actually use the SSTATDETECT protocols, this allows the z/OS V1R11 systems in the sysplex to use the new protocols, as the z/OS V1R9 and z/OS V1R10 systems peacefully share the CDS.

To sum up the implications of this new CDS format:

- SYSSTATDETECT PROTOCOL requires a SYSPLEX CDS formatted with SSTATDET.
- Support is available in z/OS V1R11.
- Toleration PTFs for APAR OA26037 are available on z/OS V1R9 and z/OS V1R10, to allow downlevel systems in the sysplex to process the higher version sysplex CDS.
- This allows the z/OS V1R9 and V1R10 systems to use the new SSD protocols.

Otherwise, BCPii requires the following:

- BCPii must be properly configured and set active.
- Control governing the new function must be properly set; see Figure 25-39.

```
SETXCF FUNCTIONS ENABLE SYSSTATDETECT
```

Figure 25-39 Control governing SSD

By default, the SSD function is enabled as soon as the necessary environmental requirements are met.

25.4.6 Hardware dependencies

Each system must set an IPL token that uniquely identifies it. The ability to store and reset the IPL token is introduced with z10 GA2, which means that systems not running on that

hardware cannot establish a token. Any such system is not eligible to be targeted by the remote reset command.

Moreover, although query commands can be directed to systems on downlevel hardware, XCF relies on observing changes in the IPL token to determine whether a system can be partitioned. Therefore, systems running on pre-GA2 hardware cannot be targeted by the SSD partitioning processing.

The support provided in this release will not be completely effective unless all systems on the sysplex are operating on z10 GA2 hardware with the following MCLs:

- MCL129 in EC Stream N10970
- Driver-76 HMC, MCL046 in EC Stream N10976

The z10 GA2 hardware provides new information when a resident image enters a wait state.

The support provided by this new function is not completely effective unless all systems on the sysplex are operating on z10 GA2 hardware. The system status detection partitioning protocol automatically limits its actions when the hardware support is not present, so no additional support is needed to disable the new function in the absence of z10 GA2 hardware.

Applicability to z/VM systems

The BCPii command interfaces, including queries directed to individual images, operate against an individual partition. Under VM, there can be multiple z/OS guests running in a single partition. BCPii therefore does not support the VM environment and does not start in a z/OS image running as a guest under VM.

So SSD partitioning protocol does not apply to VM-guest sysplexes, as BCPii is not for now recursively virtualizable as mentioned in “Activate BCPii” on page 520.

25.4.7 Migration and coexistence

Toleration support of the SSD partitioning support eliminates the requirement that the entire sysplex must consist of systems at or above z/OS V1R11 to use it. The toleration PTF does not roll down any part of the SSD partitioning function. Its sole purpose is to allow a mixed sysplex to bring a version 5 sysplex CDS into service.

New functions, particularly those that are potentially disruptive, are shipped with switches to permit an installation to disable the function if desired. This sysplex partitioning enhancements line item also includes an enabling switch. Installation control is by means of the COUPLExx OPTIONS statement in SYS1.PARMLIB(COUPLExx)” or the **SETXCF OPTIONS** command.

Because explicit action is required to bring version 5 sysplex CDSes into service, it is not necessary to require the installation to take additional action to enable the new support using these installation controls. By default, the SSD function is enabled as soon as the necessary environmental requirements are met. However, an installation can choose to disable it in the event of a defect, or for diagnostic purposes, or from an excess of caution. The setting of the switch on a given system governs both that system’s use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems.

SYSPLEX CDS

SSD protocol requires a SYSPLEX CDS formatted with SSTATDET. There is a toleration PTF for APAR OA26037 provided enabling systems below z/OS V1R11 to operate with SYSPLEX CDS formatted to support System Status Detection Partitioning Protocol (SSTATDET).

Downlevel (below z/OS V1R11) systems are not able to participate in the SSD protocols.

SFM policy

Specify (or default to) ISOLATETIME(0) to allow SFM to fence and partition a failed system without operator intervention and without undue delay.

If a system enters a status update missing condition and there is no active SFM policy, the monitoring system takes the system default against the failed system. This means if the monitoring system is a pre-z/OS V1R11 system, it uses the old system default and prompts the operator. If the monitoring system is a z/OS V1R11 system, it uses the new system default and isolates the failed system. The **`D XCF,C`** command shows the SFM action the system expects, but the monitoring system can use another action if no SFM policy is defined.

The sysplex failure management (SFM) policy allows you to specify how a system is to respond to a Status Update Missing condition. The system default is used when no action is specified for the system or when no SFM policy is active, as follows:

- Prior to z/OS V1R11, the system default action is PROMPT, which prompts the operator when a system enters a status update missing condition.
- As of z/OS V1R11, the system default action has changed to ISOLATETIME(0), which allows other systems to take immediate action to isolate the failed system.
  - If manual intervention is still required when a system goes status update missing you have to set up a SFM Policy to specify PROMPT.

Prerequisites for installation

Here are the prerequisites for installation:

- Enable the use of a SYSPLEX CDS that supports the SSD protocol on all systems in the sysplex.
  - z/OS V1R11 systems support a Sysplex CDS that is formatted for SSD Protocol.
  - Install the toleration PTF on all lower level systems.
- Format and bring into service a SYSPLEX CDS that supports the SSD protocol (primary and alternate).
- Migrate CECs to z10 GA2.
- The new function is not completely effective until all systems in the sysplex reside on z10 GA2 hardware.
- SFM Policy
  - Installations to continue to define and activate an SFM policy and establish failure detection and cleanup intervals as they do today.
    - The SFM policy, in conjunction with the FDI, governs the system's behavior if the new support fails to operate due to environmental conditions, problems with underlying services, or other factors.

26

IBM Health Checker for z/OS

This chapter describes the new health checks introduced with z/OS V1R11. The Health Checker provides a mechanism to check the health of an active system. It also can help you check for and proactively resolve potential problems before application availability is impacted or, in worst cases, before system or sysplex-wide 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.

The new health checks are designed to help check the health of active systems. Use those checks to implement the latest IBM hints, tips, and best practices. The output from these checks is useful for proactively resolving problems before they occur.

This chapter discusses the following checks added in z/OS V1R11:

- DFSMS Catalog check
- ITDS check
- SDSF check
- XCF check
- RTM check
- JES2 check

26.1 Using the new IBM Health Checks

IBM Health Checks provide better systems management and availability benefits when best practices are followed. Therefore, use those checks and download the functional PSP bucket HCHECKER as described in Chapter 10, “Migration health checks” on page 231.

Make sure you have received the APAR listed here:

Migration Health Check name LDAP_USE_TDS

APAR number OA27843

Available releases z/OS V1R8 and higher

Some checks have been updated in z/OS V1R11; you must review the following checks:

- CSV_LNKLST_SPACE
- HSM_CDSB_BACKUP_COPIES
- HSM_CDSB_DASD_BACKUPS
- HSM_CDSB_VALID_BACKUPS
- USS_PARMLIB
- USS_PARMLIB_MOUNTS
- XCF_FDI

26.2 New DFSMS catalog health check

The CATALOG_IMBED_REPLICATE check detects instances of the obsolete IMBED and REPLICATE attributes for user and master catalogs. This check avoids unplanned outages caused by the attributes.

No supported releases of z/OS honor the IMBED or REPLICATE attributes for new catalogs; they are obsoleted by newer, cached DASD devices. Now if a new catalog is defined with these attributes, they are simply ignored on the define and the catalog is defined without them.

The user is notified of any catalogs with IMBED or REPLICATE attributes in their environments. The check has the following attributes:

```
SEVERITY(LOW) - INTERVAL(TIMER) - HOURS(24)
```

When the z/OS Health Check address space is activated, **START HHZPROC**, the CATALOG_IMBED_REPLICATE check, runs automatically.

Check messages

If no catalogs are detected by the health check, then an informational message is displayed on the console as shown in Figure 26-1.

```
IGGHC103I  CHECK(IBMCATALOG,CATALOG_IMBED_REPLICATE) ran successfully and found
no exceptions.
```

Figure 26-1 CATALOG_IMBED_REPLICATE check informational message

If one or more catalogs are detected by the check with IMBED/REPLICATE attributes, then the exception message shown in Figure 26-2 is displayed on the console.

```
HZS0001I CHECK(IBMCATALOG,CATALOG_IMBED_REPLICATE):  440
IGGHC104E The CATALOG_IMBED_REPLICATE check has detected one or more
catalogs defined with the IMBED and/or REPLICATE attributes.
```

Figure 26-2 CATALOG_IMBED_REPLICATE check exception message

After message IGGHC104E is issued, a report is generated in SDSF, which you can view to identify the affected catalogs.

The report consists of four columns:

- Column 1 identifies the name of the affected catalog.
- Column 2 identifies the attribute (IMBED or REPLICATE).
- Column 3 identifies the catalog component (DATA or INDEX) with the attribute.
- Column 4 identifies whether the catalog was available for processing.

The CATALOG_IMBED_REPLICATE report is displayed in Figure 26-3.

```
CHECK(IBMCATALOG,CATALOG_IMBED_REPLICATE)
START TIME: 05/14/2009 11:20:19.566157
CHECK DATE: 20080619  CHECK SEVERITY: LOW

IGGHC106I
Following catalog(s) were inspected by the health check

------------------------------------------------------------------------
 CATALOG NAME                                ATTRIBUTE  COMP AVAILABILITY
------------------------------------------------------------------------
CATALOG.DB2ICF2.VTOTCAT                       IMBED     DATA  YES
CATALOG.DB2ICF2.VTOTCAT                       IMBED     INDEX YES
CATALOG.DB2ICF2.VTOTCAT                       REPLICATE INDEX YES
CATALOG.HCD.USERCAT                           IMBED     DATA  YES
CATALOG.HCD.USERCAT                           IMBED     INDEX YES
CATALOG.SHRICF1.VIODFPK                       IMBED     DATA  YES
CATALOG.SHRICF1.VIODFPK                       IMBED     INDEX YES
CATALOG.TOTICFM.VTOTCAT                       IMBED     DATA  YES
CATALOG.TOTICFM.VTOTCAT                       IMBED     INDEX YES
CATALOG.TOTICFM.VTOTCAT                       REPLICATE INDEX YES
MCAT.OS3RSA.VOS3CAT                           IMBED     DATA  YES
MCAT.OS3RSA.VOS3CAT                           IMBED     INDEX YES
MCAT.OS3RSA.VOS3CAT                           REPLICATE INDEX YES
MCAT.OS3R3V01.VOS3CAT                         IMBED     DATA  YES
MCAT.OS3R3V01.VOS3CAT                         IMBED     INDEX YES
```

Figure 26-3 CATALOG_IMBED_REPLICATE report

At the end of the report a summary of the check is displayed, as shown in Figure 26-4 on page 556.

```
IGGHC109I IMBED/REPLICATE HEALTH CHECK SUMMARY REPORT:

A total of 84 catalogs were inspected by the Health Check

60 catalogs did not have either IMBED or REPLICATE attributes on them.

4 catalogs were not processed by the Health Check either because the
catalog volume was not available or the catalog entry was in error.

20 catalogs were detected with either IMBED and/or REPLICATE attributes
on them.

* Low Severity Exception *
```

Figure 26-4 CATALOG_IMBED_REPLICATE check summary report

If the check encounters an error during catalog search interface processing, an informational message is displayed and the check is disabled until the error is resolved; see Figure 26-5.

```
IGGHC107I CSI has encountered an error in its processing while attempting to
return data from the Catalog Address Space. Check message HZS1002E for
diagnostic information. The check is disabled.
```

Figure 26-5 CATALOG_IMBED_REPLICATE message error

If the check found no user catalog connector in the master catalog, then an informational message is displayed; see Figure 26-6.

```
IGGHC108I No user catalog connector entries were found in the master catalog
```

Figure 26-6 CATALOG_IMBED_REPLICATE informational message

26.3 New IBM Tivoli Directory Server for z/OS check

The LDAP_USE_TDS check detects the use of the Integrated Security Services LDAP server and states the need to migrate to IBM Tivoli Directory Server for z/OS. The Integrated Security Services LDAP server is not available in z/OS R11. Migration to IBM Tivoli Directory Server for z/OS is required.

The following parameters are accepted when specifying (PARM('xxx')):

PARM(ALL)	Exception messages will be issued.
PARM('NEW(text value)')	Exception messages will only be issued for new ISS LDAP servers that are introduced to the system after the NEW(text value) is set. These ISS LDAP servers will continue to be reported each time this check is run, until text value is set to a new value.
PARM(CLEAR)	Clear all previously detected ISS LDAP servers from memory.

The check has the following attributes:

```
SEVERITY(LOW) - INTERVAL(ONETIME)
```

This check applies to z/OS V1R9 and later with APAR OA27843.

Note: This check requires an HZSPDATA data set to store persistent data.

26.4 New SDSF check

The SDSF_CLASS_SDSF_ACTIVE check determines whether the SAF class SDSF is active. Although SDSF uses several SAF classes when verifying user access to profiles, the SDSF class is used for SDSF internal functions such as mapping a user to an SDSF group. SDSF SAF-based security is more dynamic and flexible than using ISFPARMS for security.

This check is automatically registered with the Health Checker when the SDSF server address space started. The SDSF server address space is optional. If you do not run the server, it is necessary to update your PROGxx with an EXIT statement to define the SDSF health check as a dynamic exit. SDSF uses RACROUTE REQUEST=STAT to determine whether the class is active.

The check has the following attributes:

```
SEVERITY(LOW) - INTERVAL(ONETIME)
```

This check applies to z/OS V1R9 and later.

26.5 New XCF check

There is one new XCF check:

```
Check(XCF_SYSSTATDET_PARTITIONING)
```

XCF_SYSSTATDET_PARTITIONING checks whether the System Status Detection (SSD) partitioning protocol is enabled by the user. If it is not enabled, the check reports the environmental factors that prevent this system from using the SSD protocol.

The factors checked include:

- Availability of sysplex couple data sets formatted to support the protocol
- The setting of installation controls governing the use of the protocol

Enable SSD partitioning protocol to ensure that failed systems are removed from the sysplex expeditiously and with a minimum of operator involvement.

The check has the following attributes:

```
SEVERITY(MED) - INTERVAL(004:00) - PARM: ENABLED(YES)
```

When an exception is found, either the IXCH0526E or IXCH0529E message is issued confirming the environment configuration, as shown in Figure 26-7 on page 558.

```
IXCH0526E The System Status Detection (SSD) partitioning protocol is
not configured for use by XCF on the system. This is inconsistent
with the owner specification.
```

Figure 26-7 XCF_SYSSTATDET_PARTITIONING exception message

26.6 New recovery terminator manager heath check

A new health check known as RTM_IEAVTRML for the recovery termination manager (RTM) warns when IEAVTRML is still being used and provides a mechanism for accepting legitimate usage situations.

End of Task (EOT) and End of Memory (EOM) resource managers defined to RTM by using an IEAVTRML run for every task and address space termination in the system, possibly impacting system performance. For that reason the best practice method of defining a resource manager is by using the RESMGR service. This allows resource managers to be defined for specific tasks or address spaces, as well as for all tasks or address spaces.

Note: Use the RESMGR service instead of IEAVTRML.

The RTM_IEAVTRML check runs automatically every time that the IBM Health Checker is started. To invoke RTM_IEAVTRML manually, enter the following command while the Health Checker is active:

```
f hzsproc,run,check(ibmrtm,rtm_ieavtrml)
```

Like other Health Checks, RTM_IEAVTRML can also be invoked by using the CK panel under SDSF.

The check has the following attributes:

```
SEVERITY(LOW) - INTERVAL(ONETIME) - PARM: PARM('ALL')
```

The accepted PARM options parameter are:

PARM('ALL') This specifies that exceptions are to be issued for all module names specified in IEAVTRML. This is the system default.

PARM('NEW(value)') This specifies that the current contents of IEAVTRML are to be treated as correct and exceptions are to be issued only for new module names added to IEAVTRML after this time. This specification persists across restarts of the Health Checker, including IPLs. Note that the system only recognizes changes to IEAVTRML by using an IPL with CLPA.

If this health check is successful (that is, it finds no exceptions), it will issue the message in the Health Checker message log shown in Figure 26-8.

```
IEAVTRH01I CHECK(IBMRTM,RTM_IEAVTRML) was successful.  IEAVTRML contains no
module names.
```

Figure 26-8 RTM_IEAVTRML successful message check

If module names in IEAVTRML have been accepted, it will issue the message shown in Figure 26-9.

```
EAVTRH02I CHECK(IBMRTM,RTM_IEAVTRML) was successful.  IEAVTRML contains no new
module names.
```

Figure 26-9 RTM_IEAVTRML successful add module names message check

If an exception was found, the Health Checker message log will then contain a message indicating up to 16 names, as shown in Figure 26-10.

```
IEAVTRH03I The following module names in IEAVTRML caused an exception:
   MVPTTRML DFSMRCLO
```

Figure 26-10 RTM_IEAVTRML exception message

Check considerations

If a resource manager module name is still required in IEAVTRML, use the check parameter NEW(value) to set the check to accept any current module names and to only flag future additions to IEAVTRML as exceptions. The NEW(value) parameter and the resource manager module names that it specifies as acceptable persist across restarts of this check including across IPLs without CLPA.

26.7 New JES2 health check

In z/OS V1R11, JES2 has added its first health check. The JES2_Z11_UPGRADE_CK_JES2 check determines whether the system is ready to upgrade the JES2 checkpoint to z11 level. If the necessary preconditions are already satisfied on the client's system, then a message is issued recommending that the system be upgraded to z11 mode. If the prerequisite conditions have not been satisfied, then an exception message is issued indicating that the system is not ready to upgrade to checkpoint level z11.

Note: Use JES2 z11 mode.

The check has the following attributes:

```
SEVERITY(LOW) - INTERVAL(168:00)
```

If all preconditions are satisfied, then message displayed in Figure 26-11 is shown on syslog.

```
HASPH022E JES2 address space JESA can activate
checkpoint mode z11.
Explanation:
All requirements for activating to checkpoint mode
z11 have been satisfied.  The checkpoints are large
enough and there are enough free BERTs.  LARGEDS
support is activated and all MAS members are at z/OS
release 1.11.
```

Figure 26-11 JES2_Z11_UPGRADE_CK_JES2 successful message

If an exception is found, the JES2_Z11_UPGRADE_CK_JES2 check displays either the HASP022E or HASP028E exception message, as shown in Figure 26-12.

```
HASPH028E
JES2 address space JES2 can not activate checkpoint mode z11.

  Explanation:  One or more requirements for activating to checkpoint
    mode z11 have not been satisfied. Refer to the additional messages
    following this message to determine the specific requirements that
    have not been satisfied.
```

Figure 26-12 JES2_Z11_UPGRADE_CK_JES2 exceptional message

At the end of the JES2_Z11_UPGRADE_CK_JES2 report is a summary of messages detailing why the address space cannot activate to z11; Figure 26-13 shows an example.

```
HASPH025I
JES2 address space JES2 requires the
activation of LARGEDS support.
```

Figure 26-13 JES2 Z11_UPGRADE_CK_JES2 summary

27

Server Time Protocol alerts

This chapter describes an enhancement provided in z/OS V1R11 that alerts operators to various hardware malfunctions of the Server Time Protocol (STP) network. These STP alerts are delivered by two new messages.

Additionally, the chapter describes new HMC API commands that provide new ways of performing the following tasks:

- Automating STP network management
- Starting to fill the gap between Parallel Sysplex and the STP network

27.1 STP alerts overview

The reason for this enhancement is because whenever an External Time Source (ETS) fails, the only place an error is logged with Server Time Protocol (STP) is in the hardware log through the HMC console. This significantly delays the notification of operators of an error condition, and delays problem determination and correction.

No z/OS error message is generated whenever an ETS failure occurs in STP-ONLY environments, thereby significantly delaying problem determination and correction.

As shown in Figure 27-1, external interrupt presents different external damage code, and more specifically with:

Bit 26 CTN Configuration Change (CC - When 1, it indicates that a CTN-configuration-change has occurred in the CTN.)

The conditions which cause the CTN-configuration-change condition to be generated include:

- A new node with a matching CTN ID has attached to this node.
- An attached node that had a matching CTN ID is no longer attached to this node
- The CTN ID for this node has changed.
- The stratum level of this node has changed, or a change in the stratum-1 or alternate-stratum-1 for the network has changed.
- The STP active or maximum version number has changed.

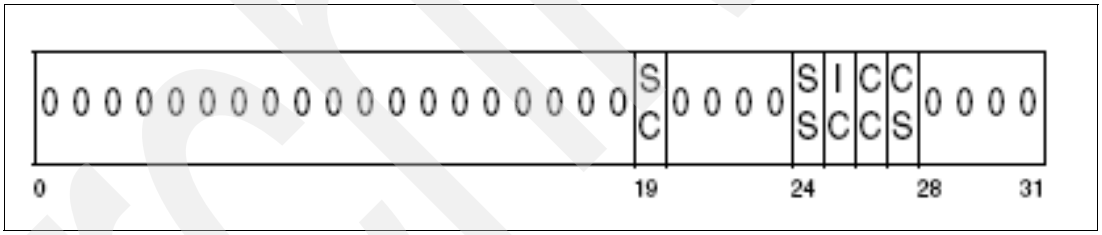


Figure 27-1 External damage code

27.2 z/OS V1R11 solution

The machine-check condition is generated for all CPUs in the configuration, and the condition for a CPU is cleared by the interruption taken by the CPU.

The z/OS V1R11 solution to this situation is a message that is issued when an STP alert is raised. The message indicates the event which caused the interruption. Improved system management and RAS of STP ETS functions by notifying the z/OS operator of failures helps problem handling, which can be investigated and corrected more quickly.

External interrupt handler

The z/OS V1R11 new support is invoked by the external interrupt handler when one of the new subclass of external interrupts is received.

This triggers the issuance of a new IEA031I console message which indicates the event that caused the STP alert interruption:

```
IEA031I STP ALERT RECEIVED. ALERT CODE= 06
```

For instance, in Figure 27-1 on page 562, alert code 06 means:

- NTP server failure - the Coordinated Timing Network (CTN) is in NTP or PPS mode and the console was not able to access any usable NTP server.
- The various other possible codes are detailed in 27.3, “STP alert event codes” on page 563.

A new subclass of STP Timing Status Change External Interrupts will be presented by hardware, which includes the following:

- All represent ETS state changes, some of which are failures.
- They are presented only when the machine is in STP-only timing mode.

New data fields identifying the new interrupt events are returned by the CHSC Store STP Information (SSTPI) command. These interrupts are then handled by the STP external SLIH.

A new message is also added to handle the case of the Preferred or Backup Servers located on stand-alone CFs; see Figure 27-2.

```
IEA395I THE CURRENT TIME SERVER HAS CHANGED TO THE cc
```

Figure 27-2 CTS switches to a separate server

Note that cc is PREFERRED or BACKUP when the Current Time Server switches to a separate server.

Hardware dependencies and PTFs

This new z/OS V1R11 support is operational only on IBM System z10 EC, z10 BC, z9 EC, and z9 BC hardware. PTFs of APAR OA28323 will be available for z/OS V1R1, along with rollbacks for V1R9 and V1R10.

27.3 STP alert event codes

Table 27-1 lists the possible event codes for these STP alerts.

Table 27-1 Event codes and their meaning

Hex	Code meaning
01	ETS not in use - an External Time Source (ETS) has not been defined for the Coordinated Time Network (CTN).
02	Dial-out time service is outside of the allowable tracking range - the time provided by the ETS differs from Coordinated Server Time (CST) by 60 seconds or more.
03	Dial-out time service within allowable tracking range - the difference between the time provided by the ETS and CST is now less than 60 seconds.
04	Dial-out access failure - the CTN is configured for dial-out access but the console was not able to perform a dial-out.

Hex	Code meaning
05	Dial-out access successful after having recognized a dial-out access failure.
06	NTP server failure - the CTN is in NTP or PPS mode and the console was not able to access any usable NTP server.
07	NTP server operational - the console is now able to successfully access an NTP server that is at stratum level 1 or greater.
08	NTP servers unsynchronized - the CTN is in NTP or PPS mode with both PPS ports enabled and the NTP-PRT offsets for the ports differ by one second or more.
09	NTP servers synchronized - the CTN is in NTP or PPS mode with both PPS ports enabled and the NTP-PRT offsets for the ports now differ by less than one second.
0A	NTP server switch to non-preferred NTP server- the CTN is in NTP mode with PPS ports disabled and the console has switched to the non-preferred NTP server to provide NTP information to the STP server.
0B	NTP server switch to preferred NTP server- the CTN is in NTP mode with PPS ports disabled and the console has switched to the preferred NTP server to provide NTP information to the STP server.
10	Preferred NTP server outside of allowable tracking range - the time provided by the preferred NTP server differs from Coordinated Server Time (CST) by 60 seconds or more.
11	Preferred NTP server within allowable tracking range - the difference between the time provided by the preferred NTP server and CST is now less than 60 seconds.
12	Preferred NTP server stratum level increase - the stratum level of the preferred NTP server has increased (for example, from stratum level 2 to stratum level 3).
13	Preferred NTP server stratum level decrease - the stratum level of the preferred NTP server has decreased (for example, from stratum level 3 to stratum level 2).
14	Preferred NTP server inaccessible - the preferred NTP server has transitioned from the accessible state to the inaccessible state.
15	Preferred NTP server accessible- the preferred NTP server has transitioned from the inaccessible state to the accessible state.
16	Preferred NTP server accessible- the preferred NTP server has transitioned from the inaccessible state to the accessible state.
17	Preferred NTP server stratum valid - the preferred NTP server has now reported a valid stratum level.
18	Preferred NTP server reference identifier (REFID) change - the REFID for the preferred NTP server has changed.
20	Non-preferred NTP server outside of allowable tracking range - the time provided by the non-preferred NTP server differs from Coordinated Server Time (CST) by 60 seconds or more.
21	Non-preferred NTP server within allowable tracking range - the difference between the time provided by the non-preferred NTP server and CST is now less than 60 seconds.
22	Non-preferred NTP server stratum level increase - the stratum level of the non-preferred NTP server has increased (for example, from stratum level 2 to stratum level 3).
23	Non-preferred NTP server stratum level decrease - the stratum level of the non-preferred NTP server has decreased (for example, from stratum level 3 to stratum level 2).

Hex	Code meaning
24	Non-preferred NTP server inaccessible - the non-preferred NTP server has transitioned from the accessible state to the inaccessible state.
25	Non-preferred NTP server accessible- the non-preferred NTP server has transitioned from the inaccessible state to the accessible state.
26	Non-preferred NTP server stratum error - the non-preferred NTP server has reported a stratum level of 0 or an invalid stratum level.
27	Non-preferred NTP server stratum valid - the non-preferred NTP server has now reported a valid stratum level.
28	Non-preferred NTP server reference identifier (REFID) change - the REFID for the non-preferred NTP server has changed.
81	Switch to non-preferred Pulse Per Second (PPS) port - the STP facility has switched to the PPS port specified as the non-preferred PPS port. This alert is sent by both the active and inactive stratum-1 servers.
82	Switch to preferred PPS port - the STP facility has switched to the PPS port specified as the preferred PPS port. This alert is sent by both the active and inactive stratum-1 servers.
83	PRT source ID change - the PRT source ID for the CTN has changed. The alert is sent by all servers when the PRT source ID for the CTN changes.
84	No PPS Signal - the STP facility no longer has access to a PPS signal.

27.4 HMC APIs

The purpose of the HMC application programming interfaces are to provide an open set of interfaces and a workstation platform for system management application providers. The interfaces provide the capability to exploit object-based industry-standard programming interfaces instead of building home-grown release-specific programs for collecting the hardware information needed to provide an integrated hardware and software system management solution.

Figure 27-3 on page 566 illustrates the integration of system management applications using the HMC application open programming interfaces to provide a single-system image (SSI) and a single point of control (SPOC).

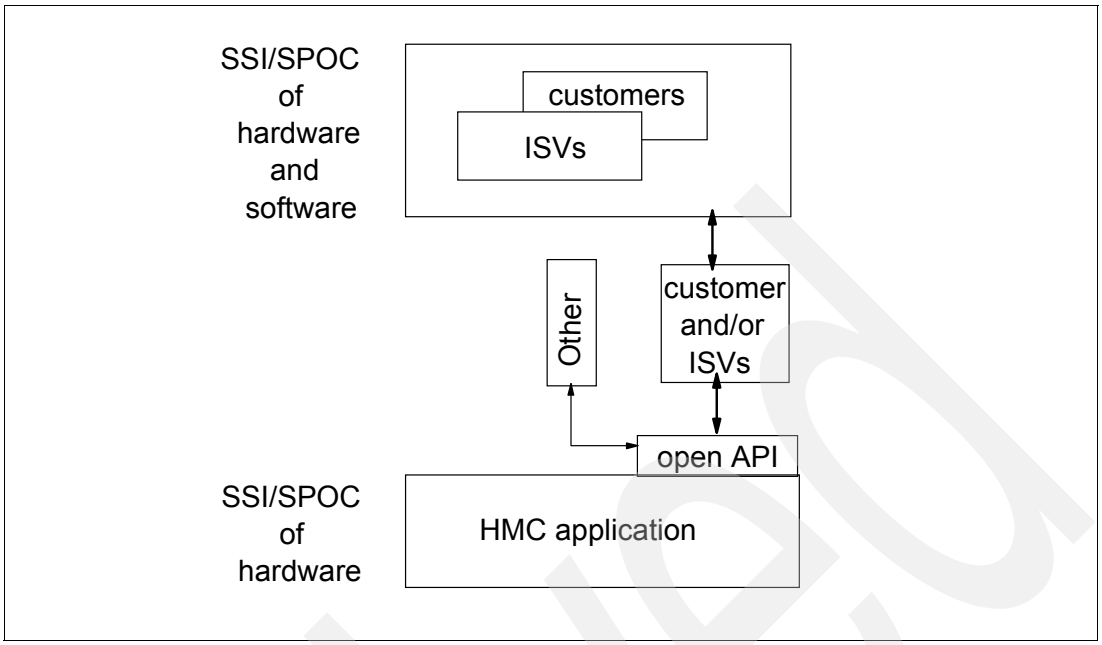


Figure 27-3 HMC APIs objectives

27.4.1 Overview of HMC support

Figure 27-4 shows a high-level architecture and flow of information for the HMC application management programming interfaces. The HMC application APIs are implemented using the Simple Network Management Protocol (SNMP) agent. The objects managed by the HMC application are stored in the Simple Network Management Protocol management information base (MIB).

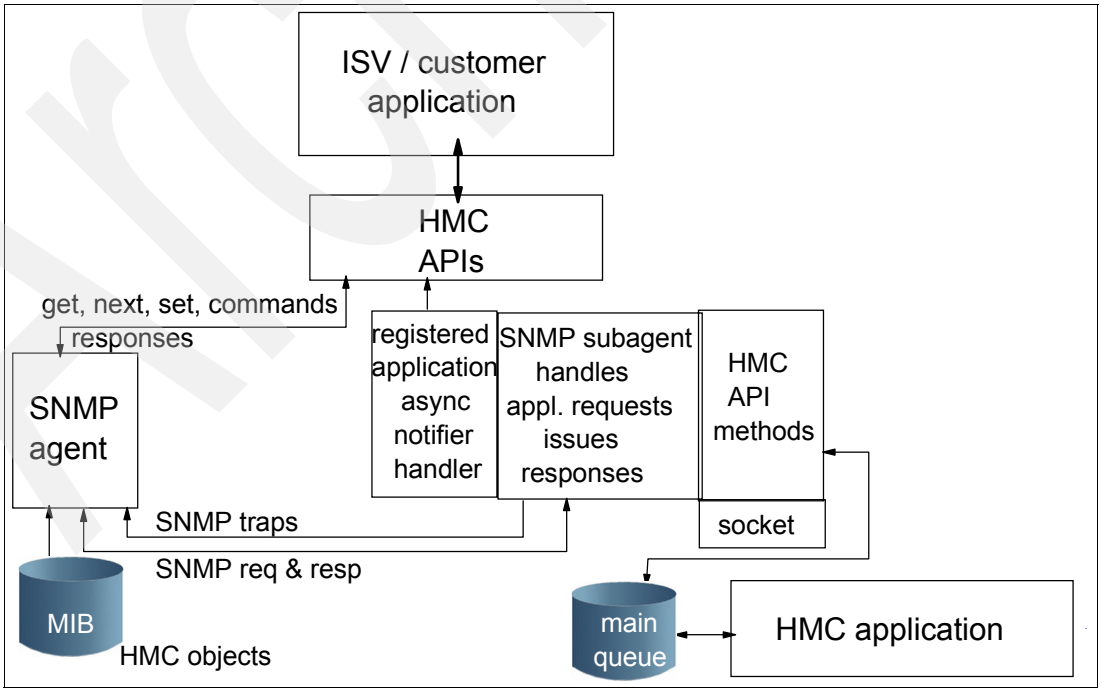


Figure 27-4 HMC application data exchange and commands API

27.4.2 HMC commands API

The HMC commands API allows applications (local or remote) to execute commands against the objects that HMC manages.

STP configuration support

As part of the HMC commands API, support for a new attribute that describes the STP configuration has been added to the Defined CPC object on all HMC version 2.10.1 or later. Also, the following STP commands were added to the Defined CPC object:

- Swap Current Time Server
- Set STP configuration
- Change STP-only CTN
- Join STP-only CTN
- Leave STP-only CTN

These new STP commands, provided in z10 systems are, the following:

- HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND

  In a configured STP-only Coordinated Timing Network (CTN), one CPC has the role of Current Time Server (CTS). If the CTN has both a Preferred Time Server and a Backup Time Server configured, either one can be the CTS. This command swaps the role of CTS from Preferred Time Server to Backup Time Server or vice versa. The target system must be the system that will become the CTS.

- HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND

  This command sets the configuration for an STP-only Coordinated Timing Network (CTN). The target system must be the system that will become the Current Time Server (CTS).

- HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_ COMMAND

  This command, sent to the Defined CPC with the role of Current Time Server (CTS) in an STP-only Coordinated Timing Network (CTN), changes the STP ID portion of the CTN ID for the entire STP-only CTN.

- HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND

  This command allows a CPC to join an STP-only Coordinated Timing Network (CTN). The target system cannot be the Current Time Server. If the CPC is already participating in an STP-only CTN, it will be removed from that CTN and join the specified one. If the CPC has an ETR ID, it will be removed.

- HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_ COMMAND

  This command removes a CPC from an STP-only Coordinated Timing Network (CTN). The target system cannot be the Current Time Server.

27.5 Proposal for STP alerts handling

The newly introduced STP alerts messages (triggered by new external interrupts) allow console messages to be issued through IEA031I or IEA395I WTOs on every image connected to the STP network.

Note that the last digit of the message IDs is the upper case letter I, which indicates that they are informational messages. Informational messages inform operators of a change (or

changes) within the STP network. Operators can then take the appropriate action, such as notifying someone to look at and repair the part related to the STP network problem.

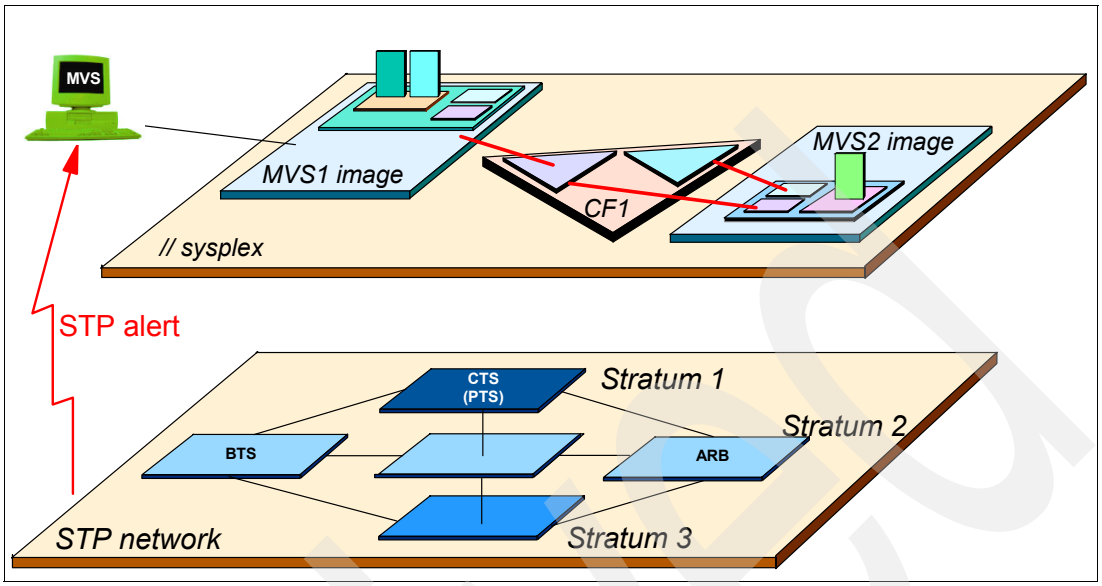


Figure 27-5 Network messages now seen by MVS operators

Message IEA395I

Message IEA395I also indicates a change in the STP network. Some of the STP alert event codes reflect a change in the STP configuration such as changes in the strati or changes in the STP role of certain systems. See *STP Implementation Guide*, SG24-7281, to learn about the operational possibilities available either through the HMC or the z/OS set of commands.

If the configuration is not too large, a well-trained operator can probably handle the IEA013I messages.

However, as soon as the configuration becomes large, one can doubt that operators, specially in the middle of the night, can easily cope with an elaborated STP network problem. It comes then to mind for the IEA013I and IEA395I messages issuance to trigger an automated tool, as part of the overall operational automation. Such an automated tool can help analyze the consequences of the STP alert event codes in a more systematic and cold manner, this is certainly something feasible with nowadays System REXX availability, as proposed in “HMC console API REXX example” on page 696.

A further step can, however, be considered as a potential use of STP alert event codes. The STP network has its own principles for managing its servers. As depicted in Figure 27-5, a sysplex environment comprised of SFM, WLM, and certain other components is managing the various z/OS and CF images and their availability. The STP network and sysplex environment, although aiming at the same goal, have so far been totally independent layers. This is why they are depicted, in Figure 27-6 on page 569, as two totally independent planes.

27.6 Summary

z/OS V1R11 now takes and handles a new subclass of STP Alert external interrupts related to ETS errors and issues:

- A new IEA031I message is added indicating the event that caused it, thus allowing proper handling of those events of interest.

- A new IEA395I message is added to handle the case of the preferred or backup servers located on stand-alone CFs, when the CTS switches to a separate server (either preferred or backup).

Note: As indicated in 27.4.2, “HMC commands API” on page 567, the HMC commands API (HWMxxx service calls) allows communication in z/OS V1R11 with the STP network. It is totally unlike existing interfaces. For example:

- The Perform Topology Function is used by HiperDispatch to communicate with PR/SM.
- Diagnose (HVC or x’83’) is used to communicate with the hardware or the Hypervisor.
- SCLP (unpublished) is used to communicate with the HMC when the messages console is displayed during the IPL phase.
- BCPii is a new address space that provides a new API (HWIxxx) which allows an authorized program to communicate with the HMC. See “BCPii overview” on page 511 for more information about this topic.

In z10 systems, along with these two new z/OS V1R11 messages, there are new HMC API commands. As shown in Figure 27-6, these complement the STP alerts WTOs and open up new ways:

- To automate STP network management
- To improve communication for operators between Parallel Sysplex and an STP network

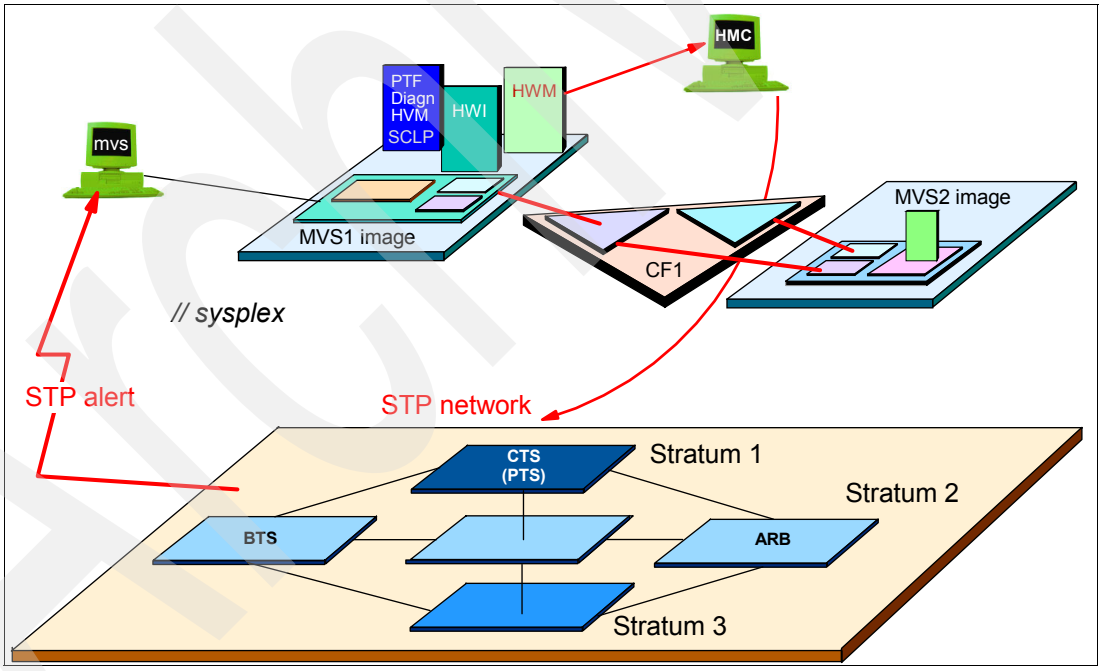


Figure 27-6 HMC commands API

28

XL C/C++/Metal C

This chapter describes new features of the XL C/C++ compiler, as well as enhancements for Metal C that are provided with z/OS V1R11.

The following features are described:

- Unicode literals
- Improved feedback features
  - Hiding skipped source code
  - Showing predefined macros
- Improved source compatibility features
  - Zero extent arrays
  - Statement expressions
- Integrated makedepend
- Upcoming C++ standard features
- Automatic prefetch
- Runtime checks
- Improved debugging support features
  - Captured source
  - MVS dbgld
  - DWARF namespace
- Metal C support for IEEE floating point
- New Metal C runtime static library capability

28.1 Unicode literals

So far, no support for unicode literals and strings has been provided. This means that storing string literals is a problem if the execution environment uses a separate code set from the one in which the string literal is encoded.

Porting code that uses Unicode strings or literals requires workarounds or extra code. For instance, one way to help alleviate the problem is by storing the string literals as Unicode literals. However, this can involve using clumsy mechanisms such as storing individual characters by their UTF value.

The XL C/C++ compiler makes this easier to do in z/OS V1R11 by allowing the entire string to be entered as natural text. It adds support for Unicode string specifiers for UTF-16 (u"...") and UTF-32 (U"...") strings as a language extension.

Common operations (such as compile time string literal concatenation) are supported for similar Unicode types and also for Unicode literals and normal strings. For instance, it adds Unicode string concatenation support for similar Unicode types and normal strings.

As shown in Figure 28-1, two string literals are stored in variables and the hexadecimal values of the individual characters that make up the string are displayed at run time.

Notice that the same string "abcd" has various values. The first string, `str1`, is stored in UTF-16 and thus displays the Unicode values for the string. The second string, `str2`, is stored in the native encoding, in this case IBM-1047. The values displayed for that second string correspond to the 1047 values for a, b, c and d.

```
#include <uchar.h>
#include <stdio.h>
int main() {
  char16_t* str1 = u"abcd";
  char* str2 = "abcd";
  printf("str1 = 0x%X 0x%X 0x%X 0x%X,
%d\n",str1[0],str1[1],str1[2],str1[3],sizeof(str1[0]));
  printf("str2 = 0x%X 0x%X 0x%X 0x%X,
%d\n",str2[0],str2[1],str2[2],str2[3],sizeof(str2[0]));
}
```

Figure 28-1 Unicode literals example

This new enhancement makes it easier to port the code containing the literals and strings, and to represent literals and strings in Unicode.

28.2 Improved feedback features

This section highlights the improved feedback features introduced with the XL C/C++ compiler of z/OS V1R11.

28.2.1 Hiding skipped source code

In C source code, it is sometimes difficult to find the code segment that is of interest to you. Also, source listings often have segments of code that are not compiled (for example, #if 0).

To address this issue, the **skipsrc** option has been added to control viewing skipped source sections.

```
SKIPSRC(SHOW|HIDE)
```

Figure 28-2 The skipsrc option

This new XL C/C++ option makes viewing the source listing easier. It allows you to hide skipped sections and to only display actual compiled code; see Figure 28-3.

```
int main() {
  #if 0
    int a = 66;
  #else
    int a = 55;
  #endif
  return a;
}

Invocation:
xlc –qskipsrc=hide –qsource

Results:
...
     1       |int main() {                  |      1
     2       |  #if 0                       |      2
     4       |  #else                       |      3
     5     1 |    int a = 55;               |      4
     6       |  #endif                      |      5
     7     2 |  return a;                   |      6
     8       |}                             |      7
...
```

Figure 28-3 Hide skipped source code

28.2.2 Showing defined macros

When programming elaborate C/C++ programs, the final values of compiler-defined macros and user-defined macros are not always easy to determine.

To alleviate this difficulty, the **showmacros** option is introduced in the XL C/C++ compiler of z/OS V1R11. Note that this option applies to preprocessed output only, and not to regular compilation.

```
SHOWMACROS(ALL|[NO]PRE)
```

Figure 28-4 Showmacros option

The showmacros option allows you to display compiler-defined macros and their values, as well as user-defined macros and their values. You can determine possible code paths and the values of macros at the end of a compilation; see Figure 28-5.

```
#define FOO bar
#undef FOO
#define FOO not bar
int a;
Invocation:
xlC -E -qshowmacros

Results:
...
#define FOO not bar
...
Note: The source code itself is not displayed
```

Figure 28-5 Showing defined macros

28.3 Improved source compatibility features

This section highlights the improved source compatibility features introduced with the XL C/C++ compiler of z/OS V1R11.

28.3.1 Zero extent arrays

Pre-C99 code from other platforms or compilers used zero extent arrays to enable functionality similar to flexible array members, but that functionality is not portable.

To address this issue, the XL C/C++ compiler of z/OS V1R11 adds support for zero extent arrays as a language extension and removes several restrictions on zero extent arrays to improve usefulness; see Figure 28-6.

```
#include <stdlib.h>
struct A {
  float a;
  int b[0];
};
int main(void) {
  struct A *s = malloc(sizeof(struct A) + sizeof(int) * 10);
  s->b[5] = 55;
  return s->b[5];
}
Invocation:
xlc
Results:
Return code 55
```

Figure 28-6 Zero extent arrays

This new support makes porting code containing zero extent arrays easier and it allows zero extent arrays at any position in structures (not just the end).

28.3.2 Statement expressions

In C, porting code that uses statement expressions requires workarounds or extra code, and this is not a feature defined by the C Standard.

To ease the porting of C code, the XL C/C++ compiler of z/OS V1R11 adds support for statement expressions as a language extension; see Figure 28-7.

```
Example:
#define maxint(a,b) \
({int _a = (a), _b = (b); _a > _b ? _a : _b; })

int main(void) {
  int a = 55;
  int b = 33;
  return maxint(a, b);
}
Invocation: xlc

Result: Return code 55
```

Figure 28-7 Statement expressions

The support of statement expressions by the XL C/C++ compiler of z/OS V1R11 makes the porting of code containing statement expressions easier, and also allows function-like macros to evaluate each operand only one time.

28.4 Integrated makedepend

Determining the dependencies for make file creation involves running a separate stand-alone utility. However, that utility may not contain all of the features of the compiler if it is another version.

The makedepend utility can also be used to create a makefile. The makedepend utility is used to analyze each source file to determine what dependency it has on other files. This information is then placed into a usable makefile. See *z/OS UNIX System Services Command Reference*, SA22-7802, for a detailed discussion of the makedepend utility.

Prior to z/OS V1R11, the stand-alone makedepend utility was used to analyze source files and determine source dependencies. As of z/OS V1R11, the M (-qmakedep) compiler option is introduced, and this compiler option is recommended to be used to obtain similar information. Specifying the M compiler option is equivalent to specifying the -qmakedep option.

The M compiler option is used to generate a make description file as a side effect of the compilation process. The description file contains a rule (or rules) suitable for make that describes the dependencies of the main compilation source file. The MF option is used in conjunction with the M option and specifies the name of the file where the dependency information is generated, or the location of the file, or both. The MF option has no effect unless make dependency information is generated.

On z/OS systems, the M compiler option resolves a number of complexities that are not properly managed by the compiler-independent makedepend utility, thereby improving the accuracy of the dependency information.

The XL C/C++ compiler in z/OS V1R11 allows you to have the compiler create the make dependency file itself. To do this, the following options have been added:

- –M and –qmakedep (generate make dependency file)
- –MF=filename (specify the file name, if desired)

Figure 28-8 on page 576 displays the integrated makedepend feature.

Note: This option is only supported using -q syntax. -M is the equivalent of specifying -qmakedep.

```
Example:
a.c:
#include “a.h"
int main(void) {
  return a;
}
a.h:
#include “a2.h"
int a = 55;
a2.h:
int b = 66;

Invocation:
xlc –M a.c

Results:
a.u (default make dependency file name):
a.o: a.c
a.o: a.h
a.o: a2.h
```

Figure 28-8 Integrated makedepend

The benefit provided by this new feature of XL C/C++ is that you are always using the same utility to compile and determine dependencies.

28.5 Automatic prefetch

This section provides background information about automatic prefetch, and also highlights the [NO]PREFETCH option introduced with the XL C/C++ compiler of z/OS V1R11.

Automatic prefetch background

z10 EC provides PREFETCH DATA and PREFETCH DATA RELATIVE LONG instructions. In z/OS V1R10, the C/C++ compiler does not emit these instructions automatically under ARCH(8). Built-in functions are available in z/OS V1R10 to allow C/C++ programmers to manually insert these instructions into their code.

These new instructions affect the prefetching/releasing of storage into and from the data or instruction cache. Generally, main storage may include one or more smaller, faster access buffer storages, sometimes called *caches*. A cache is usually physically associated with a

CPU or an I/O processor. The effects of the physical construction and use of distinct storage media are generally not observable by the program (except on performance).

Separate caches may be maintained for instructions and for data operands. Information within 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 in effect make the prefetching of storage into the data or instruction cache or the releasing of data from the cache. These instructions were introduced with the z10 EC model along with other functionalities to manage the polarization of the CPU topology. While developing C/C++ programs, however, it is sometime difficult to add such prefetching instructions in the correct location. On the other hand, prefetching code or data so that it is available at the right times can speed up performance.

PREFETCH option

When PREFETCH is in effect, the compiler may insert prefetch instructions in compiled code. When NOPREFETCH is in effect, prefetch instructions are not inserted in compiled code.

The compiler inserts prefetch instructions automatically where there are opportunities to improve code performance. The compiler will attempt to generate prefetch instructions for ARCH(8) or above. The compiler will not issue a message if PREFETCH is active and the ARCH level is below 8.

To address this issue, the XL C/C++ compiler of z/OS V1R11 introduces the [NO]PREFETCH option. With this option, the compiler automatically generates prefetching instructions where it thinks it may improve performance; see Figure 28-9.

```
xlC –qprefetch source.C
```

Figure 28-9 New prefetch option

The selection of prefetch instruction insertion location is based on heuristics and may change from time to time.

Note: Prefetch instructions are only inserted for ARCH levels at or greater than 8.

The XL C/C++ compiler may find locations to place prefetching instructions that are not obvious to determine while developing a C/C++ program. On the other hand, adding prefetch built-ins into a source code is still available with z/OS V1R11.

28.6 Runtime checking

Some common errors, such as the following types, can be difficult or time-consuming to find and correct:

- NULL pointer dereferences
- Integer divides by zero
- Array out of bounds accesses

To help you to locate such errors, XL C/C++ in z/OS V1R11 can emit signals (SIGFPE) when these error conditions occur.

This approach is based on the Compare and Trap instructions (CRT and CGRT), which were new additions to the z/Architecture machine instructions on z10 machines. These instructions 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. With z/OS V1R10, Language Environment was enhanced to provide information to condition handling routines so that they can detect that a Compare and Trap data exception has occurred.

Important: The General-Instructions-Extension Facility of z/Architecture, which is implemented on System z10, is required. Therefore, the ARCH level of XL C/C++ must be at or greater than 8.

A new option, RTCHECK, generates compare-and-trap instructions that perform certain types of runtime checking. RTCHECK generates compare-and-trap instructions, which perform certain types of runtime checking. The messages can help you to debug C programs such as the one shown in Figure 28-10.

```
#include <stdio.h>
int main(void) {
   int *a = NULL;
return *a;
}
```

Figure 28-10 A program that can be runtime-checked

You can specify the RTCHECK option more than one tim. The suboption settings are accumulated, but the later suboptions override the earlier ones.

You can use the **all** suboption along with the **no**... form of one or more of the other options as a filter. For example, using **xlc -qrtcheck=all:nonullptr** provides checking for everything except for addresses contained in pointer variables used to reference storage. If you use **all** with the **no**... form of the suboptions, then **all** is to be the first suboption.

Note: The RTCHECK option is only valid for architecture level 8 or above, and for Language Environment V1.10 and above.

If the compiler is invoked as shown in Figure 28-11, then the condition is trapped and message CEE 3234S is issued.

```
Invocation:
xlc –qrtcheck (default suboption is ALL: add all runtime checks)
Results:
CEE3234S The system detected a Compare and Trap data exception.
         From entry point main at statement 6 at compile unit offset +00000052
at entry offset +00000052 at address
         2600894A.
Floating point exception
```

Figure 28-11 Sample CEE3234S message

This new option can take the value of: RTCHECK([BOUNDS,DIVZERO,NULLPTR]).

The RTCHECK option, introduced for XL C/C++ in z/OS V1R11, makes it easier for programmers to find certain types of logic errors, and allows harnesses to trap certain errors and recover gracefully.

28.7 Improved debugging support

This section highlights the improved debugging enhancements that have been added to the XL C/C++ compiler of z/OS V1R11to improve debugging activities.

28.7.1 Captured source

Debuggers require access to the original source code so that it can be displayed to the user. Quite often, the problem is that the source may have been changed or moved to a separate directory, or debugging may be done on a separate machine.

XL C/C++ in z/OS V1R11now allows you to store the source files in the debugging information.

Figure 28-12 on page 579 shows an example that can be used to compile hello1.c and hello2.c and create a module level debug side file called hello.mdbg that contains captured source for these two source files.

A debugger can then retrieve the source file contents from hello.mdbg without the need to access hello1.c and hello2.c.

```
xlc -g hello1.c hello2.c -o hello
dbgld -c hello
```

Figure 28-12 Example of captured source

Though Debug Tool V10.1 is the only IBM debugger that supports this right now, this new debugging feature generally allows debuggers to display the source code even if the original source is no longer available in the original location.

28.7.2 MVS dbgld

The existing dbgld utility that is used for creating a module level debug side file needs to be run from z/OS Unix System Services. However, certain clients require the utility to be run from MVS, as well.

Therefore, an equivalent utility called CDADBGLD is provided with z/OS V1R11 that can be run from MVS.

As an example, after compiling and binding a program, suppose the executable module is in MYHLQ.MOD(TEST). Now the CDADBGLD utility can be invoked with the JCL as in Figure 28-13.

```
//CDADBGLD   EXEC  CDADBGLD,
//         INFILE='MYHLQ.MOD(TEST)',
//         OUTFILE='MYHLQ.MDBG(TEST),DISP=SHR'
```

Figure 28-13 CDADBGLD utility

MVS data set MYHLQ.MDBG(TEST) now contains a module-level debug side file. From now on, a module-level debug side file can be created from MVS.

28.7.3 DWARF namespace

The existing DWARF debug information generated during compilation is lacking information on namespaces. This means that debuggers cannot display any information to users about namespaces.

In z/OS V1R11, however, it is now possible to store information about namespaces in the DWARF debug information.

As an example, suppose a C++ file contains the statements shown in Figure 28-14 on page 580.

```
namespace ABC {
     int abc;
   }
   using ABC::abc;
```

Figure 28-14 Definition of a name space

Then the DWARF information will contain information as shown in Figure 28-15.

```
<1><  100>      DW_TAG_namespace
                   DW_AT_name                  ABC
                   DW_AT_sibling               <300>
   <2><  200>      DW_TAG_variable
                   DW_AT_name                  abc
                   DW_AT_MIPS_linkage_name     abc__3ABC
                   DW_AT_type                  <...>
                   DW_AT_external              yes(1)
                   DW_AT_location              ...
   <1><  300>      DW_TAG_imported_declaration
                   DW_AT_import                <200>
```

Figure 28-15 DWARF information

With z/OS V1R11, programs containing namespaces can be debugged correctly.

28.8 METAL C

Prior to z/OS V1R9, all z/OS XL C compiler-generated code required Language Environment®. In addition to depending on the C runtime library functions that are available only with Language Environment, the generated code depended on the establishment of an overall execution context, including the heap storage and dynamic storage areas. These dependencies prohibit you from using the XL C compiler to generate code that runs in an environment where Language Environment did not exist.

The XL C METAL compiler option, introduced in z/OS V1R9, generates code that does not require access to the Language Environment support at run time. Instead, the METAL option provides C-language extensions that allow you to specify assembly statements that call system services directly. Using these language extensions, you can provide almost any assembly macro, and your own function prologs and epilogs, to be embedded in the generated HLASM source file. When you understand how the METAL-generated code uses MVS linkage conventions to interact with HLASM code, you can use this capability to write freestanding programs.

Various enhancements to Metal C have been requested by users. This section highlights the two major new enhancements provided by z/OS V1R11:

- Floating point support
  - The library functions support IEEE floating point numbers.
  - The library uses code page IBM-1047 and the En_US locale definitions to perform its functions.
- Static object libraries

28.8.1 Metal C runtime library to support floating point

To satisfy user requirements, Metal C runtime library functions have been added and enhanced to support floating point values.

Therefore, floating point support is added to Metal C with the following characteristics:

- IEEE binary floating point format only
  - Functions exposed with FLOAT(IEEE) compiler option.
- Rounding modes specified using ROUND compiler option.
  - Default set to nearest.
- Floating point format and rounding mode established at compile time only
- Runtime behavior uses En_US locale
- New floating point functions have been added to the runtime
    - strtod() – string to double.
    - strtof() – string to float.
    - strtold() – string to long double.Some functions have also been enhanced for floating point
    - sprintf() and sscanf() families of function.
  - Floating point format specifiers (e, E, f, F, g, G) now supported.
- Overflow and underflow conditions are not supported by the runtime
  - It is a user responsibility to disable overflow and underflow conditions.

Proper insertion of #include <float.h> is to be done in the source file.

The float.h header file contains definitions of constants listed in ANSI 2.2.4.2.2. The constants describe the characteristics of the internal representations of the three floating-point data types: float, double, and long double. Table 28-1 lists the definitions contained by float.h.

Table 28-1 Definitions in float.h

Constant	Description
FLT_RADIX	The radix for a z/OS XL C Metal C application. For FLOAT(IEEE), the value is 2.
FLT_MANT_DIG DBL_MANT_DIG LDBL_MANT_DIG	The number of hexadecimal digits stored to represent the significand of a fraction.
FLT_DIG DBL_DIG LDBL_DIG	The number of decimal digits, q, such that any floating-point number with q decimal digits can be rounded into a floating-point number with p radix FLT_RADIX digits and back again, without any change to the q decimal digits.
FLT_MIN_10_EXP DBL_MIN_10_EXP LDBL_MIN_10_EXP	The minimum negative integer such that 10 raised to that power is in the range of normalized floating-point numbers.
FLT_MAX_EXP DBL_MAX_EXP LDBL_MAX_EXP	The maximum integer such that FLT_RADIX raised to that power minus 1 is a representable finite floating-point number.
FLT_MAX_10_EXP DBL_MAX_10_EXP LDBL_MAX_10_EXP	The maximum integer such that 10 raised to that power is in the range of representable finite floating-point numbers.
FLT_MAX DBL_MAX LDBL_MAX	The maximum representable finite floating-point number.
FLT_EPSILON DBL_EPSILON LDBL_EPSILON	The difference between 1.0 and the least value greater than 1.0 that is representable in the given floating-point type.
FLT_MIN DBL_MIN LDBL_MIN	The minimum normalized positive floating-point number.
DECIMAL_DIG	The minimum number of decimal digits needed to represent all the significant digits for type long double.
FLT_EVAL_METHOD	Describes the evaluation mode for floating point operations. This value is 1, which evaluates v All operations and constants of types float and double to type double, and v All operations and constants of long double to type long double.

28.8.2 Metal C runtime static library capability

Users have requested the ability to bind Metal C Runtime Library functions directly into their load modules for the following reasons:

- This allows Metal C applications to be deployed on a system that does not support the system vector version of the runtime.

- This allows Metal C routines to use library functions during early IPL, prior to initialization of the library's system vector.

Starting with Metal C in z/OS V1R11, the Metal C runtime library supports two versions of its library functions: a system library and a static object library. The behavior of the functions within the two versions is the same. What differs is where the functions are located and how the Metal C application interacts with them.

System library

The system library is a version of the Metal C runtime library that exists within the system's link pack area, and is made available during the system IPL process. It is suggested that you use the system library if the Metal C application is run on a level of z/OS that supports the runtime library, and the application runs after the library has been made available. This library has the added advantage of not requiring application module re-links when service is applied to the library.

To use the system library version, simply include the desired Metal C runtime library headers in the Metal C application source code. The default behavior of the headers is to generate code within the application that calls this system library. No additional binding is needed in order for these function calls to work.

Static object library support

The Static object library is a version of the Metal C runtime library that gets directly bound with a Metal C application load module. The resulting application is self-contained with respect to the library; all library function calls from the application result in the functions bound within the load module to be driven.

It is suggested that you use the static object library if the Metal C application meets either of the following requirements:

- The application is run on a supported level of z/OS that does not support the system library (before z/OS V1R9).
- The application is run during system IPL before the system library has been made available.

The static object library functions are provided in two system data sets: SYS1.SCCR3BND and SYS1.SCCR6BND. SYS1.SCCR3BND is used with Metal C applications that have been compiled using ILP32 and run AMODE 31. SYS1.SCCR6BND is used with Metal C applications that have been compiled using LP64 and run AMODE 64.

To use the static object library, you must take the following steps:

1. Define the __METAL_STATIC feature test macro before including the headers in your Metal C program, and then compile the program; for example:
   - #define __METAL_STATIC
   - #include <stdio.h>

   This will cause library function calls in the program to generate external references to the functions contained within the SCCRnBND data sets.
2. Bind the compiled object with the corresponding SCCRnBND data set. How this is done depends on the environment on which the binding takes place:
   – Batch: When using the binder from a batch job, use the CALL option, and use the SYSLIB DD to identify the static object library data set that you want to bind with.

– Unix System Services shell: From the shell, it is suggested that the ld shell command be used to bind the application with the library functions. This avoids conflicts with the Language Environment stubs that the c89 family of commands may introduce. Use the -S option to identify the static object library data set that you want to bind with. For example:
  - -S //"'SYS1.SCCR3BND'"

Note: When service is applied to the static object library, the Metal C application must be re-linked to pick up the changes.

Each supported runtime function shipped in an object that Metal C applications can bind with directly:

SCCR3BND – for AMODE 31

SCCR6BND – for AMODE 64

Static library selected instead of the system vector call using __METAL_STATIC macro

#define __METAL_STATIC in program source before including headers

General library usage notes

A Metal C application can use either the system library or the static object library, but not both. The mixing of system library calls and static object library calls within the same application is not supported.

All static objects bound to the application load module must be at compatible service levels.

Metal C runtime library functions are not supported under Language Environment and must not be used within a Language Environment program, because equivalent functions are already available.

28.9 Summary

XL C/C++ provides, with z/OS V1R11, enhancements for the following areas:

- Unicode literals
- Improved feedback features
- Improved source compatibility features
- Integrated makedepend
- Upcoming C++ standard features
- Automatic prefetch
- Runtime checks
- Improved debugging support

Two enhancements to Metal C are also provided:

- Additional interfaces to support floating point processing.
- Function objects that can be bound with user application.

29

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.

HiperDispatch was introduced with z/OS V1R10 and rolled down to JBB772S, HBB7730 and HBB7740. This chapter discusses the following:

- HiperDispatch overview
- Activating HiperDispatch
- Monitoring HiperDispatch
- HiperDispatch enhancements to z/OS V1R10
- HiperDispatch enhancements with z/OS V1R11

29.1 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 because 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 separate 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.

29.1.1 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 29-1.

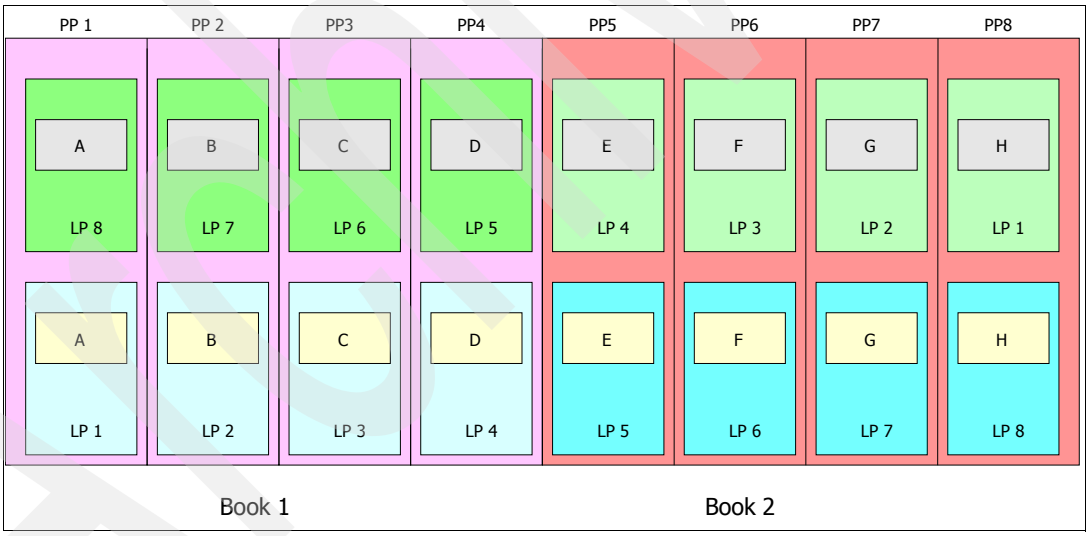


Figure 29-1 z10 without HiperDispatch

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, thus reducing L1 and L2 cache hits which in turn reduces the throughput of the system.

29.1.2 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. Because a thread is going to be dispatched among a small group of processors, the probability of a L1 cache hit also increases.

Using our hypothetical example in Figure 29-2, if both LPARs consume their full LPAR weight, then 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 among 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 29-2.

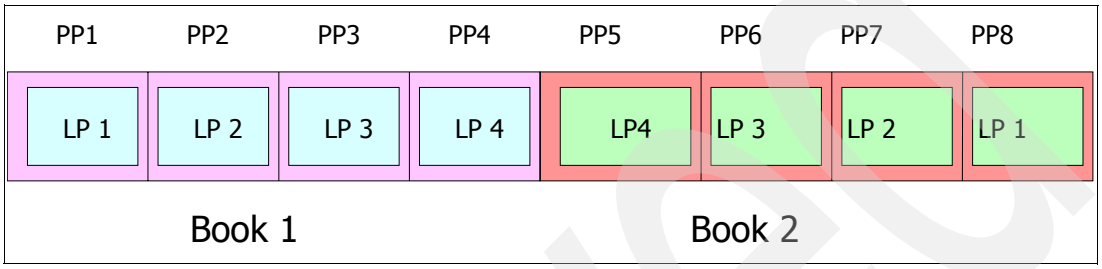


Figure 29-2 z10 with HiperDispatch and two LPARs of equal weight

Because 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 separate LPAR's vertical highs and mediums so that an LPAR that needs more capacity can receive additional capacity from a separate LPAR that 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 separate 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. After a vertical low is unparked, it will dispatch work on behalf of that LPAR. The vertical low will be parked after 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% to 10%, depending on the 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, then the larger the capacity gain from HiperDispatch.

29.2 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 JBB772S, HBB7730, and HBB7740 through the following 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 deactivated 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
```

29.3 Monitoring HiperDispatch

HiperDispatch has implications for workload management (WLM) and help processing. There are also changes in RMF reports to display HiperDispatch information.

29.3.1 WLM considerations

With HiperDispatch, the prioritization of workloads by way of 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.

Because HiperDispatch changes how work is dispatched among processors, additional attention and review of the WLM policy can 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 separate 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.

29.3.2 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 by way of 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 29-3 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 is 80% for each of the eight 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 00000000000DC6CE   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 29-3 RMF CPU Activity report

29.4 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. 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 separate 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 another 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 another affinity node or a CP if IFAHONORPRIORITY=YES.

29.4.1 Alternate wait management

The alternate wait management (AWMT) and honor priority values specified through 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 to 1000000 microseconds.

    Specifying CCCAWMT >= 500000 disables AWM.
  - Default for HiperDispatch = NO is 12000 microseconds.
  - For HiperDispatch =YES, the valid range is 1600 to 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 to 499999 microseconds.
  - Default for HiperDispatch = NO is 12000 microseconds.
  - For HiperDispatch = YES, the valid range is 1600 to 3200 microseconds. Specifying a value outside that range will result in 3200 microseconds being assigned.
  - Default for HiperDispatch =YES is 3200 microseconds.

  Note: The valid range for ZAAPAWMT has changed from 1600 - 3200 to 1600 - 499999 microseconds with APAR OA26789. See OA26789 - Improvements to PARK processing and AWMT, for more details.

- ZIIPAWMT
  - Alternate Wait Management (AWMT) value for zIIP processors.
  - For HiperDispatch = NO, the valid range is 1 to 499999 microseconds.
  - Default for HiperDispatch = NO is 12000 microseconds.

- For HiperDispatch = YES, the valid range is 1600 to 3200 microseconds. Specifying a value outside that range will result in 3200 microseconds being assigned.
- Default for HiperDispatch =YES is 3200 microseconds.

Note: The valid range for ZIPPAWMT has changed from 1600 - 3200 to 1600 - 499999 microseconds with APAR OA26789. See OA26789 - Improvements to PARK processing and AWMT, for more details.

- 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:

- 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. 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. This value is in 64-bit TOD format where bit 51 corresponds to 1 microsecond. 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.

29.5 HiperDispatch enhancements to z/OS V1R10

There have been a number of important improvements to HiperDispatch for z/OS V1R10 and lower through new APARs which have been included with base z/OS V1R11. This section discusses the improvements made through these APARs. Install them on z/OS V1R10 and earlier systems.

OA26789 - Improvements to PARK processing and AWMT

With HiperDispatch enabled, when an LPAR is very close to 100% busy and the CPC has white space available, a vertical low (discretionary) processor is to be unparked to allow the LPAR to expand into the available CPU capacity in the CPC.

With the initial implementation of HiperDispatch, vertical low (discretionary) processors were not necessarily unparked when an LPAR was very busy and there was whitespace in the CPC. This effect was mainly observed on low-weighted LPARs with only a few or no vertical

high CPs. This issue has been addressed by APAR OA26789 which has PTFs for JBB772S, HBB7730, HBB7740, and HBB7750.

In addition, with HiperDispatch enabled, the range accepted for ZAAPAWMT and ZIIPAWMT has been increased. ZAAPAWMT and ZIIPAWMT control how aggressive zAAPs and zIIPs are in asking for help. When ZAAPAWMT or ZIIPAWMT is set too low, then for LPARs with discretionary zAAPs or zIIPs, it can result in zAAPs or zIIPs asking CPs for too much help. When zAAPs or zIIPs get too much help from CPs, the zAAP or zIIP utilization can significantly decrease.

With OA26789, the valid range for zAAPAWMT and ZIIPAWMT with HiperDispatch enabled increases for 1600 to 3200 microseconds to 1600 to 499999 microseconds. This allows a higher setting for zAAPAWMT or ZIIPAWMT to be used as appropriate for LPARs with discretionary zAAPs or zIIPS available.

The APAR description of the problem and conclusion is shown in Figure 29-4.

```
PROBLEM DESCRIPTION:
1. The algorithm in IRABAADJ tries to adjust the CPU capacity of a Lpar as
needed by unparking / parking of vertical low CPs. Prior to the changes with
this APAR the algorithm agressively tried to park vertical low CPs as soon as
they appeared to be no longer useful. This led to situations that unparking did
not take place even though there was demand in the Lpar and also available CPU
capacity on the CEC.  This effect was mainly observed with low weighted Lpars
with only a few or no vertical high CPs.

2. The range of the IEAOPT parameters ZAAPAWMT and ZIIPAWMT must be extended to
allow values greater than 3200 in case of HIPERDISPATCH=YES.

PROBLEM CONCLUSION:
This APAR changes the unpark / park algorithm in module IRABAADJ.  A vertical
low CP will be unparked as soon as MVS_busy of the Lpar is getting above 95%
and there is sufficient CEC capacity available.

Parking of vertical low CPs is done when the average efficiency of the low CPs
is getting too low. The average low CPs efficiency is stored in variable
VCM_LparCapVlAvgEffect (IRABAVCM) and calculated as  VCM_LparCapVlAvgEffect =
     VCM_LparCapUsedDiscr + 256 / VCM_LparCapNonGuaran.
 - VCM_LparCapUsedDiscr represents the used capacity on vertical low CPs.
 - VCM_LparCapNonGuaran represents the CP capacity provided by unparked
vertical low CPs.

The algorithm to unpark / park vertical low CPs is performed for every CPU type
(general CP, zAAPs, zIIPs) individually. Major changes were done in macro
IRABAVCM substructure VCM_LparCaps(PtIxDim) to make room for new variables
replacing old no longer needed variables. All variables of
VCM_LparCaps(PtIxDim) will also be written to the SMF99 subtype 12 record. For
this IRASMF99 was changed accordingly.

The range of the IEAOPT keywords ZAAPAWMT and ZIIPAWMT has been
extended to 1600-499999 which corresponds to a timeframe from
1.6 to 500 milliseconds.
```

Figure 29-4 APAR OA26789

OA24920 - Performance degradation with AWMT off

Alternate wait management (AWMT) is always enabled for shared LPARs with HiperDispatch enabled. AWMT is disabled on dedicated LPARs and can be disabled for shared LPARs with HiperDispatch disabled.

Systems running on dedicated LPARs or shared LPARs with AWM off can suffer a performance degradation when processors get parked and unparked due to CONFIG CPU ONLINE/OFFLINE activity. The park and unpark processing results in the processor's wait time slice in LCCAWTSC being set too high. Both dedicated LPARs and shared LPARs with AWMT off require a short wait time slice so that CPUs wake up frequently enough to dispatch work.

The APAR description of the problem and conclusion is shown in Figure 29-5.

```
PROBLEM DESCRIPTION:
Dedicated LPAR(s) or shared LPAR(s) with alternate wait management (AWM) off
have the CPU wait time slice (LCCAWTSC) too high for those environments.
Depending on the workload running on the dedicated LPAR(s) or shared LPAR(s)
with alternate wait management (AWM) off, the LPAR(s) may see a performance
degradation.

PROBLEM CONCLUSION:
CPU wait time slice (LCCAWTSC) is set properly for dedicated LPARs and shared
LPARs with alternate wait management (AWM) off.
```

Figure 29-5 APAR OA24920

OA25825 - Refreshing need help CPU masks

The system maintains CPU masks which are used to select a good candidate CPU for help when an affinity node needs help. All CPUs in an affinity node have the same characteristics as all the other CPUs in that affinity node. When a CPU is assigned to an affinity node, the CPU masks are not necessarily updated in a timely fashion prior to OA25825.

Prior to OA25825, when a CPU joined an affinity node, the CPU masks were only updated when the affinity node's state changed. Because taking a CPU offline removes a CPU from the CPU masks, processing like IRD CPU management, which brings CPUs offline and online, can result in corrupting the CPU masks to a point where help processing is ineffective.

In HiperDispatch=NO, over many hours, days, or weeks zAAPs and zIIPs can be unable to ask CPs for help. HiperDispatch=NO was the most likely environment to experience this problem because the interval used to determine the node's state was too long. If the node's state does not change after IPL, then CPUs configured online after IPL may be unable to be chosen for help.

The same problem also existed with HiperDispatch=YES, but it was less likely to occur because the interval used to determine the node's state was significantly shorter. This shorter interval makes it much more likely that the need help state of a given node will change and thereby avoid the problem. The APAR description of the problem and conclusion is shown in Figure 29-6 on page 595.

```
PROBLEM DESCRIPTION:
In a HiperDispatch=NO environment, when zAAP/zIIP processors are not able to
handle all the zAAP/zIIP workload, HELP processing may not signal any standard
processors for HELP for the zAAP/zIIP workload.  Part of the HELP processing
decision process uses the AWUQ High_SigWait_Mask to determine which processors
are available to provide HELP. When the mask is binary zeroes, no standard CPUs
are available to provide HELP for the zAAP/zIIP workload.

The problem is that there are no CPs in this mask even though there are
standard CPs have a high significant wait.  As a result, the zAAPs/zIIPs do not
choose CPs for help due to a high significant wait.

The need help CPU masks are only refreshed when the CPUs assigned to a given
node change state. All CPUs in a node have the same state as all the other CPUs
in the same node. So the cumulative statistics of all CPUs in each node
determine the state of each node. These statistics are maintained in internal
CPU masks which are used in the need help logic. These internal CPU masks
are only updated when the node's state changes.

For example, each node has a characteristic known as significant wait.  There
are 3 categories of significant wait: high significant wait, low significant
wait, and no significant wait.  When a CPU comes online, it is assigned to the
appropriate node.  That CPU does not get added into the appropriate significant
wait mask until after the node's significant wait state changes.

This problem is most likely to appear while running with HiperDispatch=NO.
With HD=NO, the interval used to determine the need help state of each node is
too long.  The longer interval makes it less likely the need help state of a
given node will change.  Should a node's state not change at all after IPL,
then any CPUs configured online after IPL may not be added into the internal
need help CPU masks.

This problem also exists in HiperDispatch=YES, but it is less likely to occur.
In HD=YES, the interval examined to determine the need help state of each node
is much smaller.  This shorter interval makes it more likely the need help
state of a given node will change.

PROBLEM CONCLUSION:
Refresh the internal need help CPU masks when CPUs come online and are assigned
to a node.

Supporting APAR OA25841 will shorten the interval being used to determine the
need help state of each node in HD=NO.
```

Figure 29-6 APAR OA25825

29.6 HiperDispatch enhancements with z/OS V1R11

Significant enhancements have been made to HiperDispatch in z/OS V1R11. This includes improvements in:

- How dispatching priority promotion is handled for a task that holds a LOCAL or CML lock
- Help processing

Promotion of LOCAL or CML lock holder

There are significant RAS improvements in handling the promotion of a task holding a LOCAL or CML lock with HiperDispatch enabled. New fields have been added to the ASSB and ENCB to allow WLM to factor lock promote time into their calculations. The supervisor code to add this new function was updated through APAR OA27855 and the WLM code was updated through APAR OA27810.

The APAR description of the problem and conclusion is shown in Figure 29-7.

```
The HiperDispatch lock promotion algorithm has been enhanced to record the
amount of time an enclave / address space has been promoted. WLM apar OA27810
makes use of these new times. See OA27810 for more information.
```

Figure 29-7 APAR OA27855

A problem with the original implementation was that a task running at a promoted dispatching priority can monopolize a CPU. This was addressed through APAR OA28744 by limiting the amount of time a task will be promoted for and then forcing it to run at its normal dispatching priority.

The APAR description of the problem and conclusion is shown in Figure 29-8.

```
PROBLEM DESCRIPTION:
To improve RAS in HD=YES, a unit of work which was promoted for holding a local
/ CML lock that was not released in a timely fashion must give up its lock
promotion priority until all units of work between the lock promotion priority
and the unit of work's base dispatch priority has completed.

PROBLEM CONCLUSION:
In HD=YES, when a unit of work was promoted for holding a local / CML lock
which was not released in a timely fashion, force the unit of work to run at
its base dispatch priority until all units of work between the lock promotion
priority and the unit of work's base dispatch priority has completed.
```

Figure 29-8 APAR OA28744

Note: APARs OA27810, OA27855, and OA28744 have been rolled down to HBB7730, HBB7740 and HBB7750.

Improvements to help processing

The help algorithms have been enhanced so that CPs will not automatically give help to zAAPs and zIIPs before going into a wait. This prevents too much zAAP and zIIP processing from being offloaded to CPs when a zAAP or zIIP requests help from a CP.

30

Common Information Model

This chapter provides the latest enhancements to Common Information Model (CIM) provided by z/OS V1R11. The Common Information Model (CIM) is a standard data model for describing and accessing systems management data in heterogeneous environments. It allows system administrators or vendors to write applications that measure system resources in a network with different operating systems and hardware.

See *z/OS Common Information Model User's Guide*, SC33-7998, for a description of the overall CIM data model and an explanation of how to use it. That publication also explains the z/OS-specific supplements and differences, and provides installation, configuration, and security instructions for z/OS systems.

The following topics are discussed in this chapter:

- An introduction to CIM details upgrades and enhancements to do the following:
  - Comply with OpenPegasus 2.8
  - Enhance the Java client part
  - Provide instrumentation enhancements

30.1 Introduction to Common Information Model

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.

CIM allows management applications from various vendors to manage a heterogeneous environment of systems through 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.

30.1.1 CIM cross-platform management

With CIM, management applications from various vendors can manage a heterogeneous environment of systems through 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 30-1 on page 599, 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 to the calling client through the CIM/XML over HTTP access protocol.

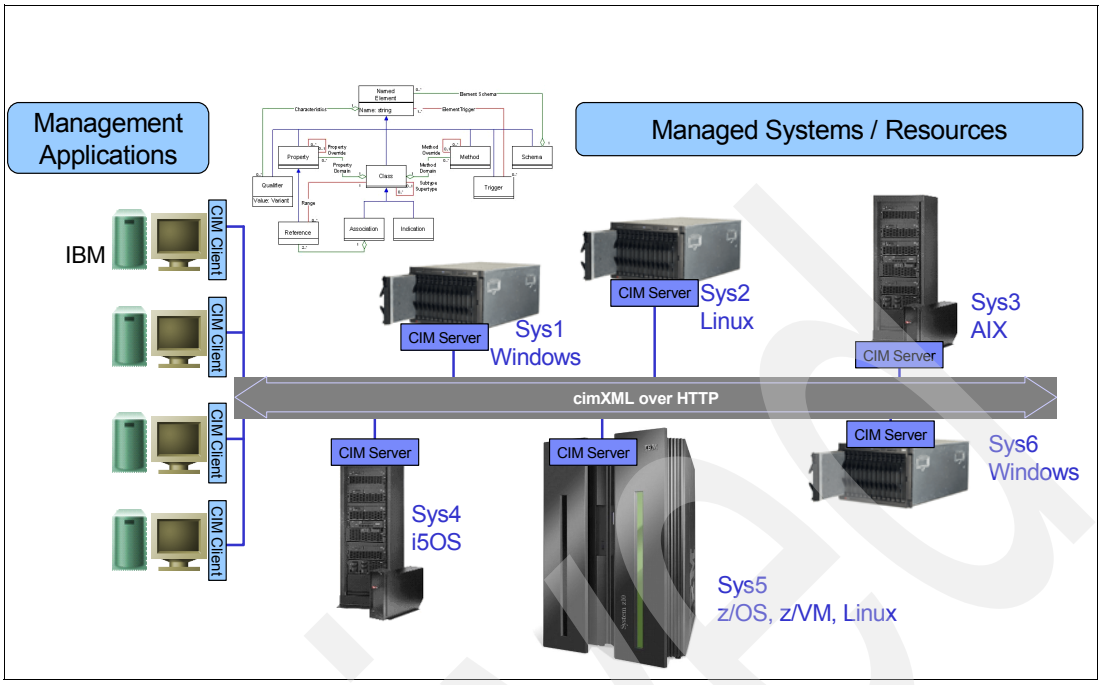


Figure 30-1 Cross-platform management with CIM

30.1.2 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 put 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 30-2 on page 600, is used to receive client requests, collect the requested metrics and data from the managed system, and return the results to the client.

CIM Server support with z/OS V1R9

The CIM Server exploits 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. 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.

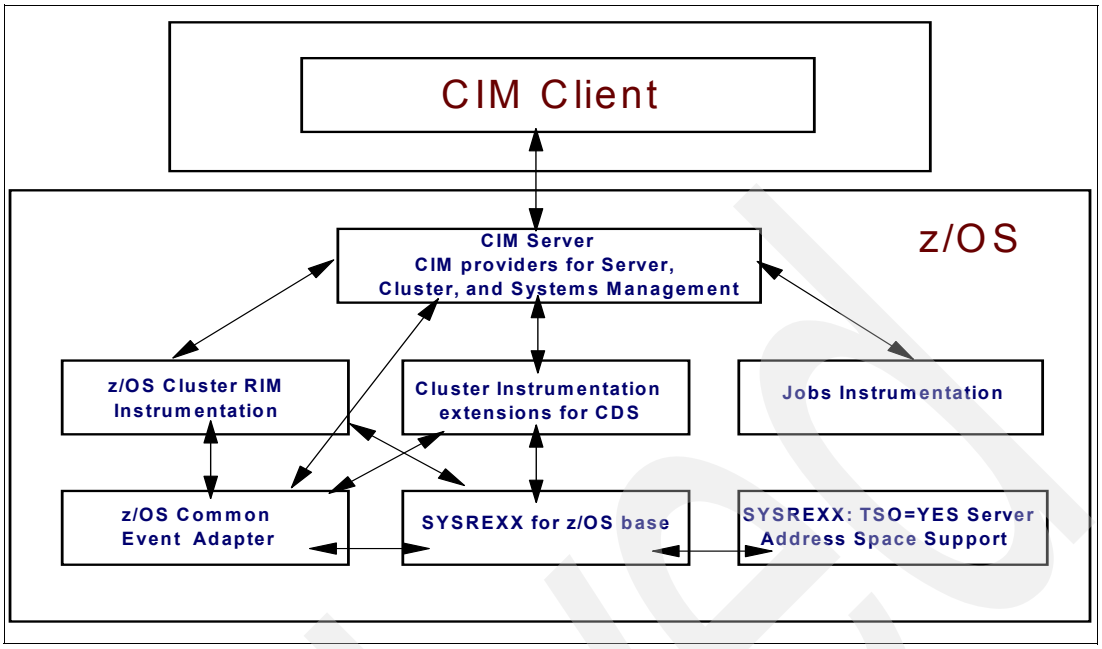


Figure 30-2 CIM components and dependencies

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 30-3 on page 601.

The jobs instrumentation extends the reach of JES and z/OS UNIX System Services management up to the CIM Server.

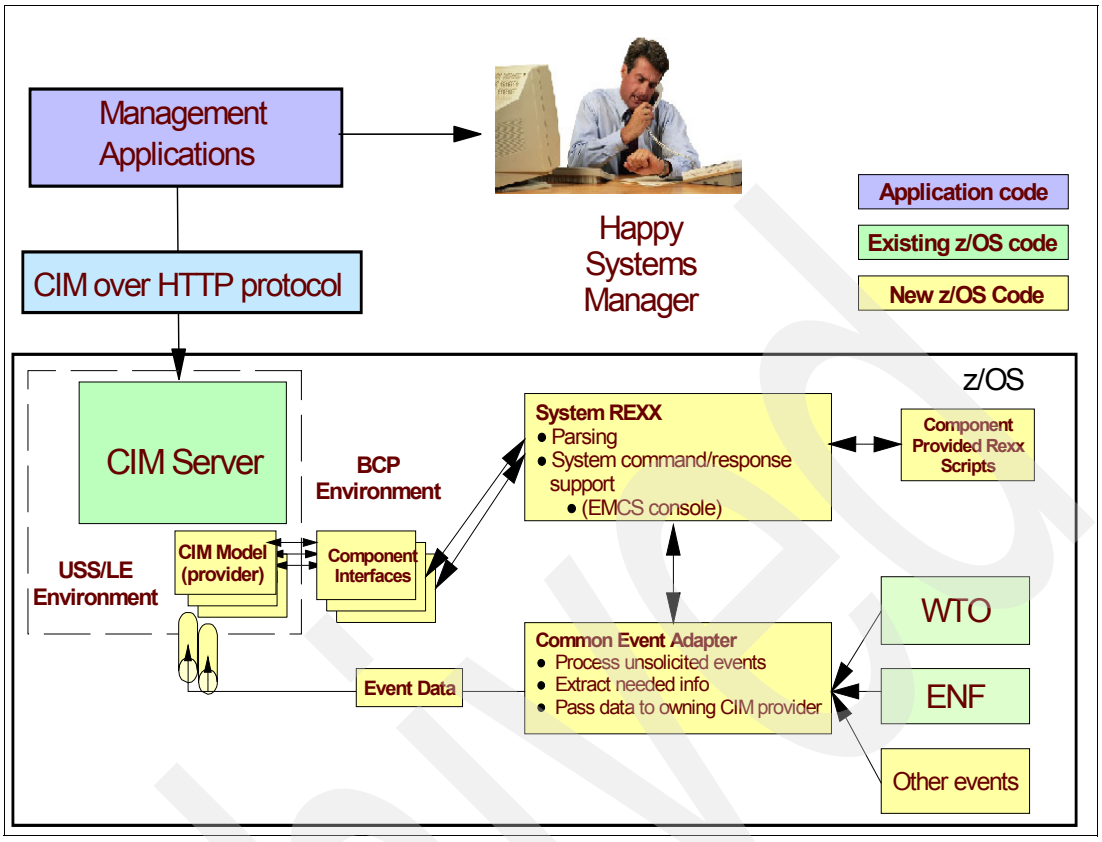


Figure 30-3 Cluster and jobs instrumentation

30.1.3 CIM Server overview

Since z/OS V1R9, the CIM Server has new components, as shown in Figure 30-4 on page 602. 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 through 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. 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.

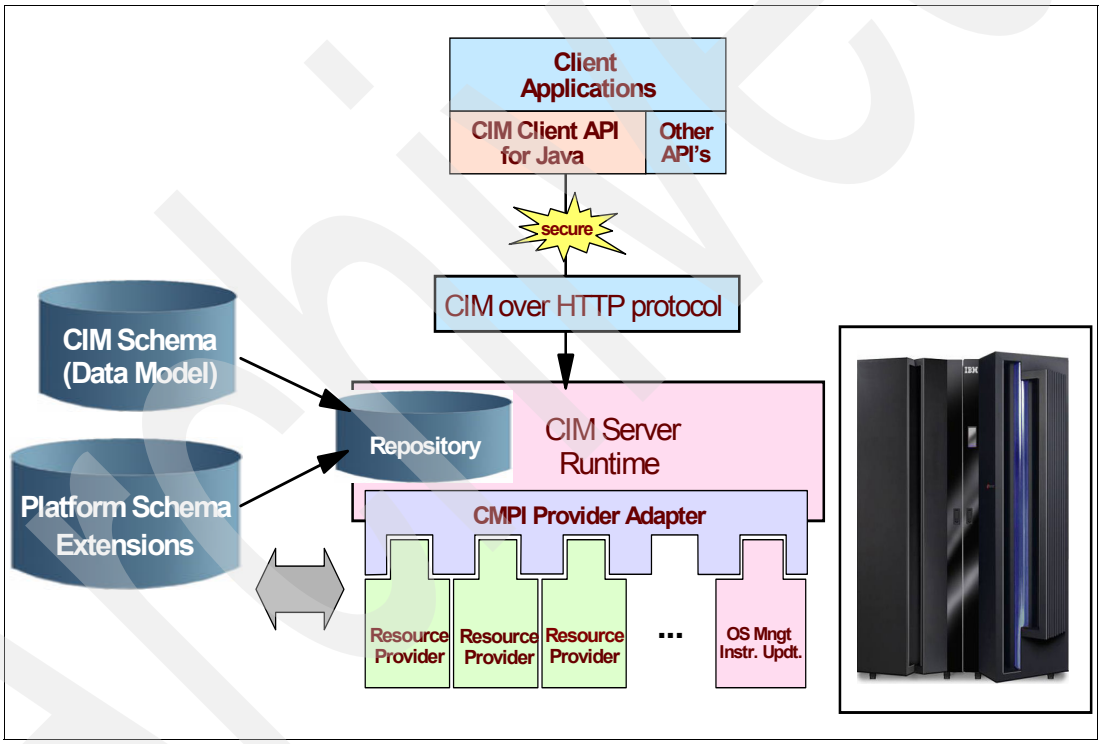


Figure 30-4 CIM Server overview

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 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 30-5, 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. 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
```

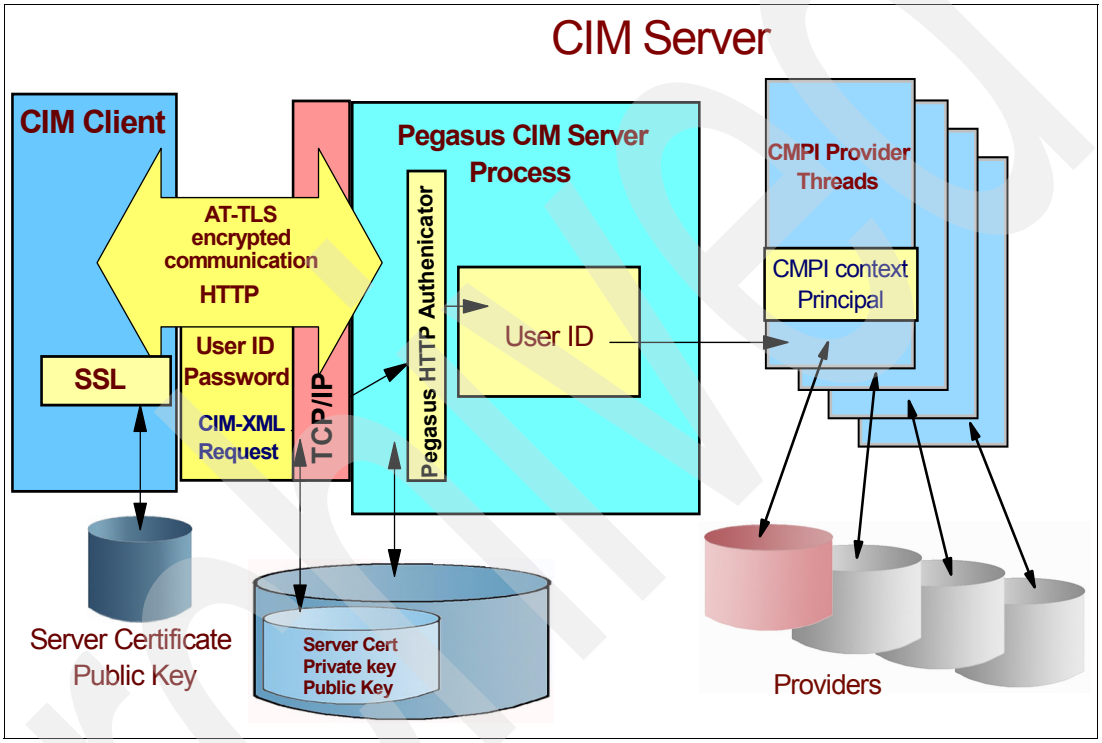


Figure 30-5 CIM client logging on to a CIM Server

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 is to 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 is used and an IEF695I message indicates 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.

30.1.4 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. SSL certificates, as well as encryption, can be used 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 30-5 on page 603. The CIM client sends an SSL certificate to AT-TLS, then AT-TLS sends the certificate to RACF, and RACF associates the certificate to the appropriate user ID, which then can access the CIM Server. Conversely, 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. 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 30-5 on page 603.

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.

30.1.5 CIM enablement

z/OS CIM comes with the z/OS package and is installed through 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 various environment settings in z/OS UNIX. All of this is described in *z/OS Common Information Model User's Guide*, SC33-7998.

Migrating from previous releases z/OS V1R8 or z/OS V1R9 is also very easy to do. There are several minor considerations that can be applicable to your own configuration that need to be checked before starting the CIM Server on z/OS V1R10 or higher. The migration itself will be performed 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 current. If it is back level, the CIM Server automatically upgrades the repository as described here.

1. The CIM Server backs up the current repository.
2. The CIM Server migrates the previous repository content to the current repository.

Note that 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
CFZ12562I: Previous repository was renamed to "/var/wbem/repository_old19_2010
CFZ12563I: Automatic repository upgrade completed successfully.
CFZ12580I: CIM server running eligible for zIIP.
CFZ17204I: CIM server authentication is using application ID OMVSAPPL.
CFZ00001I: PGS10042: IPv6 stack is not active, using IPv4 socket.
CFZ10025I: The CIM server is listening on HTTP port 5988.
CFZ10028I: The CIM server is listening on the local connection socket.
CFZ10030I: Started CIM Server version 2.8.1.
CFZ12532I: The CIM server successfully registered to ARM using element name
CFZ_SRV_SC70
CFZ12563I: Automatic repository upgrade completed successfully.
CFZ12580I: CIM server running eligible for zIIP.
CFZ17204I: CIM server authentication is using application ID OMVSAPPL.
CFZ00001I: PGS10042: IPv6 stack is not active, using IPv4 socket.
CFZ10025I: The CIM server is listening on HTTP port 5988.
CFZ10028I: The CIM server is listening on the local connection socket.
CFZ10030I: Started CIM Server version 2.8.1.
CFZ12532I: The CIM server successfully registered to ARM using element name CF
```

Figure 30-6 CIM Server startup

30.1.6 Automatic Restart Manager support

The CIM Server (only if started as a started task) automatically registers with Automatic Restart Manager, 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; see Figure 30-7 on page 606.

```
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 30-7 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
```

30.1.7 SSL certificate-based authentication

The CIM Server provides an additional authentication mechanism using certificates, as shown in Figure 30-8 on page 607. The CIM Server uses the z/OS Communications Server Application Transport TLS (AT-TLS). 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) that does the following:

- It stores CA and CIM Server certificates into key-rings.
- It maps certificate subjects to a z/OS user ID, either one-to-one or many-to-one.

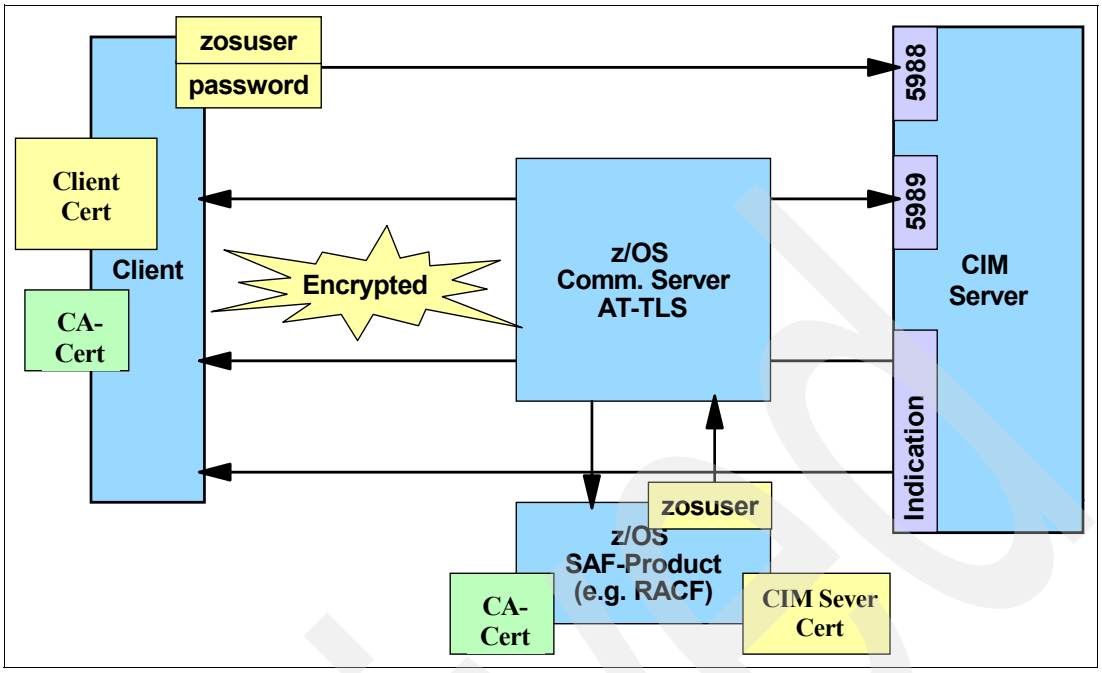


Figure 30-8 SSL certificate-based authentication

SSL certificate-based authentication configuration

The SSL certificate-based authentication configuration is executed through the **`cimconfig`** command.

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.

30.1.8 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 30-9 on page 608. This is expected to enable vendors and other exploiters to write CIM client applications in Java on z/OS.

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).

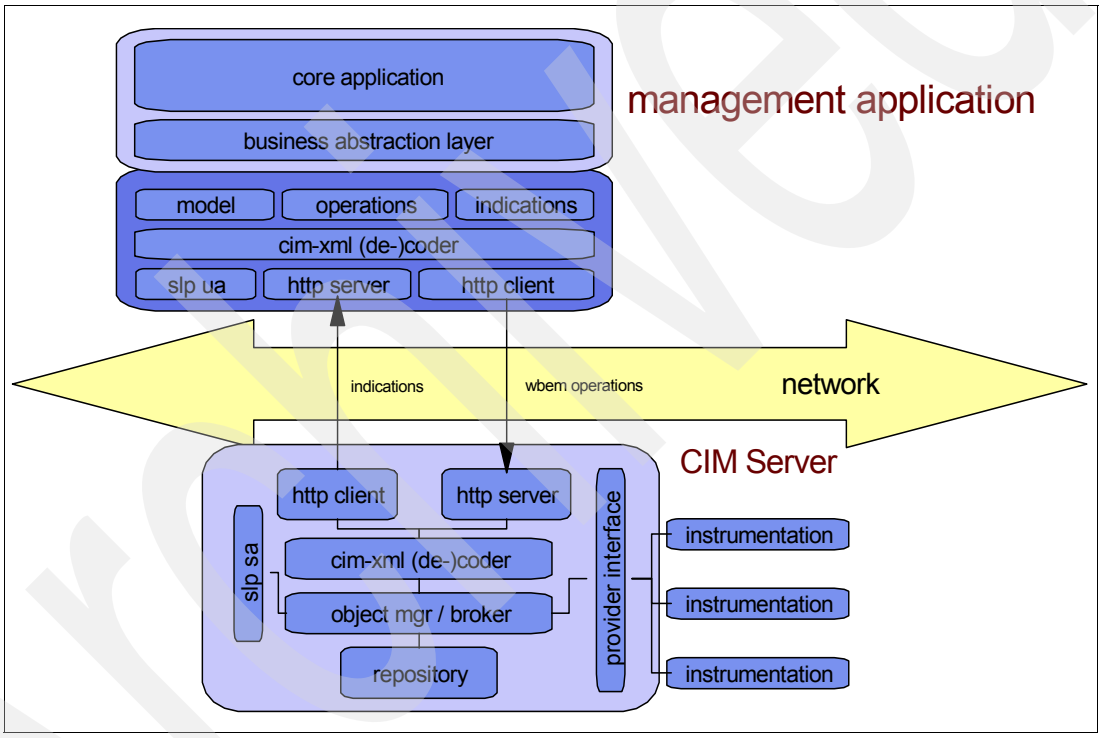


Figure 30-9 CIM client API for Java

30.1.9 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 30-10 on page 609 illustrates how the CIM Server works in the z/OS environment:

1. A CIM client application requests the CIM Server to return information about z/OS resources, in this case about basic operating system 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 30-10 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

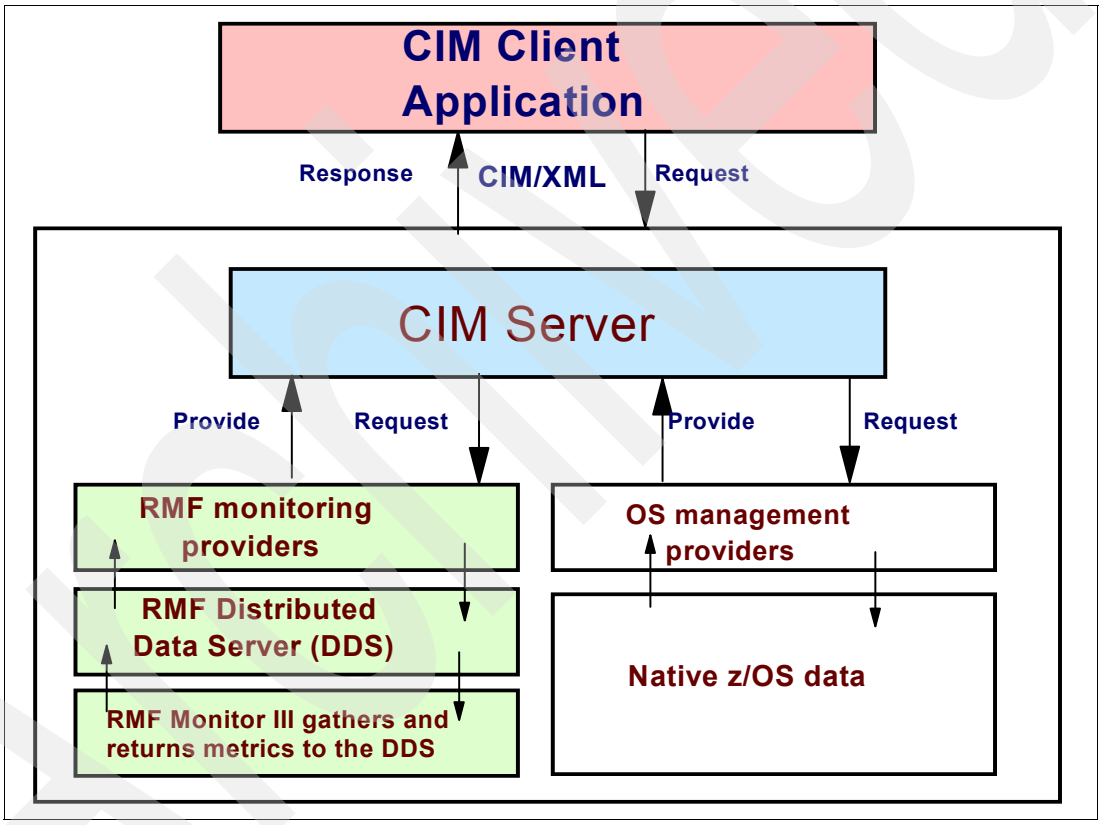


Figure 30-10 Client/server requests for data to z/OS using CIM

30.1.10 Required parmlib updates

The following parmlib parameters must be defined to enable the job and cluster providers:

- MAXCAD limit

  This parameter defaults to 50. If the installation sets a lower limit, it can be necessary to increase this setting to accommodate the Common Event Adapter (CEA) common area data space (CADS).

- APF authorize SYS1.MIGLIB

  The following statement 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), and to permit CEA to operate in UNIX System Services with the cluster, couple data set, and JES2 and JES3 jobs, and CIM providers.

30.1.11 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 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 the 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. Note that **`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.

Avoid using the **`cimmofl`** command, because it bypasses the CIM Server and directly manipulates files in the file system.

30.2 CIM server upgrade to OpenPegasus 2.8

The following aspects have been developed for CIM in z/OS V1R11:

- Local and internal binary protocol
- Tracer improvements for in-memory tracing
- IPV6 support

30.2.1 Local and internal binary protocol

To address this performance issue, local and internal binary protocol has been revisited as depicted in Figure 30-11 on page 611.

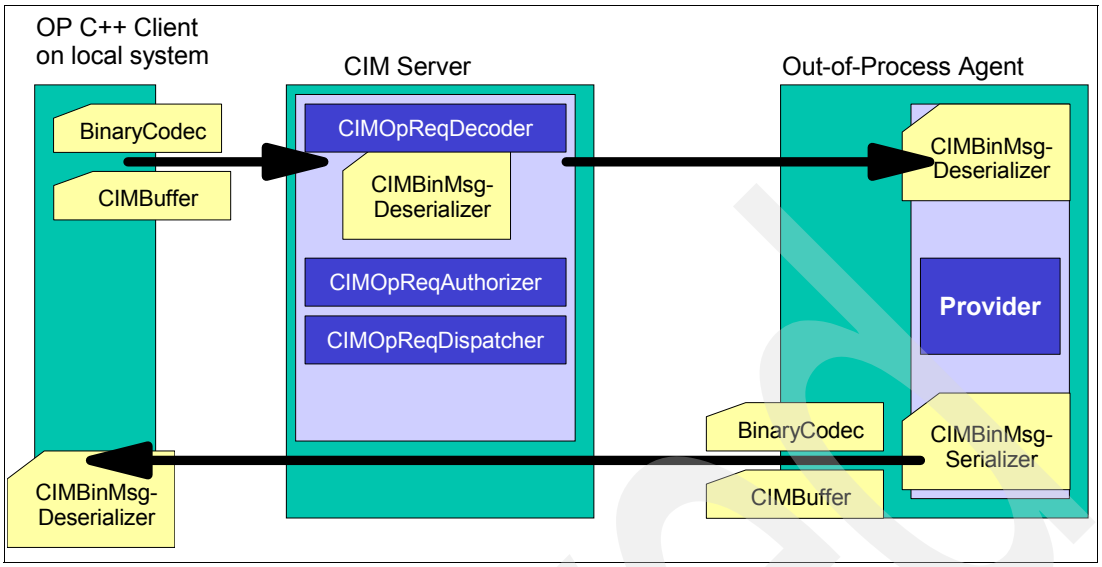


Figure 30-11 Mainline logic overview – binary protocol

C++ code is being reimplemented in libpegcommon.so (Pegasus/Common) such as CIMBinMsgDeserializer, CIMBinMsgSerializer, CIMBuffer(Object encoding) and BinaryCodec (Message encoding).

30.2.2 Tracer improvements and in-memory tracing

For serviceability reasons, first failure data capture is needed. The objective of this enhancement in z/OS V1R11 is to keep error trace information in memory available at all times.

By default, the new feature of the CIM Server to trace non-critical errors (which do not require a CIM Server shutdown or stopping working operations) to memory is enabled. This feature allows you, in case of a critical error, to track down the cause for an error without having to reproduce the error condition.

The amount of memory used for this mechanism can be changed, and the feature also can be disabled dynamically. The configuration of the CIM Server can be changed by using the **`MODIFY`** command on the console, or by using the **`cimconfig`** command in a UNIX System Services shell. For a more detailed description of how to use these commands, see *z/OS Common Information User's Guide*, SC33-7998, at the z/OS V1R11 level.

The new configuration property "traceFacility" is used to determine where traces are written to. You can configure trace to be written to a file (as supported by the CIM Server since z/OS V1R7), or to memory.

To disable tracing altogether, you can set the traceLevel to 0. You can increase or decrease the size of the memory buffer used for tracing by configuring the configuration option traceMemoryBufferKbytes. traceMemoryBufferKbytes specifies the size of the memory area, which is reserved for tracing messages in kB (1 kB = 1024B). The default is 10240. The value must be at least 16. traceMemoryBufferKbytes is a planned configuration property.

The following are new configuration properties:

- traceFacility (Destination of trace, file or memory)
- traceMemoryBufferKbytes

Five trace levels are now possible and the new default values are as follows:

- traceLevel=2
- traceComponents=All
- traceFacility=Memory
- traceMemoryBufferKbytes=10240

As shown in Figure 30-12, UNIX System Services commands can be used to switch tracing to full trace to a file and then to restart the cimserver.

```
Commands to switch tracing to full trace to a file
cimconfig –p –s traceFacility=File
cimconfig –p –s traceFilePath=/tmp/cim.trc
cimconfig –p –s traceComponents=All
cimconfig –p –s traceLevel=5
Start the cimserver
```

Figure 30-12 Commands to switch tracing

Trace is written in plain-text (ASCII), and is wrapped around. Its size is configured in traceMemoryBufferKbytes, as shown in Figure 30-13.

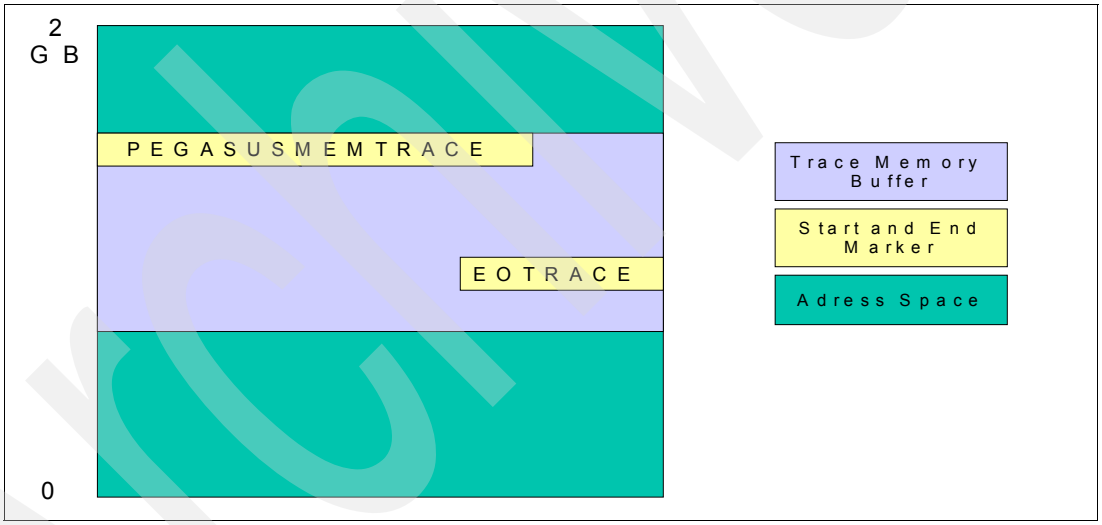


Figure 30-13 Diagnostics for in memory tracing

30.2.3 IPV6 support

To comply with government IPV6 requirements, IPV6 support has been added to CIM server in z/OS V1R11. This enhancement of the CIM server in z/OS V1R11 gives full IPV6 support, as well as full IPV6 and IPV4 coexistence.

30.3 CIM client for Java upgrade

The CIM Client for Java has been upgraded to the latest stable version 2.0.8. This version introduces several fixes and increases both client stability and standards compliance.

30.4 Installation consideration

As part of the enhancements of CIM in z/OS V1R11, the CHOWN.UNRESTRICTED and SUPERUSER.FILESYS.CHOWN profiles are no longer needed for the CIM server

30.5 Migration for service location protocol

Configuration default for Service Location Protocol (SLP) has changed with z/OS V1R11. To re-enable SLP, use cimconfig under OMVS or MODIFY command as shown in Figure 30-14.

```
F CFZCIM,APPL=CONFIG,SLP=TRUE,PLANNED
CFZ06209I: Updated planned value for slp to true
CFZ06210I: This change will become effective after CIM Server restart.
```

Figure 30-14 SLP re-enabling

31

XCF enhancements

This chapter discusses the XCF enhancements provided with z/OS V1R11:

- Sysplex partitioning enhancements using BCPii
- XCF FDI consistency
- Pending delete CF structure cleanup
- AutoIPL update
- Enhanced timer support for GDPS

31.1 Sysplex partitioning enhancements using BCPii

In a z/OS Parallel Sysplex, unresponsive systems must be partitioned from the sysplex in a timely manner to avoid both corruption of shared data and causing any “sympathy sickness” in the surviving systems of the sysplex.

z/OS systems occasionally experience critical situations where a system either enters a wait state or becomes “Status Update Missing” and is generally non-responsive. It could be hung, not processing work or looping. The removal of unresponsive systems from the sysplex is the responsibility of the z/OS Cross-System Coupling Facility (XCF) component of z/OS.

The Sysplex Failure Management (SFM) sub-component of XCF extends the base XCF sysplex services in this area by providing a policy-based mechanism for improved automation of the removal of systems from the sysplex.

A weakness of current sysplex partitioning support is that today, XCF does not really know the state of the system that is in an unknown, “status-update missing” state. All that is known is that the system looks unresponsive because it is not performing system status updates. Because of this, XCF monitors the system for an interval of time specified by the failure detection interval (FDI) to allow the system some time in which to resume normal operation.

The penalty of waiting this time interval is that while XCF is waiting to see if the system will resume normal operation, the sysplex may suffer some workload processing interruption for a period of time dictated by the system’s failure detection interval. In most sysplexes, this failure detection interval is set on the order of a minute and a half to two minutes.

There are limitations that constrain XCF’s ability to remove an ailing system from the sysplex in a timely manner. When operator intervention is required, the delay can extend to many minutes. XCF cannot unambiguously determine that a system is truly dead. XCF must infer that it is appropriate to partition a system based on the amount of time elapsed since that system has updated its heartbeat in the sysplex couple data set (CDS) or sent XCF signals.

XCF relies on the Coupling Facility (CF) **`isolate`** command to fence a target system from the channel subsystem. Isolation will ensure data integrity when partitioning a system. However, isolation may fail for many reasons and when it does, XCF has no choice but to prompt the operator to perform a manual system reset.

z/OS V1R11 XCF enhancements

New function has been added to z/OS V1R11 to allow XCF to obtain certain knowledge of the state of another image in the sysplex to eliminate these constraints to the sysplex partitioning process. These enhancements to the partitioning protocol for removing a system from the system are called System Status Detection (SSD) Partitioning Protocol. When this function is enabled, the monitoring system detects changes in target system status and may thereby be able to bypass certain delays built into the status monitoring and partitioning processes.

This support also enables the system to initiate actions, such as reset-normal, against the remote system to ensure that it is no longer actively processing. It provides new support in XCF to exploit the HMC remote query and remote reset functions, accessed through the BCPii APIs, to expedite and enhance the sysplex partitioning process.

The BCPii support being implemented in z/OS V1R11 provides mechanisms by which XCF can, in most cases, eliminate these constraints. BCPii’s query capabilities will allow XCF to determine whether a system is demised, without waiting for installation-specified intervals to elapse.

A system is considered demised based on the following criteria:

- When the system enters a non-restartable wait state.
- The system has been reset or reloaded.
- The partition has been deactivated or check stopped.

BCPii's remote command capability will provide XCF an alternative to isolation, thereby allowing XCF to reset an image when fencing fails for a demised system.

31.1.1 Benefits of System Status Detection Partitioning Protocol

When the System Status Detection Partitioning Protocol is enabled, XCF will use the BCPii APIs to expedite and enhance the sysplex partitioning process by:

- Bypassing the failure detection interval (FDI)
- Avoiding IXC102A, IXC402D, and IXC409D manual intervention
- Bypassing the cleanup interval
- Validating a "DOWN" response for messages IXC102A and IXC402D

These delays in sysplex partitioning and why they can be bypassed or avoided are discussed in the following sections.

Bypassing the failure detection interval

The sysplex failure detection interval (FDI) is used to indicate how long a system may appear unresponsive before other systems in the sysplex should start to take disruptive actions against that system. It is specified using the INTERVAL parameter in the COUPLExx parmlib member.

By using the BCPii remote query capabilities to discover when another system has become demised, XCF can immediately initiate sysplex partitioning without waiting for the FDI to elapse if the system can be determined to be demised. This will typically speed up partitioning by a few minutes.

Avoiding IXC402D manual intervention

Message IXC402D is normally issued when a system has neither updated its "heartbeat" in the sysplex CDS nor sent XCF signals within the FDI, and the SFM policy specifies an indeterminate status action of PROMPT.

Issuing IXC402D will be avoided under the following conditions.

- When XCF can determine through BCPii APIs that the unresponsive system is truly demised, then XCF may bypass the FDI and immediately proceed with partitioning without issuing the IXC402D WTOR.
- When XCF cannot establish that the unresponsive system is demised, then XCF will use the BCPii remote reset capability to perform a system reset against the unresponsive system without issuing the WTOR unless the SFM Policy specified PROMPT.

Avoiding IXC102A manual intervention

Message IXC102A is normally issued when SFM cannot successfully isolate the system image being removed and manual intervention is required. XCF will use the BCPii remote reset capability to perform a system reset against the system to be isolated without issuing the WTOR unless the SFM Policy specified PROMPT.

Bypassing the cleanup interval

When varying a system out of the sysplex in response to an operator command, XCF will use the BCPii query capability to determine when the target system becomes demised. At that point, it is no longer necessary to wait for the cleanup interval to elapse because no further cleanup is taking place on the target. XCF can immediately continue with partitioning processing. This will speed up partitioning by whatever portion of the cleanup interval is not actually required for cleanup, perhaps 10 to 20 seconds.

Validating a DOWN response for messages IXC102A and IXC402D

Messages IXC102A and IXC402D invite the operator to reset an unresponsive system and to reply `DOWN` when the reset is complete. It is critical that the operator perform the reset *before* responding to the WTOR. The reason for this is because as soon as the response is received, XCF proceeds with partitioning, freeing other systems to take over resources previously owned by the outgoing system. If the target system has not in fact been reset, and is still attempting to manipulate those resources, data integrity problems can result.

With this support, XCF will use BCPii query services to verify that the target system has in fact been reset before proceeding with partitioning, and will reprompt the operator if it has not.

31.1.2 System status detection partitioning protocol overview

During XCF initialization, an SSD monitor task is attached and a control block is updated with system information including an IPL token, the CPC name, the image name and BCPii connector tokens. An ENF listener requests notification of ENF 68 signals from BCPii and the local system's SSD eligibility status is updated. This determines whether a system can use BCPii to take action against another system, as well as whether other systems can take action against the local system.

Every system updates its heartbeat in the sysplex CDS every three seconds and looks at the status of the other systems. If the SSD protocol detects that a system has not updated its heartbeat for at least six seconds, it will issue a BCPii query to determine the status of that system. If the response to the BCPii query shows that the system is demised, the SSD protocol will take action to remove the system from the sysplex.

31.1.3 Enabling system status detection partitioning protocol

The following tasks and prerequisite hardware and software are required to enable the SSD partitioning protocol.

- The XCF address space must have adequate authorization through the z/OS Security Server (RACF) or an equivalent security product to access BCPii-defined FACILITY class resources.

  See "Assigning the RACF TRUSTED attribute" in *z/OS MVS Initialization and Tuning Reference*, SA22-7592, for information about using RACF to assign the TRUSTED attribute to the XCF address space.

- Configure BCPii to invoke z/Series hardware APIs.

  See "BCPii Setup and Installation" in *z/OS MVS Programming: Callable Services for High Level Languages*, SA22-7613, for information about installation and configuration steps and SAF authorization requirements to enable BCPii to invoke z/Series Hardware APIs.

- The system must operate on a z10 EC GA2 at Driver-76 or newer hardware and be at z/OS V1R11 to take full advantage of the functionality associated with the system status detection partitioning protocol.

  - A system running on z/OS V1R11 and downlevel hardware is only eligible to target other systems that are enabled to exploit the full functionality of the SSD partitioning protocol.
  - A system not running on the requisite hardware cannot be the target of SSD partitioning protocol functions.
- A sysplex couple data set formatted using the ITEM NAME(SSTATDET) NUMBER(1) control statement must be the primary couple data set in service in the sysplex.
- Install toleration PTFs for OA26037 on z/OS V1R10 and V1R9 systems in the sysplex to use the newly formatted sysplex couple data set required by the protocol.
- Ensure that the optional SYSSTATDETECT function is enabled on z/OS V1R10 and V1R9 systems. The SYSSTATDETECT function is enabled by default in z/OS V1R11.

31.1.4 Enabling the SYSSTATDETECT function

The SYSSTATDETECT function can be enabled using the COUPLExx parmlib member, and dynamically by using an operator command.

The setting of the SYSSTATDETECT function on a given system will govern both that system's use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems.

Whenever a system detects that any of the environmental factors preventing the enablement of the SSD partitioning protocol have changed, it will attempt to enable the protocol. In practice, this means detecting either the availability of a version 5 sysplex CDS or a change in the installation controls, because the other environmental factors will not change without an IPL.

- When either of these factors changes, the system will report the status of the protocol with message IXC104I.
- The current setting of the SYSSTATDETECT function can be determined by issuing a **DISPLAY XCF,COUPLE** command.

New parmlib changes

The FUNCTIONS statement allows you to enable and disable the SYSSTATDETECT function:

- FUNCTIONS ENABLE(SYSSTATDETECT)
- FUNCTIONS DISABLE(SYSSTATDETECT)

The default is ENABLE.

Dynamic modifications

The **SETXCF FUNCTIONS** command can be used to enable SYSSTATDETECT if a system IPLs with the SYSSTATDETECT function disabled as follows:

- SETXCF FUNCTIONS,ENABLE=SYSSTATDETECT

The SYSSTATDETECT PROTOCOL requires a SYSPLEX CDS formatted with SSTATDET. Formatting a sysplex couple data set to support the SSD protocol will increment the version number associated with the couple data set and expand the System Record. The expanded System Record will contain information for the local system as well as information for each of the other remote systems in the sysplex that enables the SSD partitioning algorithms to function.

The **SETXCF COUPLE ACOUPLE** command to define an alternate sysplex couple data set that is formatted to support the SSD protocol and SETXCF COUPLE PSWITCH to bring the alternate sysplex couple data set into service.

A **DISPLAY XCF,COUPLE** command displays summary information about the couple data sets, including:

- Information about the primary and alternate data sets including Sysplex Failure Management-related information
- Information about the system parameters set by the COUPLE statement in the COUPLExx parmlib member
- Information related to the system status detection partitioning protocol.
- The status of optional functions.

Figure 31-1 shows the format of the **DISPLAY XCF,COUPLE** command.

```
IXC357I  hh.mm.ss  DISPLAY XCF
SYSTEM sysname DATA
SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY:
SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS.

  [REASON: targetotherrsn]
SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
  [REASON: othertargetrsn]
SYSTEM NODE DESCRIPTOR: type.mfg.plant.sequence
             PARTITION: partition side CPCID: cpcid
SYSTEM IDENTIFIER: sysid
NETWORK ADDRESS: snaaddr
PARTITION IMAGE NAME: image
IPL TOKEN: ipltoken
COUPLEXX PARMLIB MEMBER USED AT IPL: COUPLExx OPTIONAL FUNCTION STATUS:
FUNCTION NAME         STATUS              DEFAULT
SYSSTATDETECT      ENABLED             ENABLED
SYSPLEX COUPLE DATA SETS
PRIMARY   DSN: dsname
          VOLSER: prisysvol     DEVN: prisysdev
          FORMAT TOD            MAXSYSTEMMAXGROUP(PEAK)     MAXMEMBER(PEAK)
          mm/dd/yyy hh:mm:ss    maxsys  maxgroup(peakgrp) maxmember(peakmem)
          ADDITIONAL INFORMATION:
        SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED
```

Figure 31-1 Format of the D XCF,COUPLE command

Where:

`SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS`

- Indicates whether this system can employ the system status detection partitioning protocol when removing other systems from the sysplex.
- If the system status detection partitioning protocol is not enabled and enablement is desired, see the description for message IXC104I for the action that is required to correct the limiting factor.

Where:

`SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS:`

- Indicates whether other systems can employ the system status detection partitioning protocol when removing this system from the sysplex.

  ```
  SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED
  ```

- The couple data set supports the system status detection partitioning protocol.
- If the system status detection partitioning protocol is not enabled and enablement is desired, see the description for message IXC104I for the actions that are required to correct the limiting factor.

Adding SSTATDET to the sysplex CDS

The sysplex CDS must be formatted to include the ITEM NAME(SSTATDET) NUMBER(1) statement by using the IXCL1DSU CDS format utility. Figure 31-2 shows the syntax of DEFINEDS that includes SSTATDET. This formats the sysplex CDS with the larger system records required by the system status detection partitioning protocol.

```
DEFINEDS
  SYSPLEX(sysplexname)
  MAXSYSTEM(maxsystems)
  DSN(dsname) [VOLSER(volser)] [UNIT(unit)]
  [CATALOG | NOCATALOG]
  [STORCLAS(storclass)] [MGMTCLAS(mgmtclass)]
  DATA TYPE(SYSPLEX)
    [ITEM NAME(GROUP) NUMBER(groupnum)]
    [ITEM NAME(MEMBER) NUMBER(memnum)]
    [ITEM NAME(GRS) NUMBER(1)]
    [ITEM NAME(CLUSTER) NUMBER(1)]
    [ITEM NAME(SSTATDET) NUMBER(1)]
```

Figure 31-2 Syntax of DEFINEDS including SSTATDET

As always, after a couple data set formatted with larger records is brought into service, it requires a sysplex-wide IPL to revert to a less-capable couple data set. However, there should be no reason to do this because the installation controls permit disabling the SSD partitioning protocol if necessary, without reverting to the older CDS format.

Restriction: Formatting a sysplex couple data set to support this protocol increments the version number associated with the couple data set. Such a couple data set cannot be brought into service if there are any systems active in the sysplex that do not have the code (either z/OS V1R11 or the toleration PTF) to accommodate the higher version number. Similarly, after such a couple data set is in use, no system without the code to accommodate it can join the sysplex.

Migration and coexistence considerations

The SSD partitioning protocol support ships as an integral part of z/OS V1R11. There are no plans to roll back the new function provided by this support. As soon as two or more systems are running z/OS V1R11, those systems can begin exploiting the SSD partitioning protocol, assuming hardware environmental requirements are met. However, the new function will not be completely effective until all systems in the sysplex are at or above z/OS V1R11.

The SYSSTATDETECT PROTOCOL requires a SYSPLEX CDS formatted with SSTATDET. Because the enhanced partitioning depends on the availability of each system's addressing information to every other system in the sysplex, no system can apply the SSD partitioning algorithms to bypass FDI or cleanup interval, initiate remote reset, and so on to any other

system unless the sysplex is operating with sysplex CDSes containing the expanded system record. The addressing information is also necessary for correlating systems with the CPC on which each resides. Therefore, even the case in which a CPC is observed to have checkstopped requires the larger CDS.

Because the expansion of the system record is an incompatible change, support will be required to accommodate it. Systems running z/OS V1R11 will understand the new, larger record. A toleration PTF is provided for releases down to z/OS V1R9 to enable downlevel systems to process the larger record and higher version number. APAR OA26037 provides the toleration support for z/OS V1R9 and V1R10.

Even though downlevel systems will not be able to actually use the SSTATDETECT protocols, this will allow the z/OS V1R11 systems in the sysplex to use the new protocols, while the z/OS V1R9 and V1R10 systems share the sysplex CDS.

Toleration support of the SSD partitioning support will eliminate the requirement that the entire sysplex consist of systems at or above z/OS V1R11 to use it. The toleration PTF will not roll down any part of the SSD partitioning function. Its sole purpose will be to allow a mixed sysplex to bring a version 5 sysplex CDS into service.

New functions, particularly those that are potentially disruptive, are shipped with switches to permit an installation to disable the function if desired. The Sysplex Partitioning Enhancements function includes an enabling switch. Installation control will be by means of the COUPLExx OPTIONS statement in SYS1.PARMLIB(COUPLExx) or via the **SETXCF OPTIONS** command.

Because explicit action is required to bring version 5 sysplex CDSes into service, it is not necessary to require the installation to take additional action to enable the new support using these installation controls. By default, the SSD function will be enabled as soon as the necessary environmental requirements are met. However, an installation may choose to disable it in the event of a defect, or for diagnostic purposes, or from an excess of caution. The setting of the switch on a given system will govern both that system's use of the SSD partitioning algorithms and its eligibility as a target for SSD partitioning initiated by other systems

Note: The SYSSTATDETECT PROTOCOL also requires BCPii to be properly configured and services active.

Hardware requirements

Each system must set an IPL token that uniquely identifies it. The ability to store and reset the IPL token will be introduced with z10 GA2, which means that systems not running on that hardware cannot establish a token. Any such system is not eligible to be targeted by the remote reset command. In addition, although query commands can be directed to systems on downlevel hardware, XCF will rely on observing changes in the IPL token to determine whether a system can be partitioned. Therefore, systems running on pre-GA2 hardware cannot be targeted by the SSD partitioning processing. The support provided by this line item will not be completely effective unless all systems on the sysplex are operating on z10 GA2 hardware with the following MCLs:

- MCL129 in EC Stream N10970
- Driver-76 HMC, MCL046 in EC Stream N10976

The system status detection partitioning protocol will automatically limit its actions when the hardware support is not present; therefore, no additional support is needed to disable the new

function in the absence of z10 GA2 hardware. The z10 GA2 hardware will also provide new information when a resident image enters a wait state.

Restriction: The BCPii command interfaces, including queries directed to individual images, operate against an individual partition. Under VM, there can be multiple z/OS guests running in a single partition. BCPii therefore does not support the VM environment and will not start in a z/OS image running as a guest under VM.

31.1.5 New messages and updated messages

There are a number of new messages and updated messages associated with the SSD Partitioning Protocol support, as listed in Figure 31-3. These messages are explained in the following sections.

Updated Messages	New Messages
IXC101I	IXC103I - Hardcopy
IXC105I	IXC104I
IXC335I	IXC106I
IXC357I	IXC107E
IXC358I	IXC108I - Hardcopy
IXC208I	IXC109I - Hardcopy
IXC409D	IXC111I - Hardcopy
IXC422I	IXC112I - Hardcopy
IXC602I	IXC113I - Hardcopy
IXC603I	IXC114I - Hardcopy
IXC611I	
IXC373I	

Figure 31-3 List of new messages and updated messages

New message IXC103I

The new IXC103I message is shown in Figure 31-4.

```
IXC103I SYSTEM IDENTIFICATION INFORMATION information
In the message, information is:
CONNECTION STATUS: {CONNECTED|NOT CONNECTED}
 SYSTEM NAME: sysname
 SYSTEM NUMBER: sysnum
 IMAGE NAME: image
 NODE DESCRIPTOR: type.mfg.plant.sequence
 PARTITION NUMBER: partition
 CPC ID: cpcid
 NETWORK ADDRESS: netid.nau
 IPL TOKEN: ipltoken

Explanation: The message displays the identification information about system
status detection partitioning protocol associated with this system
```

Figure 31-4 IXC103I message

The message displays the identification information about system status detection partitioning protocol associated with this system. This message is issued when the system performs initialization processing to become enabled to target other systems and be targeted

by other systems using the system status detection partitioning protocol. As part of the initialization process, the system establishes a logical application connection to BCPii to issue remote hardware management console API commands against other systems that are eligible targets of the system status detection partitioning protocol.

New message IXC104I

The new IXC104I message is shown in Figure 31-5.

```
IXC104I SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY: information
In the message, information is:
 SYSTEM {CAN | CANNOT} TARGET OTHER SYSTEMS.
 [REASON: targetotherrsn]
 SYSTEM {IS | IS NOT} ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
 [REASON: othertargetrsn]

Explanation: This message indicates whether the SSD protocol is enabled on this
system and the state.
 CPC ID: cpcid
 NETWORK ADDRESS: netid.nau
 IPL TOKEN: ipltoken

Explanation: The message displays the identification information about system
status detection partitioning protocol associated with this system
```

Figure 31-5 IXC104I message

The IXC104I message displays whether this system can employ the system status detection partitioning protocol when removing other systems from the sysplex. `targetotherrsn` is the reason why this system cannot use the system status detection partitioning protocol to aid in removing other systems from the sysplex.

One of the following reasons applies:

- SYSPLEX COUPLE DATA SET NOT FORMATTED TO SUPPORT PROTOCOL
- NOT ENABLED BY INSTALLATION
- OPERATING AS VM GUEST
- BCPII SERVICES NOT AVAILABLE
- SYSTEM OR HARDWARE ERROR
- INSUFFICIENT SAF RESOURCE ACCESS AUTHORITY
- UNEXPECTED SYSTEM SERVICE ERROR
- PROTOCOL NOT APPLICABLE IN MONOPLEX MODE
- PROTOCOL NOT APPLICABLE IN XCF-LOCAL MODE

Updated message IXC105I

Message IXC105I, shown in Figure 31-6 on page 625, has been updated to add new reason codes.

```
IXC105I  SYSPLEX PARTITIONING HAS COMPLETED FOR sysname
 PRIMARY REASON: reason
 REASON FLAGS: flags

reason
 SYSTEM ENTERED WAIT STATE (new use)
 SYSTEM RESET OR NEW IMAGE LOADED
 PARTITION DEACTIVATED
 CPC FAILURE
```

Figure 31-6 IXC105I message

The detection of a demised system by SSD results in partitioning with one of the mentioned partitioning reasons. SYSTEM ENTERED WAIT STATE is now also used for when a wait state is detected using the SSD protocol.

For AutoIPL, the reason will be as follows:

```
SYSTEM RESET OR NEW IMAGE LOADED.
```

The flag masks are as follows:

```
00000001 LOSS OF COUPLE DATA SET
00000002 LOSS OF CONNECTIVITY
00000004 OPERATOR VARY REQUEST
00000008 SYSTEM STATUS UPDATE MISSING
00000010 LOSS OF ETR
00000020 SYSTEM ENTERED WAIT STATE
00000080 POLICY INITIATED REQUEST
00000100 SFM STARTED DUE TO STATUS UPDATE MISSING
00000200 SFM INITIATED DUE TO SIGNALLING FAILURE
00001000 A NEW SYSTEM IS DETECTED ON THE SAME CPC
00002000 INCOMPATIBLE LOGICAL PARTITION NUMBER
00004000 TIMING NOT SYNCHRONIZED WITH SYSPLEX
00008000 SYSTEM CAUSING SYMPATHY SICKNESS
00010000 OPERATOR VARY REQUEST WITH REIPL OPTION
00020000 OPERATOR VARY REQUEST WITH SADMP OPTION
00040000 OPERATOR VARY REQUEST WITH SADMP AND REIPL OPTIONS
00080000 SYSTEM RESET OR NEW IMAGE LOADED
00100000 CPC FAILURE
00200000 PARTITION DEACTIVATED
```

New message IXC106I

The new IXC106I message is shown in Figure 31-7 on page 626.

```
IXC106I SYSTEM sysname status
  ENTERED WAIT STATE The system was observed to have entered a non-restartable
   disabled wait state.
  RESET OR NEW IMAGE LOADED The partition containing the system has been reset
   or a new system image has been loaded in the partition it formerly occupied.
  CPC FAILED The central processing complex (CPC) on which the system was
   running has failed.
  PARTITION DEACTIVATED The central processing complex (CPC) LPAR on which the
   system was running has been deactivated.

Explanation: The system initiates partitioning to remove the named system from
the sysplex. Partitioning might be able to bypass portions of the partitioning
protocol because the named system is already known to have failed.
```

Figure 31-7 IXC106I message

The IXC106I message is issued when the system initiates partitioning to remove the named system from the sysplex. Partitioning might be able to bypass portions of the partitioning protocol because the named system is already known to have failed.

New message IXC107E

The new IXC107E message is shown in Figure 31-8.

```
IXC107E SYSTEM STATUS DETECTION PARTITIONING PROTOCOL CONFIGURATION IS NOT
COMPLETE .
One of the following conditions might exist:
 - The BCPii address space and BCPii services are not available.
 - The SYSSTATDETECT function is not enabled on the local system.
 - The local system is not able to connect to the local CPC and local CPC image
or a remote CPC and remote CPC image in the sysplex, although the local system
is eligible and enabled to use the system status detection partitioning
protocol.

Explanation: An exception condition has been detected on the local system that
prevents the system status partitioning protocol from being used to its fullest
capability in the sysplex by the local system
```

Figure 31-8 IXC107E message

If the system status detection partitioning protocol cannot be used, the system processes partitioning requests using a partitioning protocol based on the sysplex failure management (SFM) policy or default indeterminate status behavior. This message will be DOMed when the exceptions causing the configuration to be incomplete are corrected.

New message IXC108I

The new IXC108I message is shown in Figure 31-9 on page 627.

```
IXC108I SYSPLEX PARTITIONING INITIATING {FENCE|SYSTEM RESET}
  SYSTEM NAME: sysname
  SYSTEM NUMBER: sysnum
  [SYSTEM IDENTIFIER: sysid]
  [NETWORK ADDRESS: netid.nau]
  [IPL TOKEN: ipltoken]

Explanation: Action is being taken to ensure that a system being partitioned
from the sysplex no longer has the capability to perform I/O to devices that
can be used by another active system in the sysplex.
```

Figure 31-9 IXC108I message

In the message text shown in Figure 31-9:

- FENCE - the action is system isolation by a system fence through Coupling Facility fencing services. All systemS in the sysplex may attempt the system fence.
- SYSTEM RESET - the action is a SYSTEM RESET-NORMAL through the HWICMD BCPii callable service. Only the local system attempts the reset.

New message IXC109I

The new IXC109I message is shown in Figure 31-10.

```
IXC109I {FENCE | SYSTEM RESET} OF SYSTEM sysname
  {SUCCESSFUL. | RESULTS: | TIMED OUT.}
  [{HWICMD|HWMCA_EVENT_COMMAND_RESPONSE}
  RETURN CODE: retcode] text

Explanation: Action was taken to ensure that a system being partitioned from
the sysplex no longer has the capability to perform I/O to devices that can be
used by another active system in the sysplex. Message IXC108I was issued when
the action was initiated. This message is issued when the results of the action
are available or the action times out.
```

Figure 31-10 IXC109I message

In the message text:

- FENCE - the action is system isolation by a system fence through Coupling Facility fencing services.
- SYSTEM RESET - the action is a SYSTEM RESET-NORMAL through the HWICMD BCPii callable service.

  sysname - the name of the system targeted by the action.
- SUCCESSFUL - the action completed successfully.
- RESULTS - the action resulted in the indicated return code.
- TIMED OUT - the action did not complete within the allotted time. Processing continues without the result of the action.
- HWICMD - the action resulted in the indicated HWICMD BCPii callable service return code.
- HWMCA_EVENT_COMMAND_RESPONSE - the action resulted in the indicated ENF68 command response return code.

retcode - the hexadecimal return code from the indicated source. text One of the following: blank No additional information is provided.

- XCFAS DOES NOT HAVE SAF AUTHORIZATION - XCFAS does not have SAF authorization to RESET-NORMAL the target system. (This is no longer required because XCF is a trusted user.)
- System action:
  - When the action is successful, partitioning can complete and message IXC105I will be issued.
  - When the action is not successful, operator intervention might be required to complete partitioning as indicated by message IXC102A being issued, or another action can be attempted as indicated by message IXC108I being issued. For example, a SYSTEM RESET may be attempted after a FENCE times out.

New message IXC111I

The new IXC111I message is shown in Figure 31-11.

```
IXC111I LOGICAL PARTITION REMOTE CONNECTION INFORMATION information
  CONNECTION STATUS: {CONNECTED | NOT CONNECTED}
    SYSTEM NAME: sysname
    SYSTEM NUMBER: sysnum
    IMAGE NAME: image
    NETWORK ADDRESS: netid.nau
    IPL TOKEN: ipltoken
    DIAG INFO: diaginfo
Explanation:The message displays identification information associated with a
system and the local system connection status to that system for the purposes
of employing the system status detection partitioning protocol.
```

Figure 31-11 IXC111I message

The message displays identification information associated with a system in the sysplex, and the local system connection status to that system for the purposes of employing the system status detection partitioning protocol.

In the message text:

- CONNECTED - the local system is connected to remote CPC image through BCPii callable services.
- NOT CONNECTED - the local system is not connected to remote CPC image through BCPii callable services. The local system cannot target remote CPC image with the system status detection partitioning protocol.
- diaginfo - the diagnostic information applicable when the connection status indicates that the local system is not connected to the remote system sysname and CPC image through BCPii callable services. The diagnostic data returned by the BCPii callable service can help determine the cause of the failed connection request. The diagnostic information contains the following data:
  - The name of the BCPii callable service that failed when the local system attempted to establish a connection to another CPC or image in the sysplex.
  - An error return code that was returned by the BCPii callable service that failed while the local system was attempting to establish a connection to a CPC or image. See z/OS MVS Programming: Callable Services for HLL for more information about BCPii services return codes and actions to take in the event of a specific return code.

– A diagnostic data area returned by the BCPii callable service, which contains information related to the BCPii service that failed.

New message IXC112I

The new IXC112I message is shown in Figure 31-12.

```
IXC112I BCPII SYSTEM ERROR. BCPII CALLABLE SERVICE bcpiiservice
  REQUEST: request
  RETURN CODE: retcode
  DIAG CODE: diagcode

Explanation: A BCPii callable service was issued on the local system by XCF in
support of the system status detection partitioning protocol, but the BCPii
callable service returned a failing return code.
```

Figure 31-12 IXC112I message

In the message text:

- `bcpiiservice` - the name of the BCPii callable service that failed. bcpiiservice is one of the following services:
  - HWICONN, the BCPii HWICONN callable service
  - HWILIST, the BCPii HWILIST callable service
  - HWIQUERY, the BCPii HWIQUERY callable service
  - HWIEVENT, the BCPii HWIEVENT callable service
  - HWICMD, the BCPii HWICMD callable service
  - HWIDISC, the BCPii HWIDISC callable service request which is the internal process that was in control at the time of the service failure.
- `request` - one of the following processes:
  - LOCAL INITIALIZATION - Local system initialization to connect to the local CPC and CPC image through BCPii.
  - SYSTEM RESET - System reset processing against a remote target system.
  - DISCONNECT - Local system disconnect processing from the local CPC and CPC image and remote CPC and CPC images through BCPii.retcode The return code from the BCPii callable service that failed.
- `diagcode` - diagnostic data that was returned by the bcpiiservice to help determine the cause of the service failure.

New message IXC113I

The new IXC113I message is shown in Figure 31-13 on page 630.

```
IXC113I BCPII CONNECTION TO SYSTEM sysname
  RELEASED text
  In the message, text is:
    DISCONNECT REASON: discreason
    IMAGE NAME: image
    NETWORK ADDRESS: netaddr
    SYSTEM NUMBER: sysnum
    IPL TOKEN: ipltoken

Explanation: The local system has released its connection to remote system
image because either the system image image has been removed from the sysplex
or the SSD protocol has been disabled on the local system.  The local system
can no longer target remote system image image with system status detection
partitioning protocol commands through BCPii
```

Figure 31-13 IXC113I message

The local system has released its connection to remote system image using the BCPii HWIDISC callable service for reason `discreason`. The local system can no longer target the remote system image with system status detection partitioning protocol commands through BCPii.

In the message text:

- `sysname` - the name of the remote system in the sysplex that the local system has released its virtual connection to.
- `discreason` - the reason why the local system released its BCPii connection to remote system `sysname`.
- One of the following lines is displayed:
  - SYSTEM IMAGE REMOVED FROM SYSPLEX - the system image has been removed from the sysplex because of a partition action taken against the system sysname.
  - SYSSTATDETECT DISABLED ON LOCAL SYSTEM - the system status detection partition protocol has been disabled on the local system by a SETXCF FUNCTIONS command. When the function is reenabled on the local system, it will attempt to connect to other remote systems in the sysplex that are eligible targets of the system status detection partitioning protocol.

New message IXC114I

The new IXC114I message is shown in Figure on page 631.

```
IXC114I LOGICAL PARTITION REMOTE STATUS INFORMATION
  SYSTEM NAME:  sysname
  SYSTEM NUMBER: sysnum
  CPC:  cpc
  STATUS: OPERATING  |  NOT OPERATING |  NO POWER |  OTHER |  N/A
  OPERSTAT: cpcstat
  IMAGE NAME:  image  STATUS: OPERATING  |  NOT OPERATING |  NO POWER |  OTHER
    |  N/A   OPERSTAT:  imagestat
  STORED IPL TOKEN:  ipltoken
  RETURNED IPL TOKEN: ipltoken
  DIAG INFO:

Explanation: When the System Status Detection Partition Protocol is enabled XCF
will obtain status information for CPC and Image Name via BCPii APIs in
response to a DOWN reply to IXC102A or IXC402D to determine if a SYSTEM RESET,
LOAD or other acceptable alternative action was performed that results in the
reset of system name
```

Figure 31-14 IXC114I message

The system continues processing. If it is determined that system name has not been RESET, then message IXC208I is issued stating that a DOWN reply was entered without system name being RESET.

If the returned status for Image is OPERATING, take appropriate action to system reset system name and reply DOWN again to message IXC102A or IXC402D. See message IXC102A or IXC402D for acceptable actions that will reset a system.

Updated message IXC373I

The updated IXC373I message is shown in Figure 31-15.

```
IXC373I XCF/XES OPTIONAL FUNCTIONS ENABLED | DISABLED
  SYSSTATDETECT

Explanation: A COUPLExx parmlib member FUNCTIONS statement or a SETXCF
FUNCTIONS command specified that one or more installation-controllable optional
functions are to be enabled or disabled. This message reports the results of
that request.
```

Figure 31-15 IXC373I message

31.1.6 Sysplex partitioning and system isolation considerations

The sysplex failure management (SFM) policy allows you to specify how a system is to respond to a Status Update Missing (SUM) condition. The system default is used when no action is specified for the system or when no SFM policy is active.

When the SSD protocol is enabled, the steps taken during sysplex partitioning and system isolation are affected and automatic action can be taken regardless of whether an SFM policy is active. The following sections discuss the actions taken during sysplex partition and system isolation, as well as the effect of having the SSD protocol enabled.

Sysplex partitioning triggers system isolation

When a system is partitioned, for example as a result of an **V XCF,systemname,OFFLINE** command, the following processing is performed:

1. Sysplex partitioning is initiated - IXC101I.
2. The CLEANUP interval elapses.
3. Isolation is attempted - messages IXC108I and IXC109I.
4. If isolation is incomplete, then a response to IXC102A is needed.
5. Sysplex partitioning is complete - IXC105I.

The system isolation will generate messages IXC108I and IXC109I. If successful, then sysplex partitioning will complete without issuing IXC102I. If isolation fails, then IXC102I will be issued and a response is required.

SSUM triggers system isolation

1. Status updates stop.
2. FDI elapses and the system is declared SSUM.
3. ISI elapses.
4. A PR/SM SSUM policy action attempted if applicable.
5. Sysplex partitioning is initiated - IXC101I.
6. Isolation is attempted (if needed) - IXC108I and IXC109I.
7. If isolation is incomplete, then a response to IXC102A is needed.
8. Sysplex partitioning is complete - IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The partitioning reason in this case is SFM STARTED DUE TO STATUS UPDATE MISSING or POLICY INITIATED REQUEST if PR/SM SSUM action was successful.

Note: Sysplex partitioning does not wait the CLEANUP interval because it does not expect that the system being partitioned can perform cleanup when it is SSUM. The only time isolation would not be needed for SSUM-triggered partitioning is when a PR/SM SSUM policy action was successful. The operator needs to ensure the system has been reset before responding to IXC102A.

SSUM triggers PROMPT

1. Status updates stop.
2. FDI elapses and the system is declared SSUM
3. OPNOTIFY elapses and a response to IXC402D is needed.
4. Sysplex partitioning is initiated - IXC101I.
5. Sysplex partitioning is complete - IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The OPNOTIFY “timer” starts when the FDI “timer” starts, so IXC402D is issued after any OPNOTIFY time in excess of the FDI. The operator needs to ensure that the system has been reset before responding to IXC402D. The partitioning reason in this case is SYSTEM STATUS UPDATE MISSING.

Cleanup interval

When MVS is removing a system from a sysplex, the cleanup interval is the maximum number of seconds that MVS waits for members of a group to clean up their processing before putting the system on which they reside in a wait state. (Note that this is not a required protocol). Some groups, such as the global resource serialization group and the console services groups, delay cleanup until the expiration of the cleanup interval. After all members finish their cleanup, or when the cleanup interval elapses, MVS puts the system into a non-restartable wait state.

Note: Sysplex partitioning does not wait the CLEANUP interval, and neither ISI nor isolation is performed.This is because the operator indicated the system is already RESET.

SSD protocol triggers system isolation

1. Status updates stop.
2. About 6 seconds elapse.
3. SSD protocol detects the system is demised and issues IXC106I.
4. A PR/SM SSUM policy action is attempted if applicable.
5. Sysplex partitioning is initiated - IXC101I.
6. Isolation is attempted (if needed) - IXC108I and IXC109I.
7. If isolation is incomplete, then a response to IXC102A response is needed.
8. Sysplex partitioning is complete - IXC105I.

A non-restartable disabled wait state is an example of when this occurs without the SSD protocol enabled. The SSD protocol checks the system's heartbeat every 3 seconds until status updates resume or the system is reset or a third system is monitoring partitioning. With SSD enabled:

- Sysplex partitioning does not wait the FDI because a "failure" has definitely been detected.
- Sysplex partitioning does not wait the ISI because status has been determined.
- Sysplex partitioning does not wait the CLEANUP interval because a system that is demised cannot perform cleanup.

The operator needs to ensure that the system has been reset before responding to IXC102A. System isolation is only needed when the system is demised due to a wait state and the PR/SM SSUM action did not request RESET/DEACTIVATE.

ISOLATETIME(0) default

There are two types of default for ISOLATETIME:

- The default for ISOLATETIME taken against the local system by other systems
- The default for ISOLATETIME taken by the local system against other systems

Without the SSD protocol enabled, both defaults for ISOLATETIME were PROMPT. With the SSD protocol enabled, both are calculated. The default is always PROMPT for systems without the SSD protocol enabled.

If a PR/SM SSUM policy action is being used and fails, XCF will then try the default action. Without this support, OPNOTIFY processing still goes on in the background when a PR/SM SSUM policy action is taking place. With SSD protocol enabled, if the fallback default action is the new default of ISOLATETIME(0), then IXC402D will not be issued and partitioning will be

initiated if the PR/SM SSUM policy action fails. The result is equivalent to ISOLATETIME(0) being used instead of the RESETTIME/DEACTTIME.

Note: When an SFM policy is not defined and SSD is enabled, the default action is equivalent to ISOLATE(0) rather than PROMPT.

Specifying SSUMLIMIT for a system without specifying ISOLATETIME, RESETTIME, or DEACTTIME previously resulted an error. It is now allowed and results in a default of ISOLATETIME(0).

- If a downlevel system uses a policy defined by an IXCMIAPU with the SSD protocol enabled, PROMPT will be taken as the default instead of ISOLATETIME(0). Specifying SSUMLIMIT without specifying ISOLATETIME, RESETTIME, or DEACTTIME will be ignored.
- If a system with the SSD protocol enabled uses a policy defined by a downlevel IXCMIAPU, then ISOLATETIME(0) SSUMLIMIT(NONE) may be taken as the default instead of PROMPT.

Note: Each system decides what the default should be. Therefore, an R11 system with the SSD protocol enabled will determine a default for ISOLATETIME that is different from an R10 system.

Displaying the SSUM values

The **D XCF,C** command output indicates the effective values for SSUM ACTION, SSUM INTERVAL, and SSUM LIMIT when an SFM policy is active on all systems.

- Without the SSD protocol enabled, the **D XCF,C** command output indicates N/A for these values when an SFM policy is not active on all systems.
- With the SSD protocol enabled, the **D XCF,C** command output more often indicates the effective values expected to be used.

Only the following situations may result in an N/A display:

- There is no other active system in the sysplex.
- There is a mixture of z/OS V1R11 and lower-level systems active in the sysplex.
- A PR/SM policy is active on some system.

In effect, the SSUM values are displayed when the expected effective values apply and can be reasonably determined. N/A is displayed for mixed systems because an R11 may isolate, while an R10 may PROMPT.

System isolation for sysplex partitioning

System isolation is performed when XCF has not already been ensured that the system being partitioned has stopped I/O to all shared devices. Without the SSD protocol enabled, XCF only attempts to isolate a system (using fencing services) when an SFM policy is active. With the SSD protocol enabled, regardless of the SFM policy, XCF attempts to fence whenever the system being partitioned stored its fence authority in a CF. System isolation is no longer tied to an SFM policy, so a system can be isolated regardless of whether there is an SFM policy.

XCF attempts to SYSTEM RESET using SSD Protocol (BCPii) whenever fencing did not isolate and the system being partitioned self-identified its image/CPC/ipltoken and the SSD Protocol is enabled for systems.

System isolation not performed (or required) when a system is demised due to one of the following:

- SYSTEM RESET OR NEW IMAGE LOADED
- PARTITION DEACTIVATED
- CPC FAILURE

System fencing messages

New messages IXC108I and ICXC109I are written to SYSLOG by the partitioning monitor system. After IXC108I is issued for fencing, all systems in the sysplex attempt to fence the system. If a system is successful, the fence will be successful. Otherwise, the fence times out after 32 seconds. Each systems cuts a symptom record to log the results of the fence attempt. If fencing times out, RESET may then be attempted or IXC102A will be issued.

System reset messages

New messages IXC108I and IXC109I are written to syslog by the partitioning monitor system. After IXC108I is issued for system reset, only the system that issued the IXC108I (the partitioning monitor) attempts to reset the system. If results are provided, IXC109I is issued with the results. Otherwise, the reset times out after 30 seconds. A successful RESET is indicated by HWMCA_EVENT_COMMAND_RESPONSE RETURN CODE 0.

For return codes from a BCPii callable service, see *z/OS MVS Programming: Callable Services for High Level Languages*, SA22-7613, for more information. For return codes from an ENF 68 command response, see *System z Application Programming Interfaces*, SB10-7030, for more information.

When the RESET is not successful, IXC102A will be issued. Only one system attempts the RESET, although every system attempts the isolate.

Important: The HWICMD return code will be'00000F02'X (3842) if XCFAS does not have the required SAF authorization.

Verification when operator indicates system reset

Operators are responsible for verifying that a system can no longer perform I/O to shared devices before replying `DOWN` to IXC402D or IXC102A:

```
IXC402D sysname LAST OPERATIVE AT hh:mm:ss. REPLY DOWN AFTER SYSTEM RESET, OR
INTERVAL=sec TO SET A REPROMPT TIME.
IXC102A XCF IS WAITING FOR SYSTEM sysname DEACTIVATION. REPLY DOWN WHEN MVS ON
sysname HAS BEEN SYSTEM RESET.
```

Or replying `SYSNAME=sysname1,DOWN` to IXC409D:

```
IXC409D SIGNAL PATHS BETWEEN sysname1 AND sysname2 ARE LOST. REPLY RETRY OR
SYSNAME=SYSNAME OF THE SYSTEM TO BE REMOVED.
```

A system must also be reset *prior* to issuing the **`VARY XCF,sysname,OFFLINE,FORCE`** command. (Operators often have not verified that the system was reset before performing these actions, whose prerequisite is a system reset.) If the verification is inconclusive, then XCF has no choice but to trust the operator as it has always done.

With the SSD protocol enabled, XCF can verify that the system has been reset and so avoid operator errors that may cause data integrity problems. A response to IXC102A, IXC402D, or IXC409D should not be required if SSD Protocol is working. This means that receiving an

XC102A, IXC402D, or IXC409D with the SSD protocol enabled indicates that there was a problem resetting the system.

Replying DOWN to IXC102A, IXC402D, or IXC409D with the SSD protocol enabled will most likely result in one of the forms of message ICXC208I shown in Figure 31-16 on page 636.

```
IXC208I THE RESPONSE TO MESSAGE msgnum IS INCORRECT: DOWN REPLY ENTERED WITHOUT
SYSTEM RESET

IXC208I THE RESPONSE TO MESSAGE IXC409D IS INCORRECT: SYSNAME=sysname1,DOWN
REPLY ENTERED WITHOUT SYSTEM RESET
```

Figure 31-16 IXC208I issued with the SSD protocol enabled

The SSD protocol can result in the **VARY XCF,sysname,OFFLINE,FORCE** command being rejected because the system was not reset, thereby resulting in message IXC370I being issued as shown in Figure 31-17.

```
IXC370I THE VARY XCF COMMAND COULD NOT BE PROCESSED: FORCE OPTION USED WITHOUT
SYSTEM RESET
```

Figure 31-17 IXC370I issued with the SSD protocol enabled

Tip: When IXC 208I or IXC370I are issued, you may need to bypass SSD to allow a reply of DOWN to be accepted by IXC102A, IXC402D, or IXC409D, or to allow the FORCE action to be taken. You can bypass SSD by using the following command:

```
SETXCF FUNCTIONS,DISABLE=SYSSTATDETECT
```

31.2 XCF FDI consistency enhancement

Prior to z/OS V1R11, if no SFM policy was active, the system specifies that the SSUM ACTION is to take "the default action." However, to a pre-R11 system, the default action is PROMPT. Prior to V1R11, if an SFM policy with ISOLATETIME was active, the system states that the SSUM ACTION was ISOLATE.

Important: z/OS V1R11 understands that "the default action" for a pre-R11 system is PROMPT, and that it is ISOLATE for an R11 or higher system.

However, pre-z/OS V1R11 does *not* understand that "the default action" for an R11 system is ISOLATE.

New in z/OS V1R11 is System Status Detection Partitioning Protocol (SSD). This enhancement to failed-system handling is designed to attempt to partition a failed system from the sysplex in a more timely way and with improved data integrity. SSD achieves this by exploiting the new BCPii support to communicate with the Service Element about the status of an LPAR.

The System Status Detection (SSD) Partitioning Protocol exploits BCPii interfaces to use z/Series hardware services to discover the status of failed systems in a sysplex during sysplex recovery processing. Using BCPii, XCF can bypass specified intervals such as the

failure detection interval (FDI), the SSUM ACTION interval, and the cleanup interval. It can also avoid fencing processing and manual intervention requirements, to shorten sysplex recovery processing time.

31.2.1 Failure detection interval

The failure detection interval (FDI) is the amount of time that a system can appear to be unresponsive before XCF takes action to remove the system from the sysplex. This is referred to as the effective FDI or INTERVAL.

The FDI is based on:

- The value of the FDI specified for the system
- The value of the OPNOTIFY specified for the system
- The value based on the system's excessive spin parameters

This allows the system's processing of excessive disabled spin conditions, the sysplex's handling of missing system heartbeats, and the initiation of sysplex partitioning to remove unresponsive systems, to be more consistent.

31.2.2 FDI and spin overview

As mentioned, the FDI is used to indicate how long a system may appear unresponsive before other systems in the sysplex begin taking disruptive actions against that system. The unresponsive system will be removed from the sysplex forcibly by using the sysplex partitioning process.

You can specify the FDI by using the INTERVAL keyword in the COUPLExx parmlib member. You can modify the FDI by using the **SETXCF COUPLE,INTERVAL=xx** command or a SETXCF command, or you can take the default.

The default (and recommended) value for INTERVAL or for the spin failure detection interval is computed as follows, where N is the number of SPINRCVY actions, and +1 indicates the implicit S:

```
(N+1) * SPINTIME + 5 (seconds)
```

A status update missing (SUM) condition occurs when a system in the sysplex does not update its status information within the FDI. There is a relationship between the INTERVAL specified for FDI and the SPIMTIME specified using the EXSPATxx parmlib member for excessive spin.

EXSPATxx parmlib member

The excessive spin parameters indicate how long a system is allowed to remain in a disabled spin loop before the system will take action to break out of the spin. Because a system generally appears unresponsive to XCF while it is in a disabled spin, the general rule is that the FDI interval must be at least as long as the excessive spin time based on the actions specified using the EXSPATxx parmlib member.

The default values for the EXSPATxx parmlib member for a system running in an LPAR with shared CPs are shown in Figure 31-18.

```
SPINRCVY ABEND,TERM,ACR
SPINTIME=40
```

Figure 31-18 IBM defaults for EXSPATxx

The resulting excessive spin actions are SPIN, ABEND,TERM,ACR because the system will always take an automatic spin action first. This would result in a valid spin time of 160 seconds (4 spin actions multiplied by the spin time interval of 40).

Prior to z/OS V1R11 the default FDI was 85 seconds, which is inconsistent with the default excessive spin time of 160 seconds.

31.2.3 Definition of new FDI- and spin-related terms

New FDI- and spin-related terminology is explained here, to help you understand the new function introduced with z/OS V1R11. As mentioned, the failure detection interval (FDI) is the amount of time that a system can appear to be unresponsive (status update missing) before XCF is to take action to resolve the problem. With z/OS V1R11, the following terms are used:

User FDI This an FDI value explicitly specified by the user, either directly by using INTERVAL in the COUPLExx parmlib member, or by using the **SETXCF COUPLE** command.

Spin FDI This is an FDI value derived by XCF from the excessive spin parameters. The default is (N+1) * SpinTime where N = the number of excessive spin recovery actions.

Effective FDI This is the larger of the user FDI and the spin FDI.

User OpNotify This is an OpNotify value explicitly specified by the user, either directly by using the COUPLExx parmlib member, or by using the **SETXCF COUPLE** command.

Relative OpNotify This is an OpNotify value that is determined relative to the effective FDI. Prior to z/OS V1R11 the OpNotify value was always an absolute value and was required to be greater than or equal to the (effective) FDI.

With a relative OpNotify value, the system automatically maintains the relationship between FDI and OpNotify. If the effective FDI changes, the effective OpNotify value changes as well.

Effective OpNotify This is the OpNotify value being used by the system. The system ensures that effective FDI is always less than or equal to the effective OpNotify value. Changes to the effective FDI could cause the effective OpNotify to change as well.

Resolved FDI The effective FDI is a value recalculated after changes to the spin loop recovery actions.

Important: In z/OS V1R11, taking the IBM defaults will result in an FDI of 165 seconds rather than 85 seconds for prior releases.

31.2.4 Relative OpNotify

XCF provides new support to allow the OPNOTIFY value to be explicitly specified as a relative value by:

- Specifying OPNOTIFY(+30) in the COUPLExx parmlib member to specify a delta value of 30 seconds
- Using SETXCF COUPLE,OPNOTIFY=+30

The plus (+) sign indicates that an OPNOTIFY value is being specified relative to the effective FDI. Without the plus sign, the specification will continue to be treated as an absolute value.

XCF requires that the (User) OPNOTIFY value be greater than or equal to the (User) FDI. This is because the system must be status update missing before the operator can be notified. Any attempt to change the FDI to a value larger than the absolute OPNOTIFY would be rejected. You must first change the OPNOTIFY to a larger value to ensure that the new FDI and the OPNOTIFY value will maintain the required relationship with the proposed new FDI setting.

31.2.5 New rules for setting FDI and OpNotify

With z/OS V1R11 there are new rules and new behavior related to FDI and spin specifications:

- The user FDI must be less than or equal to the user OPNOTIFY value.
- If the spin FDI is less than or equal to the user FDI, the effective FDI equals the user FDI and there is no change in behavior.
- If the spin FDI is greater than the user FDI, the effective FDI is the spin FDI.
- If the user OPNOTIFY value has the proper relationship to the user FDI (less than or equal) but the user OPNOTIFY is less than the effective FDI, then the effective OPNOTIFY is equal to the effective FDI.
- If OPNOTIFY is specified in the COUPLExx parmlib but FDI (via INTERVAL) is not, the default FDI is the spin FDI.

Note: XCF will not reject the parmlib member if the user absolute OPNOTIFY value is less than the default FDI. Message IXC206I will still be issued to note the exceptional condition.

A new SETXCF FUNCTIONS option USERINTERVAL is defined to allow the client to indicate that the explicitly specified user INTERVAL is to be used as the effective FDI even if that value is smaller than the spin FDI.

There is no option directly related to the OPNOTIFY specification. If the user FDI is being used unconditionally as the effective FDI, then the absolute OPNOTIFY specification inherently reverts to the old behavior.

If a relative OPNOTIFY value is specified, the old behavior can be restored by reverting to an absolute OPNOTIFY specification.

Note: USERINTERVAL is DISABLED by default, so the new behaviors apply.

31.2.6 Operations considerations

The XCF FDI consistency enhancement introduces new operator commands and parmlib parameters. The message displays the identification information about system status detection partitioning protocol associated with this system. This message is issued when the system performs initialization processing to become enabled to target other systems and be targeted by other systems using the system status detection partitioning protocol.

As part of the initialization process, the system establishes a logical application connection to BCPii to issue remote hardware management console API commands against other systems that are eligible targets of the system status detection partitioning protocol.

SETXCF COUPLE,OPNOTIFY=+nn

The **SETXCF COUPLE,OPNOTIFY=nn** command is updated to allow *nn* to be specified in the form *+xxx* where *xxx* is a number in the range 0 to 86400. Prior to this support, *nn* had to be a number in the range 3 to 86400. This new specification of *+xxx* indicates that the OPNOTIFY interval used by the system is to always be *xxx* seconds more than the effective FDI being used by the system.

If the command is accepted, the new message IXC470I is issued to document the resulting OPNOTIFY interval being used by the system even if none of the user values or the effective values actually change.

SETXCF FUNCTIONS option USERINTERVAL

There is a new SETXCF FUNCTIONS option USERINTERVAL. If USERINTERVAL is enabled, then the system will allow an explicitly specified user INTERVAL value to be used as the effective FDI even if that value is smaller than the spin FDI.

USERINTERVAL can be specified by using an operator command or by using the COUPLExx parmlib member:

- Operator command

  ```
  SETXCF FUNCTIONS, ENABLE=USERINTERVAL
  ```

- COUPLExx parmlib member

  ```
  FUNCTIONS ENABLE(USERINTERVAL)
  ```

Callable services

Callable service IXCQUERY, REQINFO = SYSPLEX returns information about each system in the sysplex. The values returned for this interface will be the effective FDI and effective OPNOTIFY values.

No external programming interfaces will be provided to return the user FDI, spin FDI or user OPNOTIFY values. Only the effective FDI and effective OPNOTIFY will be available.

New and updated messages

Message IXC206I shown in Figure 31-19 on page 640 has a new insert to indicate that the user specified (absolute) OPNOTIFY value is not valid for the default Spin FDI. IXC206I is issued if:

- User FDI > absolute User OpNotify (old)
- Spin FDI > absolute User OpNotify (new)

```
IXC206I THE COUPLE OPNOTIFY KEYWORD IS ERRONEOUS:
OPNOTIFY VALUE IS LESS THAN DERIVED SPIN INTERVAL
```

Figure 31-19 IXC206I issued when OPNOTIFY is invalid

When IXC206I is issued, the IPL continues but the effective OPNOTIFY will be the Spin FDI as displayed by message IXC470I shown in Figure 31-23 on page 643. The specified OPNOTIFY should be increased to be at least as large as the derived Spin FDI, or a relative OPNOTIFY should be specified.

Message IXC357I, shown in Figure 31-20, is the output to DISPLAY XCF,COUPLE. This displays the effective FDI interval.

```
IXC357I DISPLAY XCF, COUPLE

SYSTEM sysname DATA
INTERVAL OPNOTIFY MAXMSG CLEANUP RETRY CLASSLEN
interval opnotify maxmsg cleanup retry classlen
XX=fffff    nnnnn
SSUM ACTION  SSUM INTERVAL  SSUM LIMIT  WEIGHT  MEMSTALLTIME
action       sfminterval    ssumlimit   weight  memstalltime

DEFAULT| PARMLIB | SETXCF USER INTERVAL:  165
DERIVED SPIN INTERVAL:  165
DEFAULT | PARMLIB | SETXCF USER OPNOTIFY: +5
```

Figure 31-20 IXC357I displays the output from D XCF,COUPLE

IXC357I message

In the IXC357I message:

- `INTERVAL` is the effective FDI interval.
- `OPNOTIFY` is the effective OPNOTIFY.
- `DEFAULT USER INTERVAL` is derived from the excessive spin parameters that are currently defined for the system. The value may change dynamically in response to the SET EXS command.
- `PARMLIB USER INTERVAL` was explicitly set using parameters specified in the COUPLExx parmlib member at IPL time.
- `SETXCF USER INTERVAL` was set via the **`SETXCF COUPLE,INTERVAL`** command.
- `DERIVED SPIN INTERVAL` is the FDI value derived from the excessive spin recover parameters.
- `DEFAULT USER OPNOTIFY` is set to the default value. The default OPNOTIFY interval is a relative value of three seconds. Thus the effective OPNOTIFY value will be three more than the effective INTERVAL value (but not more than the maximum value of 86400).
- `PARMLIB USER OPNOTIFY` was explicitly set using parameters specified in the COUPLExx parmlib member at IPL time.
- `SETXCF USER OPNOTIFY` was set using the **`SETXCF COUPLE,OPNOTIFY`** command.

Figure 31-21 on page 642 is an example of the **`D XCF,COUPLE`** output from a running system.

```
IXC357I  15.12.46  DISPLAY XCF       FRAME  1     F      E   SYS=D13ID71SYSTEM
D13ID71 DATA
   INTERVAL   OPNOTIFY     MAXMSG    CLEANUP      RETRY   CLASSLEN
        325        325       3000         60         10        956

   SSUM ACTION  SSUM INTERVAL  SSUM LIMIT     WEIGHT  MEMSTALLTIME
        PROMPT            325         N/A        N/A           N/A

   PARMLIB USER INTERVAL:     60
   DERIVED SPIN INTERVAL:    325
   PARMLIB USER OPNOTIFY:     60 < - - - snip - - - >
OPTIONAL FUNCTION STATUS:
  FUNCTION NAME                    STATUS      DEFAULT
  DUPLEXCF16                       ENABLED     DISABLED
  SYSSTATDETECT                    ENABLED     ENABLED
  USERINTERVAL                     DISABLED    DISABLED
```

Figure 31-21 IXC357I display for a running system

Message IXC373I, shown in Figure 31-22, displays the status of the USERINTERVAL.

```
IXC373I XCF/XES OPTIONAL FUNCTIONS ENABLED | DISABLED USERINTERVAL

Explanation: A COUPLExx parmlib member FUNCTIONS statement or a SETXCF
FUNCTIONS command specified that one or more installation-controllable optional
functions are to be enabled or disabled. This message reports the results of
that request.
```

Figure 31-22 IXC373I displays the status of USERINTERVAL

IXC470I message

The new message IXC470I, shown in Figure 31-23, is issued to document the effective failure detection interval (INTERVAL) and the effective operator notification interval (OPNOTIFY), as well as the data used to compute these values. The message is issued by XCF in response to changes that can impact the failure detection interval or the operator notification interval.

```
IXC470I SYSTEM SC70 EFFECTIVE VALUES: INTERVAL=165 OPNOTIFY=168
DEFAULT USER INTERVAL:    165
DERIVED SPIN INTERVAL:    165
DEFAULT USER OPNOTIFY: +    3
COMPUTED FOR:  XCF INITIALIZATION
```

Figure 31-23 IXC470I displays the effective FDI

Note: The system uses the indicated effective intervals as described here:

- If the system fails to update its status within the effective failure detection interval, then it is considered to be status update missing.
- If the system fails to update its status within the effective operator notification interval, the operator can be notified of the condition by an XCF message (for example, IXC402D).

A system that is status update missing can cause sympathy sickness on other systems in the sysplex; remove it from the sysplex to avoid further problems.

In the IXC470I message, the possible fields are listed here:

- `INTERVAL` is the effective FDI interval.
- `OPNOTIFY` is the effective OPNOTIFY.
- `DEFAULT USER INTERVAL` is derived from the excessive spin parameters that are currently defined for the system. The value may change dynamically in response to the **`SET EXS`** command.
- `PARMLIB USER INTERVAL` was explicitly set using parameters specified in the COUPLExx parmlib member at IPL time.
- `SETXCF USER INTERVAL` was set using the **`SETXCF COUPLE,INTERVAL`** command.
- `DERIVED SPIN INTERVAL` is the FDI derived from the current excessive spin recovery parameters expressed in seconds.

  The value is computed as follows: SpinFDI = (N+1)*SpinTime + 5 where *N* is the number of excessive spin recovery actions and *SpinTime* is the excessive spin loop time-out interval.
- `DEFAULT USER OPNOTIFY` is set to the default value. The default OPNOTIFY interval is a relative value of three seconds. Thus the effective OPNOTIFY value will be three more than the effective INTERVAL value (but not more than the maximum value of 86400).
- `PARMLIB USER OPNOTIFY` was explicitly set using parameters specified in the COUPLExx parmlib member at IPL time.
- `SETXCF USER OPNOTIFY` was set using the **`SETXCF COUPLE,OPNOTIFY`** command.
- `COMPUTED FOR` is the processing for which this message was issued.

31.2.7 Migration and coexistence considerations

There are no migration or coexistence actions required. It is important to understand the new behavior with respect to the current COUPLExx INTERVAL and OPNOTIFY settings.

Fallback to the previous z/OS level will be possible after this function is enabled. However, if COUPLExx OPNOTIFY is modified to be a relative value, then systems without this support would not be able to process the parmlib member.

If a user specified an FDI that is less than the spin FDI that is required to be used as the effective FDI, then the XCF option USERINTERVAL should be ENABLED.

If an OPNOTIFY value is set in the COUPLExx parmlib member that is less than (N+1)*spintime+5 and the system takes the default FDI value, then the parmlib could be acceptable to the new release but not the old. The COUPLExx parmlib member would have to be modified appropriately before using it for a downlevel release. This means that changes to the COUPLExx should not be made to shared or cloned parmlib members until fallback or shared use is no longer needed.

System programmers are typically responsible for specifying the FDI and OPNOTIFY intervals, as well as the excessive spin parameters. No changes are required, but reevaluate the settings in light of the new behavior. The information in new message IXC470I or the updated output of the **DISPLAY XCF,COUPLE** command can be used to determine the effective FDI being used. The messages show the user FDI, spin FDI, user OPNOTIFY, and effective OPNOTIFY that are in use by the system.

Prior to z/OS R1V11, the OPNOTIFY value could only be specified as an absolute value. However, because most clients choose the setting to achieve the desired time lag (delta) from the FDI, then specifying a relative OPNOTIFY value is worth considering.

The default (spin) FDI is suitable for most production environments. Clients who explicitly code an FDI should consider taking the default.

31.2.8 Function summary

The failure detection interval (FDI) in use by systems in a sysplex is automatically adjusted, taking into account the value of the FDI specified for the system and the value based on the system's excessive spin parameters. This value is know as the effective FDI.

XCF will ensure that the OPNOTIFY interval is also adjusted to be the larger of the user OPNOTIFY or the effective FDI.

31.3 Pending delete CF structure cleanup

Coupling Facility (CF) structures with a status of PENDING DEALLOCATION can lead to confusion when displayed on **DISPLAY XCF** command output or reported by an IXCQUERY request. There have been many client requests for a method to use to remove these structures from the display.

In z/OS V1R11, new function has been added to support a new option on the **SETXCF FORCE** command to request cleanup of pending deallocation structures. This allows confusing structure instances on XCF CF structure displays to be removed.

Problem overview

When a CF containing structures loses connectivity from all systems in the sysplex, XCF will remember information about those structures that were in the lost CF. XCF will checkpoint information about those structure instances in the CFRM Active Policy.

When the active instances of these CF structures move elsewhere, XCF continues to remember the structure instances that existed in the lost CF. The checkpoint information is the way XES will remember to deallocate these CF structures if connectivity to the lost CF is regained.

In most scenarios when a system regains connectivity to the lost CF, CFRM will clean up these “pending deallocation” structure instances as soon as some system in the sysplex reestablishes connectivity to the CF. XCF will deallocate the structure instances in the CF (if they still exist) and delete any checkpoint entries for those structures from the CFRM active policy. The problem generally fixes itself eventually, without needing any manual action.

However, cases in which the CF is going to be permanently removed are examples of instances where “pending deallocate” CF structure entries will still reside in the CFRM Active Policy. These CF structures show up as “pending deallocation” structures in IXCQUERY and **DISPLAY XCF** command output. The appearance of these “phantom” structure instances in inaccessible CFs can be confusing.

These instances of the CF Structures in the lost CF will be eventually deallocated or never come back. They may already be gone if the lost CF has been deactivated or activated. In some cases, such as a CEC upgrade, the CF and its structures will be permanently gone.

New function overview

New function has been added to z/OS V1R11 to allow clients to indicate to XCF that the checkpoint information, and therefore the displays for these structures, can be deleted.

The checkpoint information may be deleted if the CFNAME still exists in the CFRM Policy and the CF has lost connectivity and become inaccessible to all the systems in the sysplex.

The checkpoint information may not be deleted if the CFNAME has been deleted from the CFRM Policy or the CFNAME has been redefined to another physical CF.

SETXCF FORCE, PNDSTR command

A new keyword can be specified using the **SETXCF FORCE** command to remove pending-deallocation structure entries from the CFRM Active policy:

```
SETXCF FORCE,PNDSTR,CFNAME=cfname
```

- The CFNAME specified should be a CF that is not connected to any system in the sysplex.
- The CF will remain inaccessible for an extended period of time or will be brought back online with all structures removed.

Note: CFRM will perform the same cleanup automatically when the connectivity to the CF (with all structures removed) is restored.

New messages

Two new messages are provided with this support, IXC554I and IXC555I.

IXC554I and IXC555I will be issued in response to the **SETXCF FORCE PNDSTR** command to remove pending deallocation structure information from the CFRM active policy.

- IXC555I is a hardcopy-only message and will be issued for each structure record deleted.
- IXC554I is a console message that will be issued at the completion of the force processing.

Figure 31-24 shows the IXC554I completed message. If the **SETXCF FORCE PNDSTR** command completed, then the **D XCF,CF,CFNM** command can be used to verify that the structures pending deletion were removed from the active policy. See the system log for an instance of message IXC554I for each structure pending deletion that was removed from the CFRM active policy.

```
IXC554I THE SETXCF FORCE PNDSTR REQUEST FOR COUPLING FACILITY cfname WAS
COMPLETED:
PENDING-DEALLOCATION INFORMATION REMOVED FOR 10 STRUCTURE(S)
```

Figure 31-24 IXC554I completed message

Figure 31-25 shows the IXC554I rejected message. The most likely reason for the rejection is because the CF has not been removed from the CFRM policy.

```
IXC554I THE SETXCF FORCE PNDSTR REQUEST FOR COUPLING  FACILITY cfname WAS
REJECTED:

COUPLING FACILITY IS ABLE TO BE USED BY THE SYSPLEX
COUPLING FACILITY NOT DEFINED IN THE CFRM ACTIVE POLICY
UNEXPECTED ERROR
XES FUNCTION NOT AVAILABLE
COUPLE DATA SET FOR CFRM NOT AVAILABLE
```

Figure 31-25 IXC554I rejected message

Message IXC555I

Message IXC555I shown in Figure 31-26 is issued when the **SETXCF FORCE,PNDSTR command** is used to remove pending-deallocation structure information from a CF. The pending-deallocation structure information for the structure strname has been processed and will be successfully removed from the CFRM active policy upon completion of the **FORCE,PNDSTR** command as indicated by message IXC554I. The system displays the names of the structures pending deletion that are deleted.

```
IXC555I DELETED PENDING-DEALLOCATION STRUCTURE INFORMATION FOR STRUCTURE
strname,  PHYSICAL VERSION: C3C6B4A2 6F5D45A3
        COUPLING FACILITY SIMDEV.IBM.EN.ND0100000000
                          PARTITION: 00    CPCID: 00
```

Figure 31-26 IXC555I deleted structure message

Migration and coexistence considerations

The **SETXCF FORCE** command option for PNDSTR will only execute successfully on V1R11 systems and above. Lower-level releases will correctly handle the deleted checkpoint entries and display the correct information if a V1R11 system has deleted the structure checkpoint entries.

This support will not be rolled down to lower-level releases.

31.4 AutoIPL update

AutoIPL is an availability function, introduced in z/OS V1R10, designed to allow the system to automatically IPL stand-alone dump, z/OS, or both, when the system requests certain disabled wait states to be loaded, or when specified as part of a request to VARY the system out of a sysplex.

AutoIPL was disabled for Parallel Sysplex environments with a Sysplex Failure Management (SFM) policy active. As of January 2009, the use of AutoIPL with SFM is no longer disabled. The support to allow AutoIPL to be reenabled requires the enabling PTF plus appropriate firmware MCLs.

AutoIPL restriction

The use of AutoIPL in multisystem-capable sysplex configurations where a Sysplex Failure Management (SFM) policy is active was previously disabled by PTF UA43911 (APARs OA26371 and OA26574).

This was because the IPL actions performed by AutoIPL could prevent or delay the successful completion of system isolation (fencing) and system removal actions performed by SFM. With APARs OA26371 and OA26574, when AutoIPL is activated in these configurations where SFM is active, the automatic IPL actions are not performed.

Removal of AutoIPL restriction

The PTF UA45446 for (APARs OA26993 and OA26995), along with underlying LPAR firmware support, now enables AutoIPL to be used in configurations where an SFM policy is active.

With this support, requested AutoIPL actions are performed in accordance with the DIAGxx parmlib member, even when an SFM policy is active in the sysplex. The use of SFM and system isolation to quickly and automatically remove a failed system from the sysplex remains a best practice for sysplex availability. In multisystem sysplexes, the prompt removal of failed systems from the sysplex to permit cleanup of shared resources is of primary importance, enabling the remaining systems in the sysplex to continue normal operation.

Hardware dependencies

The use of AutoIPL on a system in a multisystem-capable sysplex configuration where an SFM policy is active requires the following LIC levels be installed:

- For z9 Systems at Driver-67: MCL006 in EC Stream (LPAR) G40954 (Bundle 38)
- For z10 Systems at Driver-73: MCL009 in EC Stream (LPAR) F85901 (Bundle 45b)
- For z10 Systems at Driver-76: MCL003 in EC Stream (LPAR) N10965 (Bundle 8)

Software installation

PTF UA45446 for (APARs OA26993 and OA26995) should be installed on all systems in the sysplex, via a rolling IPL. This PTF will not be fully effective on the system for which it is being applied until the PTF is applied to all systems in the sysplex; a rolling IPL is sufficient to accomplish the activation.

The PTF and the underlying LPAR firmware support described in “Hardware dependencies” can be installed independently, in any order, and with any timing.

Important: The AutoIPL function must not be used in a sysplex with SFM active until *both* the LPAR firmware support and the software support have been installed on all affected systems.

31.5 Timer support for GDPS

In a GDPS implementation, the GDPS controlling system (K-SYS) performs critical functions for system availability and disaster recovery. It manages disk mirroring via PPRC and performs vital functions for both primary and secondary disk failures:

- For a primary disk failure, it performs a HyperSwap to the secondary disk.
- For a secondary disk failure, it performs the PPRC FREEZE function to ensure the data integrity of the mirrored disk at the recovery site.

The GDPS controlling system (K-SYS) normally resides at the recovery site. However, because it is part of the production sysplex, it can be impacted by a disaster at the production site. When cross-site connectivity is lost, it is possible that the K-SYS will lose its time source. This is true in both an ETR and STP environment. When this occurs, the K-SYS will issue the synchronous WTOR IEA015A (for ETR) or IEA394A (for STP) and may not have successfully completed the FREEZE or HyperSwap function.

The K-SYS has its own infrastructure data sets (such as the master catalog, RACF database, and JES2 spool), and it does not access the production disk. Therefore, there is no data integrity problem in allowing it to continue running in LOCAL timing mode when the sysplex time source is lost.

The K-SYS only needs to be synchronized with, and run with a degree of time synchronization that allows it to correctly participate in, heartbeat processing with other systems in the sysplex. This means that the K-SYS could run using the local TOD clock of the CEC for timing and unsynchronized with the sysplex time source for a considerable amount of time.

New function has been added to both the Supervisor timer and XCF to allow the K-SYS to continue running in LOCAL timing mode for up to 80 minutes when the sysplex time source is lost.

Software dependencies

This new function to allow the K-SYS to continue running in LOCAL timing mode without issuing IEA015A or IEA394A for up to 80 minutes requires:

- GDPS V3R6
- z/OS V1R11 or
- z/OS V1R9 or z/OS V1R10 with:
  - APAR OA28323 for Supervisor timer
  - APAR OA26085 for XCF

New function details

This new function allows the GDPS controlling system (K-SYS) to switch to LOCAL timing mode on receipt of a loss of time source machine check for both ETR and STP timing networks.

When either the switch to local machine check handler or STP clock source machine check handler receives a loss of time source machine check, a new GDPS control block is used to determine whether this is a GDPS controlling system. If it is, then:

- The synchronous loss of time source WTOR, IEA015A or IEA394A, is not issued.
- A flag is set to indicate to XCF that this system is a K-SYS running in LOCAL timing mode. This is reset when clock resynchronization occurs or the system is IPLed.
- The timing mode is switched from either ETR or STP timing mode to LOCAL timing mode.
- One of the following messages will be issued, depending on whether the timing mode was ETR or STP:

  IEA261I NO ETR PORTS ARE OPERATIONAL. CPC CONTINUES TO RUN
  IN LOCAL MODE

  IEA381I THE STP FACILITY IS NOT USABLE. SYSTEM CONTINUES
  IN LOCAL MODE.

- XCF allows the K-SYS to continue to run unsynchronized in the sysplex until:
  - It regains synchronism and is in ETR or STP timing mode.
  - The time limit of 80 minutes is reached when XCF will partition the system with WAIT state 0A2-114 for ETR or 0A2-158 for STP.

The message shown in Figure 31-27 is issued when wait state 0A2-114 is loaded after the 80-minute time limit expires.

```
IXC462W XCF IS UNABLE TO ACCESS THE ETRAND HAS PLACED THIS SYSTEM INTO A
NON-RESTARTABLE
WAIT STATE CODE: 0A2 REASON CODE: 114
```

Figure 31-27 IXC462W for WAIT state 0A2-114

The message shown in Figure 31-28 is issued when wait state 0A2-158 is loaded after the 80-minute time limit expires.

```
IXC468W XCF IS UNABLE TO ACCESS THE CTNAND HAS PLACED THIS SYSTEM INTO A
NON-RESTARTABLE
WAIT STATE CODE: 0A2 REASON CODE: 158
```

Figure 31-28 XC462W for WAIT state 0A2-158

New and updated messages

A new highlighted “eventual action” message IXC410E will be issued by XCF on the K-SYS when timer synchronization is lost. Unlike IEA015A and IEA394A, the new message will not cause the system to wait for a reply via a synchronous WTOR. The message will be deleted when sysplex time synchronization is resumed. If time synchronization is not restored within 80 minutes, then the system will enter a non-restartable wait state.

When the K-SYS loses Sysplex Timer synchronization, existing processing will issue message IXC438I with a new insert before issuing the new message IXC410E. Figure 31-29 shows the messages issued when the K-SYS loses time synchronization.

```
IEA261I NO ETR PORTS ARE OPERATIONAL. CPC CONTINUES TO RUN
             IN LOCAL MODE

IEA381I THE STP FACILITY IS NOT USABLE. SYSTEM CONTINUES
            IN LOCAL MODE.

IEA387I STP DATA CANNOT BE ACCESSED. SYSTEM CONTINUES IN LOCAL TIMING MODE


IXC438I COORDINATED TIMING INFORMATION HAS BEEN UPDATED FOR SYSTEM sysname
PREVIOUS {ETR NETID|CTNID}: {etrid|stpid|stpid-etrid}
CURRENT TIMING: LOCAL

IXC410E SYSTEM  SysName LOST SYNCHRONIZATION WITH THE TIMING NETWORK.
LOCAL TIMING MODE WILL BE ALLOWED FOR 80 MINUTES.

IXL161I CF REQUEST TIME ORDERING: NOT-REQUIRED AND NOT-ENABLED
        COUPLING FACILITY SIMDEV.IBM.EN.ND0100000000
                          PARTITION: 00  CPCID: 00
```

Figure 31-29 Messages issued when K-SYS loses time synchronization

Note: When the K-SYS was running in ETR timing mode, IEA261I is issued. When the K-SYS is running in STP timing mode, then IEA381I and possibly IEA387I are issued, depending on the timing of the STP interrupts received.

When the timing network is restored, the K-SYS will automatically regain time synchronization. Figure 31-30 on page 650 shows the messages issued when the K-SYS regains time synchronization.

```
IEA260I THE CPC IS NOW OPERATING IN ETR MODE.

IEA380I THIS SYSTEM IS NOW OPERATING IN STP TIMING MODE.

IXL161I CF REQUEST TIME ORDERING: REQUIRED AND ENABLED
        COUPLING FACILITY SIMDEV.IBM.EN.ND0100000000
                          PARTITION: 00  CPCID: 00

IXC438I COORDINATED TIMING INFORMATION HAS BEEN UPDATED
FOR SYSTEM SYSTEMK1
PREVIOUS CTNID: LOCAL
CURRENT TIMING: {etrid|stpid|stpid-etrid}
```

Figure 31-30 Messages issued when K-SYS regains time synchronization

Note: When the K-SYS regains ETR timing, then IEA260I is issued. When the K-SYS regains STP timing, then IEA380I is issued.

When IXC410E is issued and the K-SYS is running in LOCAL timing mode, a system can only join the sysplex if some other system in the sysplex is able to maintain time synchronization with the timing network. If there is another active system to maintain time synchronization,

then message IXC414I is issued, as shown in Figure 31-31, when a system attempts to join the sysplex.

```
IXC414I CANNOT JOIN SYSPLEX sysplex-name. LOCAL TIMING MODE IS BEING USED BY
THE ACTIVE OR IPLING SYSTEM(S). TIMING NETWORK SYNCHRONIZATION IS REQUIRED.
```

Figure 31-31 IXC414I issued when a system attempts to join the sysplex before the timing network is restored

It is possible for only some systems to lose time synchronization when a disaster or failure occurs. For example, consider GDPS configuration where there are production systems at both site 1 and site 2 and the active K-SYS is at site2. If a failure of cross-site connectivity occurs, resulting in the systems at site 2 losing time synchronization, then the K-SYS will continue in LOCAL timing mode. The production systems at site 2 will issue IEA015A or IEA394A.

The production systems at site 1 will continue running and could be reIPLed without IXC414I being issued. However, the production systems at site 2 would not be able to join the sysplex and IXC414I would be issued until cross-site connectivity was restored and the timing network was fully operational.

Restriction: The indication that a system is a K-SYS is set after the K-SYS has joined the sysplex. This means that if the K-SYS is reIPLed before the timing network is restored, it will not be able to join the sysplex.

The K-SYS can only switch to LOCAL timing mode while it is actively running in the sysplex.

IXCQUERY interface

The existing IXCQUERY interface provides timing information for all systems (z/OS V1R7.0 and above) in the sysplex. See QuasLocalTimingMode, QuasEtrTimingMode, and QuasStpTimingMode bits in the IXCYQUAA mapping macro.

With this new function, this interface provides a means to observe the K-SYS in local timing mode. Without this new function, in a multi-system-capable sysplex, IXCQUERY should never indicate that a system is in local timing mode.

Migration and coexistence

This function may not be fully effective in allowing the K-SYS system to switch to local timing mode until all systems in the sysplex have been migrated to support this function. Systems without this new function (pre-z/OS V1R11 systems without the supporting PTFs) may initiate sysplex partitioning to remove a K-SYS from the sysplex when it is in local timing.

32

Message Flood Automation

Many z/OS systems are troubled by cases where a user program or a z/OS process itself issues a large number of messages to the z/OS consoles in a short time. Cases of hundreds (or even thousands) of messages a second are not uncommon. These messages are often very similar or identical, but are not necessarily so. Techniques to identify similar messages can be very difficult and time-consuming to use.

Message Flood Automation addresses this problem. This implementation does not claim to identify all cases of erroneous behavior, or to take the “correct” action in all cases. Instead, its intention is to identify runaway WTO conditions that can cause severe disruptions to z/OS operation, and to take installation-specified actions in these cases.

The problem that Message Flood Automation is attempting to solve has been with the operating system since the earliest days. It is most often caused by malfunctioning input or output devices that flood the system with a very large number of error messages, usually within a matter of seconds. Message floods can also be caused unintentionally by improperly designed programs that get stuck in a message loop, as well as by malicious programs that deliberately generate large volumes of messages in an attempt to cause a system outage.

Message floods are a concern because they monopolize system resources and prevent the operator (and automation) from seeing and reacting to other system messages in a timely manner. In certain cases, message floods monopolize system resources so much that resource shortages develop and a system outage occurs.

This chapter includes a brief description of the Message Flood Automation that was incorporated into z/OS V1R9.

Also in this chapter, the following changes with z/OS V1R11 are described:

- Message Flood Automation implementation
- Problem statement and solution
- Installation, loading and activating
- Customization and tuning
- Operator commands

32.1 Message Flood Automation implementation

Message Flood Automation is designed to detect and react to a message flood before those consequences have had an opportunity to occur. A defined installation policy allows installations to tailor Message Flood Automation to an individual environment by deciding how quickly the Message Flood Automation is to react to a potential message flood and then, what actions are to be taken if a message flood occurs. Depending on the policy that is established, Message Flood Automation can prevent message buffers from filling, console queues from becoming overly long, and console displays from becoming unreadable.

Because Message Flood Automation deals with the flood messages as they are being generated, large numbers of flood messages do not have the opportunity to accumulate in message buffers or in various queues. This means that there is no need to take action, either automated or manual, to “flush” these unwanted messages from buffers or queues. (In past releases, flushing unwanted messages from the queues of each console was one of the more serious aspects of dealing with message floods.)

Furthermore, Message Flood Automation deals with the messages going to all of the various consoles, including the EMCS consoles used by SDSF and by automation products such as Tivoli® NetView® and Tivoli System Automation.

32.1.1 Message flood problems

Message Flood Automation is an implementation designed to handle disruptions that are caused by the following situations:

- Large numbers of messages to the z/OS consoles can obscure important messages and delay them from being acted on.
- Large numbers of messages to the automation system can delay the processing of normal messages.
- Messages can use excessive CPU and storage resources. Buffering excessive message traffic can use large amounts of virtual and real storage, and can cause SQA to overflow into CSA. This can cause jobs, subsystems, and even complete systems to be delayed or fail.

Messages can be produced at very high volumes due to:

- Malfunctioning I/O devices such as DASD, DASD controllers
- ESCON/FICON switches
- Malfunctioning network devices
- Errant programs (unintentional)
- Malicious programs

Message Flood Automation can react to potential message flooding situations in a matter of tens or hundreds of messages (specifiable by the installation), well before buffers have begun to fill, well before console queues have begun to build, and well before console message rates have begun to become enormous.

Furthermore, its actions do not result in residual buffers or queues of messages that must be “worked down” to return to normal processing. Because its processing is targeted to the messages that are causing the problem, very few uninvolved messages are acted upon.

By contrast, the act of flushing console queues (by using the **K Q** command) can result in throwing away many innocent and often important messages. Message Flood Automation can potentially eliminate the need to issue the **K Q** command by preventing flood messages

from ever reaching a console. Message Flood Automation has a policy that allows installations to target individual messages, individual jobs, or started tasks. By targeting specific messages and units of work, Message Flood Automation is able to minimize the disruption to other work in the system.

32.1.2 MPF processing

The message processing facility (MPF) controls message processing for an MVS system. Message Flood Automation runs as part of MPF processing, which occurs after the control block that represents the message has been created. Message Flood Automation is able to see and alter any processing of the message that occurs prior to the creation of this control block.

Note: Some automation products replace the Write-To-Operator (WTO) Supervisor Call (SVC) with their own code and then invoke the WTO code when they are finished. Message Flood Automation is able to see and react to messages that have been "front-ended" by other automation in this way.

32.1.3 MPFLSTxx parmlib member

The IEAVMXIT installation exit or an MPF installation exit (one that you specify on the USEREXIT parameter in an MPFLSTxx parmlib member) allows you to modify message processing in a system or sysplex. The IEAVMXIT installation exit was part of this support for z/OS V1R9 and z/OS V1R10.

The following changes have been introduced with z/OS V1R11.

z/OS V1R11 implementation

Message Flood Automation access to messages was restricted, because the IEAVMXIT might not receive all messages. Message Flood Automation only saw messages that did not have a specific MPF exit defined. This means that Message Flood Automation could not react to a flood of messages that had MPF exits.

Note the following changes with z/OS V1R11:

- Message Flood Automation is now capable of acting on all messages. However, it continues to ignore WTOR, WTO minor lines, and operator commands.
- Message Flood Automation had a limit of 30 SPECIFIC message IDs that can be monitored. This number is being increased to 50 SPECIFIC message IDs with z/OS V1R11.

MPFLSTxx parmlib member

Because all messages are processed by MPF, the MPFLSTxx parmlib member tells MPF what to do with each message. The following is a list of changes that can be made to a particular message or set of messages:

Suppression Message appears in a hardcopy log but not on a console.

SUP(YES/NO)

Automation This lets the automation subsystem know to process a particular message.

AUTO(YES/NO/token)

Specifying AUTO(YES) will route the message to EMCS consoles with the AUTO attribute.

Presentation This controls color, highlighting, and intensity attributes that the system uses when displaying messages on an operator console. You can specify these attributes on the .MSGCOLR statement.

32.1.4 MPF processing exit

MPF processing is specific to a certain type of message or a particular message ID that is defined in the MPFLSTxx parmlib member. Figure 32-1 on page 656 shows when MPF processing is invoked when a message is issued.

This exit is used primarily to perform the following kinds of processing of a message:

- Modify the presentation of a message
- Modify the routing of a message
- Suppress or affect the automation of a message

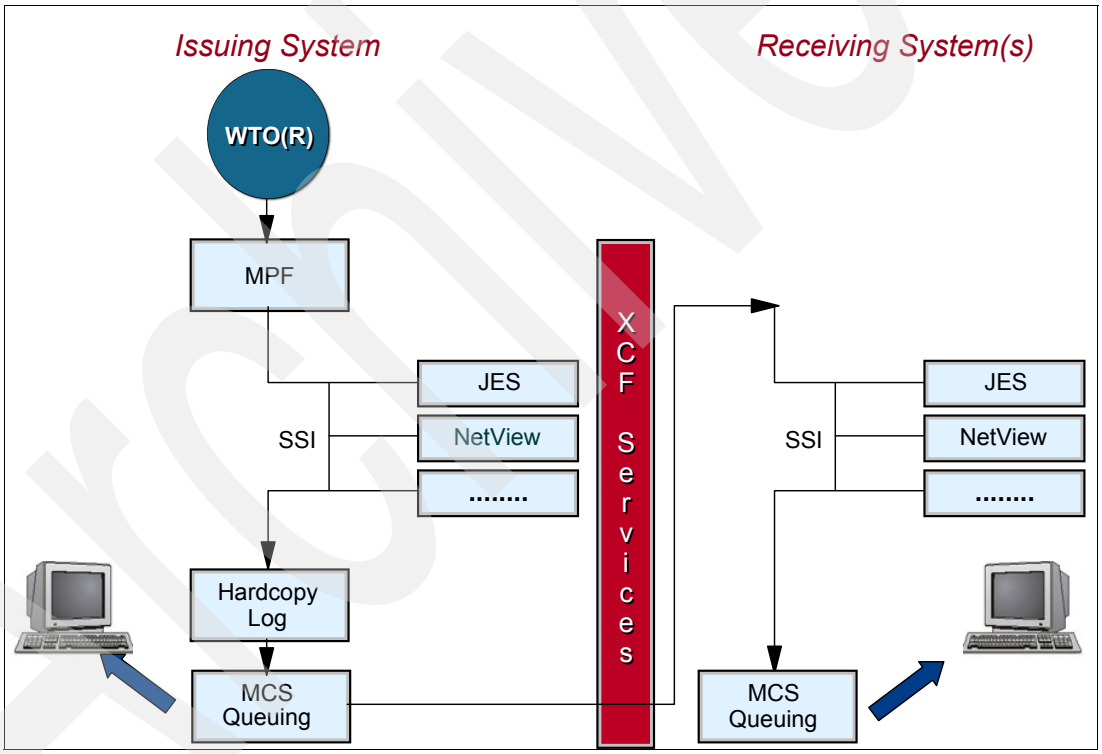


Figure 32-1 Message processing exit (MPF or IEAVMXIT)

Message flood use of this exit

Message Flood Automation is implemented as a message processing facility (MPF) IEAVMXIT routine that is called as a part of z/OS WTO processing. The exit examines each message in the z/OS system, and attempts to identify when too many WTOs are being issued and by whom.

It then takes appropriate action, usually to suppress the message from being displayed at a z/OS console, and to indicate that automation processing is not required. It can also issue commands to cancel the user or process.

MPFLSTxx parmlib member

At IPL, the system uses the MPFLSTxx member or members indicated on the MPF keyword on the INIT statement in the CONSOLxx parmlib member. You can specify multiple MPFLSTxx members on the MPF keyword. In a sysplex, MPF processing has system scope; thus, you must plan MPF processing on each system in the sysplex.

The MPFLSTxx parmlib member contains statements that can affect the display, automation, and retention of messages. During MPF processing, the RETAIN, AUTO and SUP parameters on an MPFLSTxx parmlib member are processed first. Then either a user exit (specified by the USEREXIT parameter) is invoked or IEAVMXIT is invoked, but not both. A small part of an MPFLSTxx parmlib member is shown in Figure 32-2 on page 657.

```
.NO_ENTRY,SUP(NO),RETAIN(I),AUTO(YES)
.DEFAULT,SUP(NO),RETAIN(NO),AUTO(NO)
IST1051I,SUP(YES),RETAIN(YES),AUTO(YES)
IST1062I,SUP(YES),RETAIN(YES),AUTO(YES)
AOF*,SUP(NO),RETAIN(NO),AUTO(YES)
CSA*,SUP(YES),RETAIN(NO),AUTO(YES)
EQQ*,SUP(NO),AUTO(YES)
EVJ*,SUP(NO),AUTO(YES)
IXG054A,USEREXIT(MPFOPLOC)
IEF125I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF403I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF126I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF404I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEE391A,USEREXIT(MPFSMFC)
IEE366I,USEREXIT(MPFSMFC)
BDT313O,USEREXIT(MPFSBDTN)
```

Figure 32-2 MPFLSTxx parmlib member

For the message IDs shown in Figure 32-2, the ones with the USEREXIT parameter will use the specific MPF exit specified as shown in Figure 32-3 on page 658. All other messages will use the IEAVMXIT exit, which goes through Message Flood Automation; this explains why individual messages do not go through both paths. Therefore, Message Flood Automation cannot override specifications set by individual MPF exits.

Note: If an MPFLSTxx entry does not exist for a message, then the settings from the NO_ENTRY specification are applied. NO_ENTRY allows you to specify the default processing you want for messages that are *not* identified in any of the active MPFLSTxx parmlib members.

Using the IEAVMXIT exit

Prior to z/OS V1R11, Message Flood Automation message processing runs together with IEAVMXIT. Therefore, it can override the RETAIN, AUTO and SUP specifications set by the MPFLSTxx entry for the message or set by the NO_ENTRY specification.

Important: Figure 32-3 on page 658 shows at which point IEAVMXIT is entered with z/OS V1R09 and z/OS V1R10.

This processing of messages is removed with z/OS V1R11.

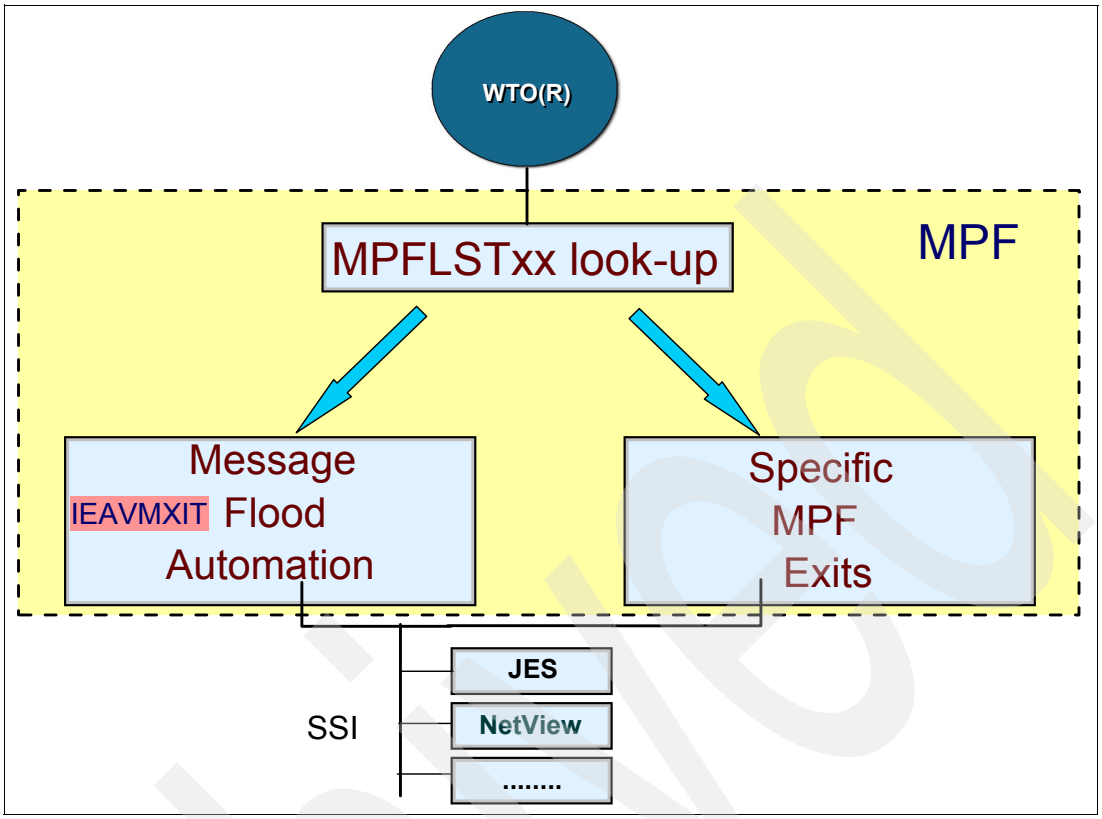


Figure 32-3 Message Flood Automation IEAVMXIT exit

32.2 z/OS V1R11 processing

Figure 32-4 on page 659 illustrates the difference in the changed Message Flood Automation processing from previous releases (as shown in Figure 32-3). Message Flood Automation processing is now performed after the MPFLSTxx process, which allows message processing to not overlook messages and cause Message Flood Automation to be removed from the IEAVMXIT. Because it is now performed before any specific MPF exit processing, this allows an MPF exit to override the changes made by Message Flood Automation.

Note: Because Message Flood Automation message processing runs *before* MPF exit processing, MPF exits can override the RETAIN, AUTO, and SUP specifications set by Message Flood Automation.

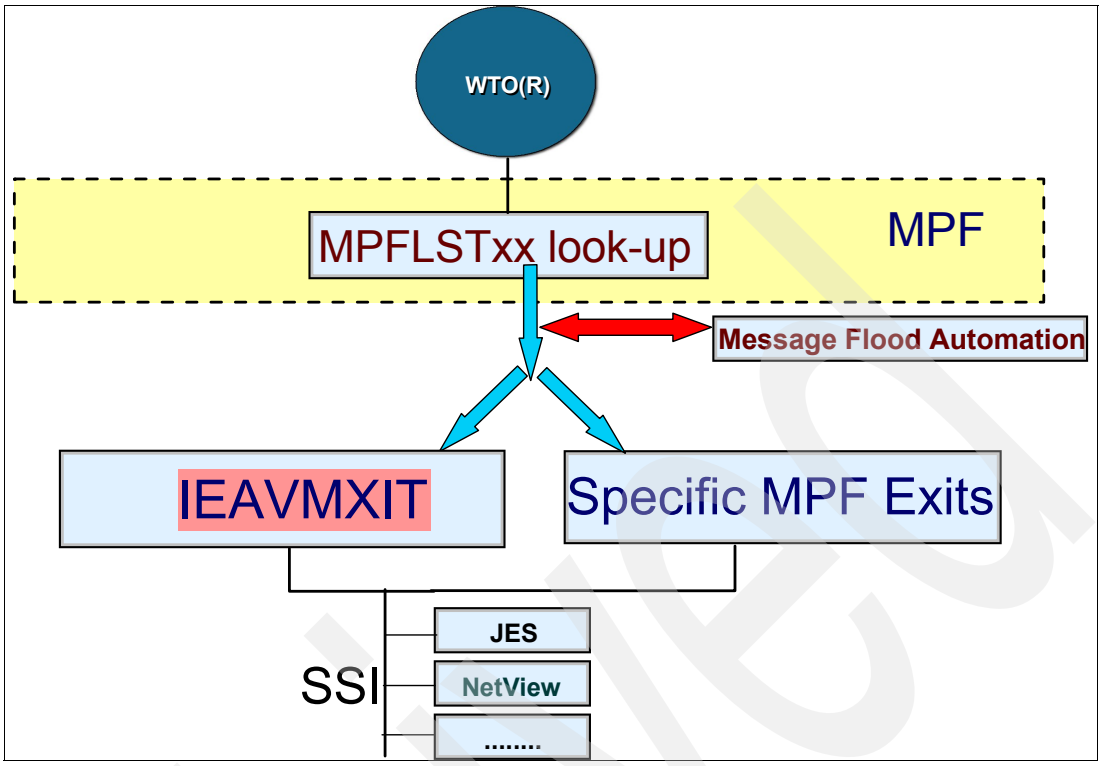


Figure 32-4 Message processing change with z/OS V1R11

32.2.1 z/OS V1R11 command processing changes

The Message Flood Automation enhancement moves the command processing to be under the normal flow of commands. This gives Message Flood Automation commands security protection because all commands have to be authorized before being processed. (Previously, there were no authorization checks on Message Flood Automation commands.) The commands will also run in the normal command address space (*MASTER*) instead of the MGCRE issuer's address space.

The **SET MSGFLD=xx** command is changed to run in the master address space instead of running in the DUMPSRV address space.

It is also important to note that the Message Flood Automation had a restriction that its **SET** command could not be combined with other z/OS **SET** commands. This restriction has now been lifted. An operator can now issue a **SET** command with multiple options.

Note: The **SETMF FREE** command has been deleted, because Message Flood Automation control blocks exist for the life of the system.

Command processing prior to z/OS V1R11

Figure 32-5 on page 660 shows where Message Flood Automation receives control during command processing prior to z/OS V1R11. Previously, Message Flood Automation command processing was invoked as a command exit. This occurred before normal system command processing, and it prohibited the command authorization process for the Message Flood Automation commands.

The figure also shows where Message Flood Automation operates relative to other automation, such as the NetView Command Automation and NetView Subsystem interface processing.

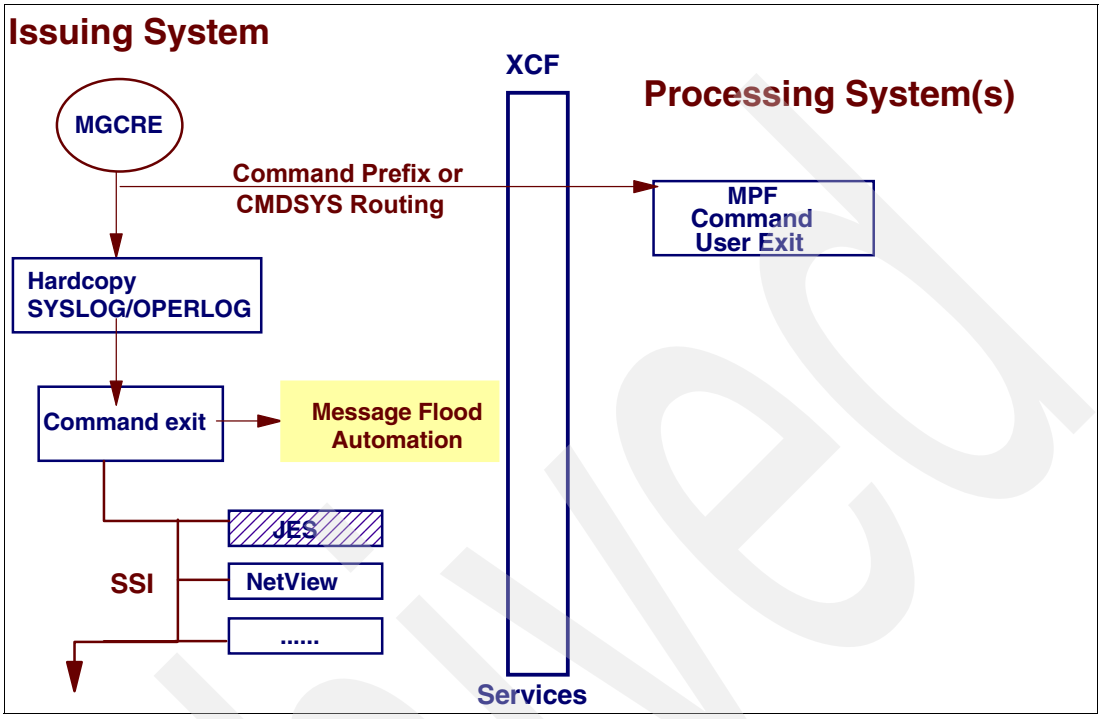


Figure 32-5 Command processing prior to z/OS V1R11

Command processing with z/OS V1R11

Figure 32-6 on page 661 shows where Message Flood Automation receives control during command processing with z/OS V1R11. Now, Message Flood Automation is part of the normal system command processing. It allows for command authorization and the use of the **L=** operand to control where the output is to be displayed.

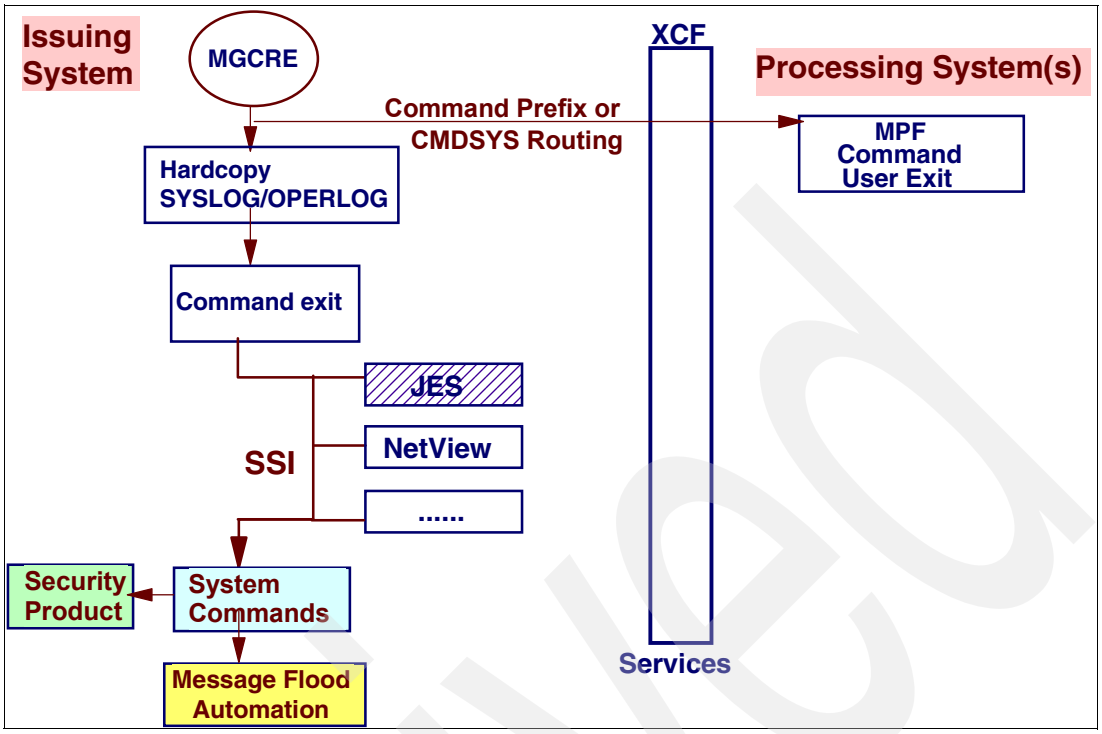


Figure 32-6 Command processing with z/OS V1R11

32.3 Migrations to z/OS V1R11

If you previously used Message Flood Automation, it is important to follow the migration and installation steps to have a successful initialization. The MSGFLDxx parmlib member still works unmodified. If you have been using Message Flood Automation and want to install the z/OS V1R11 version then take the following actions, keeping in mind how you are using this function in the current system.

Note: To run Message Flood Automation for z/OS V1R9 and z/OS V1R10, compatibility APAR OA25602 is required.

Users of IEAVMXIT only

If Message Flood Automation is the only user of IEAVMXIT, then before you load the initial program of z/OS V1R11, follow these steps:

1. Replace the UEXIT(Y) parameter with the UEXIT(N) parameter on the INIT statement within the CONSOLxx parmlib member.
2. Remove IEAVMXIT from the LINKLIST concatenation.

Multiple users of IEAVMXIT

If Message Flood Automation is not the only user of IEAVMXIT (that is, if you have a preexisting IEAVMXIT exit and you want to modify the exit to make sure that it coexists with Message Flood Automation), then prior to IPLing IPL z/OS V1R11, take *one* of the following actions:

- Fall back to the earlier version of the exit.

- Or, remove the invocation of Message Flood Automation from the exit. Then reassemble and rebind the exit and verify that the new exit has replaced the old exit in the LINKLIST concatenation.
- Or, remove the invocation of Message Flood Automation from the CNZZVXT2 sample program if you have moved the IEAVMXIT exit function into the sample program. Then reassemble and rebind the exit and verify that the new exit has replaced the old exit in the LINKLIST concatenation.

Users of the system command exit

If Message Flood Automation is the *only* user of the system command exit, then before you load the initial program of z/OS V1R11, take the following action:

- Remove the .CMD statement from all MPFLSTxx members in which it has been specified. If Message Flood Automation is not the only user of the exit, simply remove the CNZZCMXT entry from all .CMD statements.

32.3.1 Other migration concerns

Remove all invocations of the **SETMF FREE** command from the system, including COMMNDxx and automation usage.

You are not required to make changes to the MSGFLDxx parmlib member.

Because Message Flood Automation commands are now subject to authorization checking, you can define the Message Flood Automation commands to your security products. For profiles defined to security products, see the table “MVS Commands, RACF Access Authorities, and Resource Names” in *z/OS MVS System Commands*, SA22-7627.

33

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 though 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 V1R11 regarding the following z/OS UNIX System Services-related applications:

- Network File System services
- z/OS Distributed File Service SMB

33.1 Network File System

With z/OS V1R11, the following enhancements have been made to the Network File System (NFS):

- NFS server mount symlink support
- NFS V4 server delegation support
- Elimination of mvslogin/mvslogout for RPCSEC_GSS
- NFS V4 client RPCSEC_GSS Security negotiation
- Convert NFS client memory management 0C1 abends to 0F4 or 801 abends
- NFS client message globalization
- More meaningful NFS client reason codes
- NFS debug tracing enhancement to reduce data volume
- Dynamically detect and report external functional request delay
- Completion messages for operator commands

Note: The numbers in brackets are the formal line item numbers that are related to each topic and that are provided just to provide a complete information.

33.1.1 NFS server mount symlink support

Previously, NFS V4 mounts did not support symbolic links (a *symbolic link* is an alias for a file or directory, possibly in another file system), which can be problematic.

The problem arises because NFS supports two file system types:

- z/OS UNIX file systems - these are identified by an HFS prefix on path names (for example, /prfx/a/b/c)
- MVS data sets - these are identified by the lack of a prefix on path names (for example, a.b.c)

Because NFS V4 emulates mounts by a series of lookup requests, the NFS client sends lookup requests for one qualifier at a time:

- PUTROOTFH, LOOKUP, GETFH, GETATTR
- PUTFH, LOOKUP, GETFH, GETATTR

When the NFS client detects a symbolic link, it sends a READLINK request and later restarts LOOKUP processing with PUTROOTFH and LOOKUP using a relative path name (a path name without a prefix).

The z/OS NFS server sees the restart and interprets the LOOKUP path specification as an MVS path and the processing fails.

Expanded file system path type specification

Now, NFS server file system type prefix support provides the capability to support NFS V4 mounts with symbolic links based on having three prefix settings:

- HFSPREFIX - the default value is /HFS.
- MVSPREFIX - the default value is /MVS.

- IMPPREFIX - the default value is MVS.

Exploiting the new path prefix settings by using IMPPREFIX(HFS) or IMPPREFIX(HFS,MVS) the desired results are received.

Note: When you specify IMPPREFIX(HFS,MVS), a UNIX path is searched first. If not found, the search is continued by looking for an MVS path.

The NFS client restarts the LOOKUP processing after detecting a symbolic link. The z/OS NFS server now interprets LOOKUP correctly as addressed to the z/OS UNIX file system and the processing succeeds.

Coexistence consideration

The coexistence APAR OA25864 allows the use of the NFS server site attribute file with the new attributes on previous NFS server levels. The PTFs are listed in Table 33-1 on page 665.

Table 33-1 PTFs for APAR OA25864

PTF	NFS release	z/OS release
UA46260	18N	z/OS V1R8
UA46261	19N	z/OS V1R9
UA46259	AN0	z/OS V1R10

33.1.2 NFS V4 server delegation support

NFS V4 protocol includes an optional capability to delegate file management to an NFS client. Previously, this function was not available.

This capability has been added to the z/OS NFS server for MVS data sets. It provides the following benefits:

- Additional NFS V4 protocol compliance
- Potential MVS data set access performance improvement

Installation considerations

For activation there are new site attributes for NFS start-up processing:

DELEGATION This turns on delegation mode.

NODELEGATION This turns off delegation mode and is the default setting.

For dynamic changes of this setting, new operator modify commands have been created:

```
F mvsnfs,V4DELG=ON - Turn on delegation mode
F mvsnfs,V4DELG=OFF- Turn off delegation mode
```

In these commands, **mvsnfs** is the name of the NFS server. Furthermore, a new **LISTLOCK** operator modify command also lists active delegations.

Coexistence consideration

The coexistence APAR OA25864, previously mentioned in “Coexistence consideration” and in Table 33-1, allows the use of the NFS server site attribute file with new attributes on previous NFS server levels. The PTFs associated with this APAR are shown in Table 33-1 on page 665.

33.1.3 Elimination of mvslogin and mvslogout for RPCSEC_GSS

Previously, the following problems existed:

- In SAF modes of operation, the z/OS NFS server required NFS client users to issue **mvslogin** and **mvslogout** for RPCSEC_GSS workloads.
- Different client users on the same client machine at the same time were unable to use the same RACF user ID to perform an mvslogin to the z/OS NFS Server.
- Communication using RPCSEC_GSS between a Windows NFS client and a z/OS NFS SAF server was not supported.

Updates were made to the z/OS NFS server such that:

- Using **mvslogin** and **mvslogout** is no longer required for RPCSEC_GSS workloads.
- For system authentication for which mvslogin and **mvslogout** is still needed, a RACF user ID can be shared across client users.

The new support provides the following benefits:

- It facilitates processing because an **mvslogin** and **mvslogout** for RPCSEC requests is no longer required.
- It facilitates administration by allowing the same RACF user ID to be used for **mvslogin** by separate client users on the same client machine at the same time.
- Windows NFS clients can now communicate with the z/OS NFS server using RPCSEC_GSS.

33.1.4 NFS V4 Client RPCSEC_GSS security negotiation

Previously, a z/OS NFS client was unable to determine the security policy deployed on the server's name space.

Updates have been made to the z/OS NFS client to enable it to determine and negotiate the security policy with the server. This allows the client to now dynamically determine and negotiate the security policy on the server mounts.

Installation considerations

Consider the following implementation concerns:

- Not specifying the security flavor in the "secure" keyword in the **mount** command causes the client to negotiate security.
  - Security negotiation during mount is not done when a security flavor is specified in the "secure" keyword.
  - During mount, when data caching is specified and a security negotiation was attempted, then data caching is turned off.
  - The "secure" keyword in the **mount** command now accepts the sys flavor.
- For existing files or objects, security negotiation is not done when data caching is on for that file or object.
- The z/OS NFS client's preferred flavors in descending order of preference are `sys`, `krb5`, `krb5i`, and `krb5p`.
- On existing mount points or objects, security is negotiated only when it is an upgrade to a more secure flavor.

- A new errno value, `JRNfs_Wrongsec`, which has a numerical value of 1013 associated, can be returned when a security negotiation either cannot be done or was attempted but failed.

33.1.5 Convert NFS client memory management from S0C1 to S0F4 or U0801 abend

NFS client memory management previously generated an S0C1 abend if it detected an error during its processing. This has been converted and the NFS client generates the following abends:

S0F4 This is provided during cross-memory processing.

U0801 This is generated during RPC request processing.

This provides improved NFS Client RAS because it allows this situation to be more readily identified and avoids the z/OS UNIX abend dump suppression.

33.1.6 NFS client message globalization

Previously, the z/OS NFS client only supported the generation of console messages in English.

With the message globalization line item, the z/OS NFS client is able to provide the national language support for Japanese translation for its console messages. The client console messages can now be translated into Japanese.

33.1.7 More meaningful NFS client reason codes

Previously, the NFS client returned many reason codes that were “file and line” type, meaning they referred to a point in the source code. These required that complaints be referred to those with access to the source code to determine the nature of the error.

Now, more meaningful NFS client reason codes are used and provided within the standard manuals. A problem with a reason code that is officially documented can be researched at the customer location and resolved more quickly. The new reason codes are documented in *z/OS Network File System Guide and Reference*, SC26-7417.

33.1.8 NFS debug tracing enhancements

The following enhancements have been added to improve performance in the tracing functionality of the NFS server:

- Server log events are processed run-time, causing performance penalties. Reducing the logging eliminates the vast majority of these events.
- Component trace events are processed at dump processing, making these events less intrusive run-time.
- Internal code optimization was also included.

As a result of these enhancements, the server will perform better, according to the settings used. The logs will contain less detail, but the CTraces will still contain a larger set of server events.

Installation considerations

The server startup job no longer accepts the **–DEBUGn** (n=0-9) parameter. Instead, only **ERROR**, **WARN**, or **INFO** are accepted to specify the amount of tracing into the server log files.

The new, optional startup job parameter **STARTFAIL** has been introduced, as explained here:

- Specifying **STARTFAIL=DUMP** forces a dump when the server terminates itself.
- Using **STARTFAIL=IGNORE** will not force a dump. This is the default value, when **STARTFAIL** is not specified.

No changes have been added in component trace invocation.

Migration and coexistence considerations

- Customers specifying **–DEBUGn** in their server startup now receive a console message at startup stating that the parameter is not accepted. Nevertheless, the server starts up.
- The most information that goes into the server logs is **INFO**, which includes **WARN** and **ERROR** types. This is the default, and is the value assumed if the **–DEBUGn** parameter is encountered.
- Customers wanting to obtain debug information for a server that has terminated during its startup procedure need to update their server startup job to include **-STARTFAIL=DUMP** as a parameter. The component trace specification for the server in SYS1.PARMLIB also indicates the desired level of debug information.

33.1.9 Dynamically detect and report external functional request delays

A new timer-activated event to check for outstanding external calls generates console messages when such a call is found; when many such outstanding calls are found; and when all outstanding calls have been resolved, which means that no more delay events exist.

This means that operators cab now determine if an indicated file system is not behaving or performing as expected.

The support is invoked by the server site attribute **DlyDTimeout(nn)**:

- Valid values of nn are 0 (off), or 5 - 60; the values are specified in seconds.
- If not specified, the default value is 10 seconds.

To dynamically change the value, an operator command with the same valid values has been introduced:

```
Modify mvsnfs,DlyDTimeout=nn
```

In this command, *mvsnfs* is the name of the NFS server.

The following list contains the new console messages:

GFSA1030W This is displayed when a delay is detected.

GFSA1031I This is shown when delays are resolved.

GFSA1032E This indicates that the DlyDTimeout parameter is out of range.

GFSA1033E This is displayed when multiple delays are detected. It prevents too many GFSA1030W messages from flooding the console. Each message GFSA1030W can still appear in the logs.

The most interesting console messages are listed here:

```
GFSA1030W mvsnfs Network File System Server subtask is waiting for a reply from
the OMVS.ZFS.FS (ZFS)
GFSA1031I mvsnfs Network File System Server subtask was waiting for reply from
v_lookup for the OMVS.ZFS.FS for 23 sec
GFSA1033E mvsnfs There are many delays detected. There is more information in
Network File System Log.
```

In these messages, *mvsnfs* is the name of the NFS server.

Coexistence consideration

APAR OA25864, as previously mentioned in 33.1.1, “NFS server mount symlink support” on page 664 and listed in Table 33-1 on page 665, already allows the use of the NFS server site attribute file with new attributes on previous NFS server levels and without failing to start up when an unknown site attribute is encountered. The PTFs associated with this APAR are listed in the table.

33.1.10 Completion messages for operator commands

A generic completion message was created so that an operator command that had no information dependency in its responses can simply generate an indication that the message was completed.

Important: An operator issuing a command will now get a positive indicator that the command was accepted and executed.

The following new console message is displayed:

```
GFSA796I <mvsnfs> commandstr completed successfully
```

In this message, <mvsnfs> is the name of the NFS server and *commandstr* is the command entered.

33.2 z/OS Distributed File Service SMB

With z/OS V1R11, the following enhancements are available with the DFS SMB support:

- Windows Vista® is now supported.
- The SMB installation no longer requires UID(0).
- User credential information is no longer cached.
- Dynamic trace table support has been added.

33.2.1 Windows Vista compatibility and usage

Windows Vista is now supported, and compatibility has been tested. Support for NTLMv2 authentication is added.

Note: To access SMB at prior releases using Windows Vista, it was necessary to change the default authentication mechanism used. We have removed this restriction. In addition, we verified the server’s behavior, and now list Vista as a supported client.

Using NTLMv2 authentication requires that the domain be correctly sent from the client. You might need to issue a **net use** command similar to the following:

```
net use * \\zOS01\myshare /user:my.domain.name\userid password
```

In this command, `zOS01` is the computer name, `myshare` is the share name, `userid` is the client's SMB user ID, and `password` is the z/OS or SMB password.

Note: See “Windows Vista” in the chapter “Accessing data” in *z/OS Distributed File Service SMB Administration*, SC24-5918, for more information.

33.2.2 Elimination of the UID(0) requirement for SMB installation

Because having many users with UID(0) is undesirable, z/OS SMB can now be started under the authority of a user with a non-zero UID. Using the BPX.SUPERUSER FACILITY, z/OS SMB gets into superuser mode.

It is the TSO user doing the installation of SMB that needs to be permitted to FACILITY profile BPX.SUPERUSER. See *RACF Security Administrator's Guide*, SA22-7683, for more information.

Note: z/OS SMB must be able to read its configuration files before getting into superuser mode.

33.2.3 Removal of ACEE caching

Previously, z/OS SMB cached user credentials (ACEEs). Adding or removing a user from a group had no effect until the z/OS SMB server was restarted.

z/OS SMB now uses **pthread_security_np()** to manage user credentials.

Important: Changes to access rights now take effect without restarting the server.

Because a migration action is required due to the removal of ACEE caching, SMB requires access to the following FACILITY class profiles:

- BPX.DAEMON
- BPX.SERVER

Note: See “Defining RACF definitions for SMB” in *z/OS Distributed File Service SMB Administration*, SC24-5918, for more information.

33.2.4 Dynamic trace table support

Previously, when collecting documentation for a problem, the table was often too small. Changing the size required restarting the server.

With z/OS V1R11, SMB allows you to allocate a new, larger table using a **modify** command. This is achieved by introducing a new parameter **tsize** on the **dfskern daemon trace** command.

In the following example, the trace table size is set to 100 MB:

```
modify DFS,send dfskern,trace,tsize,100M
```

Note: See “modify dfs processes” in *z/OS Distributed File Service SMB Administration*, SC24-5918, for more information.

Do not use a changed trace table size unless you are asked to by IBM support, because the default size of 64 MB is used for normal operation.

Note: The **_IOE_TRACE_TABLE_SIZE** environment variable can be set if another value is desired, although this is not recommended.

A

EAV VSAM CA size

This appendix provides a new tool to help you find data sets that will have a larger CA size. Information about the tool is also given.

A.1 Introduction to EAV VSAM CA size

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 because a CA must be compatible with the multicylinder unit (MCU) for cylinder-managed space. 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 additional volumes that may be ones which support cylinder-managed space. Also note that VSAM data sets physically copied from a non-EAV to an EAV may have an incompatible CA and thus are not EAS-eligible. This means extends for additional space do not use cylinder-managed space.

The system may adjust the primary and secondary quantity to select a CA.

As a consequence, the VSAM data set index CI size may change when defined or redefined. When a VSAM data set or ICF Catalog is defined or redefined on z/OS V1R10 or higher, the index cisize may be increased.

This can occur if the data CA size was in tracks and the amount specified was *not* 1, 3, 5, 7, 9, or 15 tracks. For example, if the space requested was TRK(2,2), then the allocation amount will be TRK(3,3) on z/OS V1R10 or higher. Because the data CA size is now larger, the minimum index cisize may increase. For applications such as IMS™ or CICS® that have (or may have) a static LSR pool definition, then a change to the index cisize may cause a data set to no longer open after being redefined.

A.2 JCL and documentation for the tool

Example A-1 contains the JCL to run the tool as well as information regarding the tool itself.

Example A-1 JCL to run the tool

```
//INDXCI10 JOB ,'INDXCHANGE ZOSR10',CLASS=A,
//         MSGCLASS=H,NOTIFY=&SYSUID
//*
//STEP1     EXEC PGM=IKJEFT1B
//SYSPROC   DD DISP=SHR,DSN=INDXCI10.JCL.CNTL
//* the above DSN should be this data set or wherever you placed the
//* member INDXCI10
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
   %INDXCI10  ** CATALOG(SYS1.ICFCAT1)
   %INDXCI10  CICS.** CATALOG(SYS1.ICFCAT)
   %INDXCI10  **
/*
//*
//* A filter key of '**' causes a search of the entire catalog in the
//* CATALOG() parm. Anything other than a '**' would do a search based
//* on the key and report accordingly. In the above examples the first
//* will search the catalog SYS1.ICFCAT1 for any vsam data sets that
//* will have the index cisize increased due to the change in z/OS 1.10
//* that may increase the CA size. The second example will search the
```

```
//* catalog SYS1.ICFCAT2 for any vsam data sets that begin with CICS
//* that will have the index cisize increased due to the change in
//* z/OS 1.10 that may increase the CA size. The third example will
//* search all catalog for vsam data sets that may have the index
//* cisize increased due to the change in z/OS 1.10 that may increase
//* the CA size.
//*
//* Specifying a catalog name in the CATALOG parm is the recommended
//* approach with a filter key of '**' or a high level qualifier as
//* shown in the above examples.
//*
//* This tool reports only datasets with INDEX components which
//* includes Catalogs,KSDSs,VRRDSs and AIXs.
//*
//* If the CATALOG() parm is omitted, then a filter key of '**' is
//* recommended. This would cause a search of every catalog in the
//* master catalog of the system this is run on. Note that this may
//* result in a lot of the following messages to the system console:
//* 0008 IEF238D INDXCI10 - REPLY DEVICE NAME OR 'CANCEL'.
//* IEF244I INDXCI10 STEP1 - UNABLE TO ALLOCATE 1 UNIT(S) 560
//*         AT LEAST 1 OFFLINE UNIT(S) NEEDED.
//* IEF877E INDXCI10 NEEDS 1 UNIT(S) 561
//* FOR STEP1 SYS00006
//* FOR VOLUME WITPAK
//* OFFLINE, NOT ACCESSIBLE
//* 0800 0803-0804 080A-080C 080E 0810 0812-0813 0816-0819 081D-0820 0822
//* 0825-0827 082A-082F 0834-0837 083A 083F 08C1 08C3-08FF 0E01-0EFF
//* 0F4F-0F5F 0F63-0FBF
//* :
//* IEF878I END OF IEF877E FOR INDXCI10 STEP1 SYS00006
//*
//* A filter key of anything else when the CATALOG() parm is
//* omitted would do a search of Master Catalog ONLY.
//*
//* If there catalog entries for vsam data sets that the volumes
//* offline or no longer exist you will receive the following
//* two messages:
//* The Catalog Search Interface, ended with an error in  catalog module
//* IGGOCLFS with a 100-4
//* The Catalog Search Interface, detected an  error 50-5 in catalog
//* module IGGOCLE3 for entry SOME.VSAM.DATA.SET
//*
//* If you search a very large catalog with a filter key of '**' you
//* may receive the following message:
//* The Catalog Search Interface, ended with an error in  catalog module
//* IGGOCLFS with a 100-8
//* In this case there are more entries in this catalog than can be
//* processed via the CSI (Catalog Search Interface). You will need to
//* use a more restrictive filter key such as the second example above.
//*
```

A.3 Find those data sets with a larger CA size.

Example A-2 contains a REXX procedure to find data sets that have a larger CA size.

Example A-2 Tool to find larger CA size data sets

```
/* REXX                                                              */
/*-------------------------------------------------------------------*/
/*      NAME: INDXCISZ which was modeled after IGGCSIRX
 Description: This rexx exec is used to call the catalog search
              interface.
       Input: Catalog name to be searched for any datasets who's
              INDEX CI Size may be affected during migration to
              R10. If CI size remains the same, only change in DATA CA
              size is mentioned.
              Filter key -
              A filter key of '**' causes the entire catalog in the
              CATALOG() parm to be searched for all datasets in it.
              This combination is recommended.

      Output: Data returned from the catalog search interface for
              which a parsing routine exists.
              This tool only reports datasets with INDEX components
              which includes Catalogs,KSDSs,VRRDSs and AIXs.

      Condition Codes:
       None
*/
/*-------------------------------------------------------------------*/
 arg parmin                          /* Save any parms passed to exec*/
 upper parmin                        /* Convert parms to uppercase   */
 /*TRACE A */
 Call Initialize                     /* Initialize exec variables    */
 Call Call_IGGCSI00                  /* Call catalog search interface*/
 /*TRACE OFF */
 exit(0)

/*************************** Subroutines *****************************/
/*-------------------------------------------------------------------*/
/* Initialize the variables used by this exec                        */
/*-------------------------------------------------------------------*/
Initialize:
 numeric digits 12
 if words(parmin) > 0 then           /* Were parms passed to exec?   */
  do                                 /* Yes,                         */
   If pos('HELP',parmin) > 0 then Call Write_Help
   key = word(parmin,1)              /* Set filter key to 1st word   */
  end
 else
  if SYSVAR(SYSENV) = 'FORE' then    /* if running in foreground     */
   do
    SAY 'INDXCISZ does not run in the foreground' /*not in foreground*/
    exit(20)                         /* Exit                         */
   end
  else
```

```
   do
    say ' This exec requires an input parm'
    say ' of the filter key to be passed to IGGCSI00.            '
    exit(20)
   end
 ind = 0
 header = 0
 exitrc = 0                        /* Set exit rc to zero         */
 count = 0
 x = 0
/* Set default CSI parameter list                                  */
 MODRSNRC = SUBSTR(' ',1,4)          /* CLEAR MODULE/RETURN/REASON   */
 CSIFILTK = SUBSTR(KEY,1,44)         /* MOVE FILTER KEY INTO LIST    */
 CSICATNM = SUBSTR(' ',1,44)         /* CLEAR CATALOG NAME     */
 CSIRESNM = SUBSTR(' ',1,44)         /* CLEAR RESUME NAME            */
 CSIDTYPS = SUBSTR('CG',1,16)        /* RETURN CLUSTER AND AIXs      */
 CSICLDI  = SUBSTR('Y',1,1)          /* INDICATE DATA AND INDEX      */
 CSIRESUM = SUBSTR(' ',1,1)          /* CLEAR RESUME FLAG            */
 CSIS1CAT = SUBSTR(' ',1,1)          /* INDICATE SEARCH > 1 CATALOGS */
 CSIOPTNS = SUBSTR(' ',1,1)          /* An F entry = fullword lengths*/
                                     /* anything else use halfwords  */
 CSINUMEN = '0005'X                  /*   INIT NUMBER OF FIELDS      */
 CSIFLD1    = ''                     /*   INIT FIELD 1 FOR VSAMTYPE  */
 CSIFLD1    = CSIFLD1||SUBSTR('VSAMTYPE',1,8)
 CSIFLD1    = CSIFLD1||SUBSTR('NOTRKAU',1,8) /* Data CA size      */
 CSIFLD1    = CSIFLD1||SUBSTR('AMDCIREC',1,8) /* CI SIZE           */
 CSIFLD1    = CSIFLD1||SUBSTR('VSAMSTAT',1,8) /* VSAM STATS        */
 CSIFLD1    = CSIFLD1||SUBSTR('AMDKEY  ',1,8) /* KEY LENGTH        */
 fieldkey_count = c2d(CSINUMEN)    /* Get count of field keys used */

  If pos('CATALOG(',parmin) > 1 then
   do
    /* Get the catalog name passed by the user */
    parse var parmin 'CATALOG(' catalog_name ')' junk
    CSICATNM = SUBSTR(catalog_name,1,44)
   end
 /* Build the selection criteria fields part of parameter list      */
 CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIOPTNS
 CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
 CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1

 /* Initialize and build work area output part of parameter list    */
 CATNAMET = SUBSTR(' ',1,44)
 DNAMET   = SUBSTR(' ',1,44)
 PREVNAME = ''                      /* Clear previous name variable  */
 RESUME   = 'Y'

 WORKLEN = 1048576                                          /*@01C*/
 DWORK = '00100000'X || COPIES('00'X,WORKLEN-4)            /*@01C*/

return

/*-------------------------------------------------------------------*/
/* This routine will do the actual call to IGGCSI00 and parse the
   resulting output.                                               */
```

```
/*-------------------------------------------------------------------*/
Call_IGGCSI00:
 field_offset = 1                 /* Offset into list of field kys*/
 fieldname_list = ''              /* Blank separated field key lst*/
 /* Create an array of the list of field keys passed to IGGCSI00   */
 do fkc = 1 to fieldkey_count
  CSIFLD1.fkc = substr(CSIFLD1,field_offset,8)
  /* Add to list of field names after removing any extra blanks    */
  fieldname_list = fieldname_list space(CSIFLD1.fkc,0)
  if fkc < fieldkey_count then     /* Increment to next field key  */
    field_offset = field_offset + 8
 end

 say ' '
 say ' Catalog Search Interface is being called with filter key:' key
 say ' and catalog field name(s):' fieldname_list

/* Set up loop for resume (if a resume is ncessary)                  */
 Do While RESUME = 'Y'

 /* Issue link to catalog generic filter interface                   */
 ADDRESS LINKPGM 'IGGCSI00  MODRSNRC  CSIFIELD  DWORK'
 CSIRC = RC                       /* Save return code from call   */
 csiusdln = c2d(substr(dwork,9,4))  /* Get amount of work area used  */

 If CSIRC <> 0 THEN               /* Test for non-zero return code*/
  Do
   csiretc = c2d(SUBSTR(MODRSNRC,4,1))
   csiresc = c2d(SUBSTR(MODRSNRC,3,1))
   csimodn = SUBSTR(MODRSNRC,1,2)
   if CSIRC = 4 then
    say ' The Catalog Search Interface, ended with an error in',
        'catalog module IGGOCL'csimodn 'with a' csiretc '-' csiresc
   else
    do
      say ' Processing ends with a ' csiretc '-'csiresc' in',
          ' catalog module IGGOCL'csimodn
      exit(csirc)
    end
  End

 RESUME = SUBSTR(CSIFIELD,150,1) /* GET RESUME FLAG FOR NEXT LOOP */

 rwaindex=15                      /*Start loc of Returned Work Area*/

/* Process data returned in work area                               */
 Do while rwaindex < CSIUSDLN       /* DO UNTIL ALL DATA IS PROCESSED*/
  If SUBSTR(DWORK,rwaindex+1,1) = '0' then /*Is this a catalog entry?*/
   Do                              /* process catalog entry        */
    CATNAME=SUBSTR(DWORK,rwaindex+2,44) /* Get catalog name from area*/
    /* Catalog entries are 50 bytes, index to next entry in work area*/
    rwaindex = rwaindex + 50
   End
  If rwaindex < CSIUSDLN THEN       /* IF STILL MORE DATA       @02A*/
   Do                               /* CONTINUE WITH NEXT ENTRY @02A*/
```

```
  rwaentry_len = 46               /* Offset for len of rwa entry  */

  /* This will point us to the MODID part if error in entry or the
     the DATA part if not error in entry                           */
  next_rwa_entry = rwaindex + rwaentry_len /* Find next entry      */
/*say 'next_rwa_entry' next_rwa_entry */
  /* Check the FLAG field to see if an error is returned for entry */
  CSIEFLAG = SUBSTR(DWORK,next_rwa_entry-46,1)
  CSIENTER =bitand(CSIEFLAG,"40"x)

  /*if CSIENTER is ON, then error is returned for the entry, issue
    a message with the error code and module and the entryname     */
  if CSIENTER = "40"x then
    do
      temperr = SUBSTR(DWORK,NEXT_RWA_ENTRY,4)
      tempname = SUBSTR(DWORK,NEXT_RWA_ENTRY-46,46)
      rwaindex = next_rwa_entry + 4
      CSIENAME = SUBSTR(tempname,3,44)
      CSIRETN = SUBSTR(temperr,1,4)
      csientrtc = c2d(SUBSTR(CSIRETN,4,1))
      csientrsn = c2d(SUBSTR(CSIRETN,3,1))
      csientmod = SUBSTR(CSIRETN,1,2)
  /*  IF SUBSTR(CSIENAME,1,10) = 'SYS1.VVDS.' THEN
       DO
       END */
      IF csientrtc = 50 & csientrsn = 5 THEN
       DO
       END
      ELSE
       say ' The Catalog Search Interface, detected an',
           'error' csientrtc'-'csientrsn 'in catalog module',
           'IGGOCL'csientmod 'for entry' CSIENAME
    end
  else     /* CSIENTER is not ON, so no error is found in entry */
    do
  /* Calculate location of next entry in the returned work area    */

  next_rwa_entry = next_rwa_entry + C2D(SUBSTR(DWORK,next_rwa_entry,2))
  /* Calculate length of data portion of this entry                */
  length_temp_worka = next_rwa_entry - rwaindex
/*say 'length_temp_worka' length_temp_worka */

  /* Copy the data for this return work area entry to simplify     */
  /* parsing the data                                              */
  temp_worka = SUBSTR(DWORK,rwaindex,length_temp_worka)
  dsname = SUBSTR(temp_worka,3,44) /* GET ENTRY NAME               */

 /* Assign entry type name                                         */
 ENTRY_TYPE = SUBSTR(temp_worka,2,1) /* Get entry type             */
 Select                             /* Assign name for this entry */
  WHEN ENTRY_TYPE = 'O' THEN DTYPE = 'CATALOG '
  WHEN ENTRY_TYPE = 'C' THEN DTYPE = 'CLUSTER '
  WHEN ENTRY_TYPE = 'D' THEN DTYPE = 'DATA    '
  WHEN ENTRY_TYPE = 'I' THEN DTYPE = 'INDEX   '
  WHEN ENTRY_TYPE = 'A' THEN DTYPE = 'NONVSAM '
```

```
   WHEN ENTRY_TYPE = 'H' THEN DTYPE = 'GDS     '
   WHEN ENTRY_TYPE = 'B' THEN DTYPE = 'GDG     '
   WHEN ENTRY_TYPE = 'R' THEN DTYPE = 'PATH    '
   WHEN ENTRY_TYPE = 'G' THEN DTYPE = 'AIX     '
   WHEN ENTRY_TYPE = 'X' THEN DTYPE = 'ALIAS   '
   WHEN ENTRY_TYPE = 'U' THEN DTYPE = 'UCAT    '
   WHEN ENTRY_TYPE = 'L' THEN DTYPE = 'ATLLIB  '                  2A*/
   WHEN ENTRY_TYPE = 'W' THEN DTYPE = 'ATLVOL  '                  2A*/
   Otherwise DTYPE = ''
  End
  If DTYPE = '' then iterate        /* Unknown entry type so skip it*/

 /* Have name and type, get volser info.  if we found a valid type  */
 /* for this catalog, now print the catalog name (first entry only).*/
  If CATNAME ^= CATNAMET THEN /* IF RESUME NAME MAY ALREADY   @02C  */
   Do                         /*    BE PRINTED                     */
    CATNAMET = CATNAME                                  /*@02C  */
   End                                                  /*@02C  */
  /* If all that was returned was a catalog entry then iterate     */
  If DTYPE = 'CATALOG ' then iterate
  rwaentry_len = 47
  /* What was the total length of data returned by csi?            */
  CSITOTLN= C2D(SUBSTR(temp_worka,rwaentry_len+0,2))

  len_offset   = 4                  /* Offset of field len in rwa   */

  /* Create an array of the length of each data field returned by   */
  /* IGGCSIOO for each key                                          */
  do fkc = 1 to fieldkey_count
   CSILENF.fkc = C2D(SUBSTR(temp_worka,rwaentry_len+len_offset,2))
   if fkc < fieldkey_count then     /* If within key count then     */
    len_offset = len_offset + 2     /* Increment to next field key  */
  end

  /* Find the start location for data in this entry                 */
  index2_rwadata = rwaentry_len+len_offset + 2

  /* Process returned data for each field key                       */
  do fkc = 1 to fieldkey_count-4
   Select
    When CSIFLD1.fkc = 'VSAMTYPE' then Call Parse_Generic
    Otherwise
     do
      exit(20)
     end
   end                          /* Do fkc = 1 to fieldkey_count*/
  end                           /* SELECT */
   rwaindex = next_rwa_entry    /* Point to next work area entry      */
  END                           /* ELSE LOOP - CSIENTER = "40"x       */
 END                           /* DO WHILE RWAINDEX < CSIUSDLN        */
END                            /* DO WHILE RESUME = 'Y'               */

if resume = 'y' &,             /* if we've tried this entry @01a     */
 PREVNAME = dsname THEN        /*  TWICE, WE'VE GOT TO QUIT      @01A*/
 DO                            /*                                @01A */
```

```
   say strip(dsname) 'Cannot be processed with the work area size ',
        'provided - you must increase the work area and retry'
     RETURN                    /*                                  @01A*/
  end                          /*                                  @01a*/
 PREVNAME = dsname             /*      SAVE FOR NEXT ITERATION     @01A*/
END                            /*      Call_IGGCSIOO routine           */
return

/*-------------------------------------------------------------------*/
/* Generic parsing routine                                           */
/*-------------------------------------------------------------------*/
Parse_Generic:
 numeric digits 12             /*  Set processing to 12 digits        */
 if header = 0 | ind = 1 then
   do
    header = 1
    ind = 0
    SAY ' '
    SAY '                                        Version 1.0'
    SAY ' '
    SAY '               DATA CA Size and INDEX CI Size Changes',
        'When Migrating to z/OS Version 1 Release 10'
    SAY ' '
    SAY 'CLUSTER NAME                                PRE-R10 ',
        'CA-SIZE IN  PRE-R10    CI-SIZE IN                  '
    SAY '                                            CA-SIZE ',
        '   R10      CI-SIZE       R10                      '
    SAY ' '
   end
 If CSILENF.fkc = 0 then       /* No data returned for this key      */
    return

 /* Get data returned by CSI for this entry */
 somefield = SUBSTR(temp_worka,index2_rwadata,CSILENF.fkc)

 if DTYPE = 'CLUSTER' | DTYPE = 'AIX' then
    clname = dsname

 /*If cluster name is all zeroes, then Catalog name is cluster name*/
 if SUBSTR(clname,1,1) = "00"x then
    clname = CATNAME

 if DTYPE = 'DATA' then           /* If Data component */
 do
  isksds = ' '
  isvrrds = ' '
  KIND = bitand(somefield,"8000"x)
  VIND = bitand(somefield,"0001"x)
  IF KIND = "8000"x | VIND = "0001"x then /* If KSDS or VRRDS */
   do
     x = 1
     IIND = bitand(somefield,"2000"x)
     IF IIND = "2000"x then
        imbed = 1
     else
```

```
        imbed = 0

     index2_rwadata = index2_rwadata + 2
     nextfield = SUBSTR(temp_worka,index2_rwadata,2)

     index2_rwadata = index2_rwadata + 12

     datacipca =  C2D(SUBSTR(temp_worka,index2_rwadata,2))

    end
     else                    /* not KSDS or VRRDS */
      do
          index2_rwadata = index2_rwadata + 62
          return
      end
 End                    /* DATA TYPE DATA         */

if DTYPE = 'INDEX' & x = 1 then        /*       If index component */
do
 /* RIND = bitand(somefield,"1000"x)
    IF RIND = "1000"x then
       replicate = 1
    else
       replicate = 0 */

    index2_rwadata = index2_rwadata + 4

    if imbed = 0 then
       cisize = c2d(SUBSTR(temp_worka,index2_rwadata,4))
    else
     do
       index2_rwadata = index2_rwadata + 2
       cisize = c2d(SUBSTR(temp_worka,index2_rwadata,4))
     end

 /* say 'R19' cisize */

    if nextfield = "0002"x then
     do
       index2_rwadata = index2_rwadata + 10
       ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
       index2_rwadata = index2_rwadata + 46
       keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
  /*   say 'ciperca' ciperca */
       datacipca = datacipca%2*3
   /*  say 'datacipca' datacipca */
    /* ciperca = ciperca%2
       ciperca = ciperca*3 */

   /*  say 'keylen' keylen */
       IF datacipca < 256 Then
           L_Pointer_Len = 1
       Else
           L_Pointer_Len = 2
       L_Best_Compressed_Key = Min(keylen,4)
```

```
 /* say 'L_Pointer_Len' L_Pointer_Len
    say 'L_Best_Compressed_Key' L_Best_Compressed_Key */

    L_Index_Entry_Len = L_Pointer_Len + 2,
    + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)
/*  say 'L_Index_Entry_Len' L_Index_Entry_Len */

    L_Nest_Level = 0
    DO L_Nest_Level = 1 BY 1,
            UNTIL(datacipca%(L_Nest_Level*L_Nest_Level)=0)
    END
    L_Nest_Level = L_Nest_Level - 1
/*  SAY 'L_Nest_Level' L_Nest_Level */

    Step1 = 31 + (L_Nest_Level * 2)
/*  SAY 'STEP1' Step1 */

    Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
 /* SAY 'STEP2' Step2 */

    Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
 /* SAY 'STEP3' Step3 */

    Step4 = Step3 % 512 * 512
 /* SAY 'STEP4' Step4 */

   If Step4 > 8192 Then
     Step4 = (Step4 + 2047)%2048*2048

  /* R1Ocisize =  Max(Step4,cisize) */
     R1Ocisize =  Step4
 /*  SAY 'R1OCISIZE' R1Ocisize */
    if cisize = R1Ocisize | R1Ocisize < cisize then
     do
       SAY  clname ' '2' TRKS    '     3' TRKS '
     end
    else
     do
      SAY  clname ' '2' TRKS    '     3' TRKS     'cisize' ',
            '     'R1Ocisize
     end
      count = count + 1
      if count = 50 then
       do
         count = 0
         ind = 1
       end
    x = 0
 end

 if nextfield = "0004"x then
  do
    index2_rwadata = index2_rwadata + 10
    ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
```

```
   index2_rwadata = index2_rwadata + 46
   keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
   datacipca = datacipca%4*5
 /* say 'datacipca' datacipca */

   IF datacipca < 256 Then
       L_Pointer_Len = 1
   Else
       L_Pointer_Len = 2
   L_Best_Compressed_Key = Min(keylen,4)


   L_Index_Entry_Len = L_Pointer_Len + 2,
   + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

   L_Nest_Level = 0
   DO L_Nest_Level = 1 BY 1,
           UNTIL(datacipca%(L_Nest_Level*L_Nest_Level)=0)
   END
   L_Nest_Level = L_Nest_Level - 1
/*  SAY 'L_NEST' L_NEST_LEVEL */
   Step1 = 31 + (L_Nest_Level * 2)
   Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
   Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
   Step4 = Step3 % 512 * 512
 /* SAY 'STEP4' STEP4 */
  If Step4 > 8192 Then
    Step4 = (Step4 + 2047)%2048*2048

    R1Ocisize =  Step4
 /*  SAY 'R1OCISIZE' R1Ocisize */
   if cisize = R1Ocisize | R1Ocisize < cisize THEN
    do
      SAY  clname ' '4' TRKS    '     5' TRKS '
    end
   else
    do
     SAY  clname ' '4' TRKS    '     5' TRKS     'cisize' ',
           '    'R1Ocisize
    end
/* if replicate = 1 then
     SAY  clname ' '4' TRKS    '     5' TRKS     '1024  ' ',
       '     '2048
  else
   do
     if cisize = 1536 then
      do
       SAY  clname ' '4' TRKS    '     5' TRKS     '
      end
     else
      do
        SAY  clname ' '4' TRKS    '     5' TRKS     '1024  ' ',
           '     '1536
      end
   end */
```

```
    count = count + 1
    if count = 50 then
     do
      count = 0
      ind = 1
     end
    x = 0
end

if nextfield = "0006"x then
 do
    index2_rwadata = index2_rwadata + 10
    ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    index2_rwadata = index2_rwadata + 46
    keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    datacipca = datacipca%6*7
 /* say 'datacipca' datacipca */
/*  ciperca = ciperca%6
    ciperca = ciperca*7 */

    IF datacipca < 256 Then
        L_Pointer_Len = 1
    Else
        L_Pointer_Len = 2
    L_Best_Compressed_Key = Min(keylen,4)


    L_Index_Entry_Len = L_Pointer_Len + 2,
    + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

    L_Nest_Level = 0
    DO L_Nest_Level = 1 BY 1,
            WHILE(datacipca%(L_Nest_Level*L_Nest_Level)^=0)
    END
    L_Nest_Level = L_Nest_Level - 1

    Step1 = 31 + (L_Nest_Level * 2)
    Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
    Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
    Step4 = Step3 % 512 * 512

   If Step4 > 8192 Then
    Step4 = (Step4 + 2047)%2048*2048

    R10cisize =  Step4
 /*  SAY 'R10CISIZE' R10cisize */
    if cisize = R10cisize | R10cisize < cisize THEN
     do
       SAY  clname ' '6' TRKS    '     7' TRKS '
     end
    else
     do
      SAY  clname ' '6' TRKS    '     7' TRKS     'cisize' ',
            '    'R10cisize
     end
```

```
/* if replicate = 1 then
        SAY  clname ' '4' TRKS    '     5' TRKS     '1536  ' ',
          '     '2560
   else
    do
     if cisize = 2048 then
      do
       SAY  clname ' '6' TRKS    '     7' TRKS     '
      end
     else
      do
       SAY  clname ' '6' TRKS    '     7' TRKS     '1536  ' ',
             '     '2048 */
       count = count + 1
       if count = 50 then
        do
          count = 0
          ind = 1
        end
   x = 0
 end

if nextfield = "0008"x then
  do
     index2_rwadata = index2_rwadata + 10
     ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     index2_rwadata = index2_rwadata + 46
     keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     datacipca = datacipca%8*9
  /* say 'datacipca' datacipca */

     IF datacipca < 256 Then
         L_Pointer_Len = 1
     Else
        L_Pointer_Len = 2
     L_Best_Compressed_Key = Min(keylen,4)


     L_Index_Entry_Len = L_Pointer_Len + 2,
     + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

     L_Nest_Level = 0
     DO L_Nest_Level = 1 BY 1,
            UNTIL(datacipca%(L_Nest_Level*L_Nest_Level)=0)
     END
     L_Nest_Level = L_Nest_Level - 1
 /*  SAY 'L_NEST' L_NEST_LEVEL */
     Step1 = 31 + (L_Nest_Level * 2)
     Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
     Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
     Step4 = Step3 % 512 * 512
 /*  SAY 'STEP4' STEP4 */
    If Step4 > 8192 Then
      Step4 = (Step4 + 2047)%2048*2048
```

```
     R1Ocisize =  Step4
 /*  SAY 'R1OCISIZE' R1Ocisize */
    if cisize = R1Ocisize | R1Ocisize < cisize THEN
     do
       SAY  clname ' '8' TRKS    '     9' TRKS '
     end
    else
     do
      SAY  clname ' '8' TRKS    '     9' TRKS      'cisize' ',
            '    'R1Ocisize
     end
    count = count + 1
    if count = 50 then
     do
      count = 0
      ind = 1
     end
    x = 0
end

 if nextfield = "OOOA"x then
  do
    index2_rwadata = index2_rwadata + 10
    ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    index2_rwadata = index2_rwadata + 46
    keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    datacipca = datacipca%10*15
 /* say 'datacipca' datacipca */
 /* ciperca = ciperca%10
    ciperca = ciperca*15 */

    IF datacipca < 256 Then
        L_Pointer_Len = 1
    Else
        L_Pointer_Len = 2
    L_Best_Compressed_Key = Min(keylen,4)


    L_Index_Entry_Len = L_Pointer_Len + 2,
    + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

    L_Nest_Level = 0
    DO L_Nest_Level = 1 BY 1,
           WHILE(datacipca%(L_Nest_Level*L_Nest_Level)^=0)
    END
    L_Nest_Level = L_Nest_Level - 1

    Step1 = 31 + (L_Nest_Level * 2)
    Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
    Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
    Step4 = Step3 % 512 * 512

   If Step4 > 8192 Then
     Step4 = (Step4 + 2047)%2048*2048
```

```
     R10cisize =  Step4
   /* SAY 'R10CISIZE' R10cisize */
     if cisize = R10cisize | R10cisize < cisize then
      do
        SAY  clname ''10' TRKS   '    15' TRKS '
      end
     else
      do
       SAY  clname ''10' TRKS   '    15' TRKS     'cisize' ',
             '    'R10cisize
      end
 /*  if imbed = 1 | replicate = 1 then
        do
          if cisize = 4608 then
            SAY  clname ''10' TRKS   '    15' TRKS     '
          else
            SAY  clname ''10' TRKS   '    15' TRKS     '3072  ' ',
               '     '4608
        end
      else
        do
          if cisize = 4096 then
           do
             SAY  clname ''10' TRKS   '     15' TRKS     '
           end
          else
           do
            SAY  clname ''10' TRKS   '     15' TRKS     '3072  ' ',
               '     '4096     */
            count = count + 1
         if count = 50 then
          do
           count = 0
           ind = 1
          end
       x = 0
     end

  if nextfield = "000B"x then
   do
     index2_rwadata = index2_rwadata + 10
     ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     index2_rwadata = index2_rwadata + 46
     keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     datacipca = datacipca%11*15
   /* say 'datacipca' datacipca */
  /*  ciperca = ciperca%11
     ciperca = ciperca*15 */

     IF datacipca < 256 Then
         L_Pointer_Len = 1
     Else
         L_Pointer_Len = 2
     L_Best_Compressed_Key = Min(keylen,4)
```

```
   L_Index_Entry_Len = L_Pointer_Len + 2,
   + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

   L_Nest_Level = 0
   DO L_Nest_Level = 1 BY 1,
           WHILE(datacipca%(L_Nest_Level*L_Nest_Level)^=0)
   END
   L_Nest_Level = L_Nest_Level - 1

   Step1 = 31 + (L_Nest_Level * 2)
   Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
   Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
   Step4 = Step3 % 512 * 512

  If Step4 > 8192 Then
    Step4 = (Step4 + 2047)%2048*2048

    R1Ocisize =  Step4
/*  SAY 'R1OCISIZE' R1Ocisize */
   if cisize = R1Ocisize | R1Ocisize < cisize THEN
    do
      SAY  clname ''11' TRKS   '    15' TRKS '
    end
   else
    do
     SAY  clname ''11' TRKS   '    15' TRKS     'cisize' ',
          '    'R1Ocisize
    end
/*  if imbed = 1 | replicate = 1 then
      do
        if cisize = 4608 then
          SAY  clname ''11' TRKS   '    15' TRKS     '
        else
          SAY  clname ''11' TRKS   '    15' TRKS     '3072  ' ',
             '     '4608
      end
    else
      do
       if cisize = 4096 then
        do
          SAY  clname ''11' TRKS   '    15' TRKS     '
        end
        else
         do
           SAY  clname ''11' TRKS   '    15' TRKS     '3072  ' ',
           '     '4096 */
           count = count + 1
       if count = 50 then
         do
           count = 0
           ind = 1
         end
      x = 0
   end
```

```
 if nextfield = "000C"x then
  do
    index2_rwadata = index2_rwadata + 10
    ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    index2_rwadata = index2_rwadata + 46
    keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
    datacipca = datacipca%12*15
 /* say 'datacipca' datacipca */
/*  ciperca = ciperca%12
    ciperca = ciperca*15 */

    IF datacipca < 256 Then
        L_Pointer_Len = 1
    Else
        L_Pointer_Len = 2
    L_Best_Compressed_Key = Min(keylen,4)


    L_Index_Entry_Len = L_Pointer_Len + 2,
    + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

    L_Nest_Level = 0
    DO L_Nest_Level = 1 BY 1,
           WHILE(datacipca%(L_Nest_Level*L_Nest_Level)^=0)
    END
    L_Nest_Level = L_Nest_Level - 1

    Step1 = 31 + (L_Nest_Level * 2)
    Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
    Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
    Step4 = Step3 % 512 * 512

   If Step4 > 8192 Then
    Step4 = (Step4 + 2047)%2048*2048

    R1Ocisize =  Step4
 /*  SAY 'R1OCISIZE' R1Ocisize */
    if cisize = R1Ocisize | R1Ocisize < cisize THEN
     do
       SAY  clname ''12' TRKS   '    15' TRKS '
     end
    else
     do
      SAY  clname ''12' TRKS   '    15' TRKS     'cisize' ',
            '    'R1Ocisize
     end
 /*  if imbed = 1 | replicate = 1 then
       do
         if cisize = 4608 then
           SAY  clname ''12' TRKS   '    15' TRKS     '
         else
           SAY  clname ''12' TRKS   '    15' TRKS     '3584  ' ',
              '     '4608
       end
```

```
    else
      do
       if cisize = 4096 then
        do
           SAY  clname ''12' TRKS   '    15' TRKS     '
        end
        else
         do
           SAY  clname ''12' TRKS   '    15' TRKS     '3584  ' ',
           '     '4096 */
           count = count + 1
       if count = 50 then
         do
           count = 0
           ind = 1
         end
      x = 0
   end

if nextfield = "000D"x then
 do
   index2_rwadata = index2_rwadata + 10
   ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
   index2_rwadata = index2_rwadata + 46
   keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
   datacipca = datacipca%13*15
/* say 'datacipca' datacipca */
/* ciperca = ciperca%13
   ciperca = ciperca*15 */

   IF datacipca < 256 Then
       L_Pointer_Len = 1
   Else
       L_Pointer_Len = 2
   L_Best_Compressed_Key = Min(keylen,4)


   L_Index_Entry_Len = L_Pointer_Len + 2,
   + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)

   L_Nest_Level = 0
   DO L_Nest_Level = 1 BY 1,
          WHILE(datacipca%(L_Nest_Level*L_Nest_Level)^=0)
   END
   L_Nest_Level = L_Nest_Level - 1

   Step1 = 31 + (L_Nest_Level * 2)
   Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
   Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
   Step4 = Step3 % 512 * 512

  If Step4 > 8192 Then
    Step4 = (Step4 + 2047)%2048*2048

    R10cisize =  Step4
```

```
 /* SAY 'R1OCISIZE' R1Ocisize */
   if cisize = R1Ocisize | R1Ocisize < cisize THEN
    do
      SAY  clname ''13' TRKS   '    15' TRKS '
    end
   else
    do
     SAY  clname ''13' TRKS   '    15' TRKS     'cisize' ',
           '    'R1Ocisize
    end
 /*  if imbed = 1 | replicate = 1 then
     do
       if cisize = 4608 then
         SAY  clname ''13' TRKS   '    15' TRKS     '
       else
         SAY  clname ''13' TRKS   '    15' TRKS     '3584  ' ',
            '     '4608
     end
    else
      do
       if cisize = 4096 then
        do
           SAY  clname ''13' TRKS   '    15' TRKS     '
        end
        else
         do
           SAY  clname ''13' TRKS   '    15' TRKS     '3584  ' ',
           '     '4096 */
           count = count + 1
        if count = 50 then
         do
           count = 0
           ind = 1
         end
       x = 0
   end

if nextfield = "000E"x then
  do
     index2_rwadata = index2_rwadata + 10
     ciperca = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     index2_rwadata = index2_rwadata + 46
     keylen = C2D(SUBSTR(temp_worka,index2_rwadata,2))
     datacipca = datacipca%14*15
  /* say 'datacipca' datacipca */

     IF datacipca < 256 Then
         L_Pointer_Len = 1
     Else
         L_Pointer_Len = 2
     L_Best_Compressed_Key = Min(keylen,4)


     L_Index_Entry_Len = L_Pointer_Len + 2,
     + Max(keylen%3 + Min(1,keylen//3),L_Best_Compressed_Key)
```

```
        L_Nest_Level = 0
        DO L_Nest_Level = 1 BY 1,
                UNTIL(datacipca%(L_Nest_Level*L_Nest_Level)=0)
        END
        L_Nest_Level = L_Nest_Level - 1
     /* SAY 'L_NEST' L_NEST_LEVEL */
        Step1 = 31 + (L_Nest_Level * 2)
        Step2 = Step1 + (datacipca-1) * L_Index_Entry_Len
        Step3 = Step2 + (L_Pointer_Len + 2 + keylen) + 512 - 1
        Step4 = Step3 % 512 * 512
     /* SAY 'STEP4' STEP4 */
       If Step4 > 8192 Then
         Step4 = (Step4 + 2047)%2048*2048

         R1Ocisize =  Step4
        if cisize = R1Ocisize | R1Ocisize < cisize THEN
         do
           SAY  clname ''14' TRKS   '    15' TRKS '
         end
        else
         do
          SAY  clname ''14' TRKS   '    15' TRKS     'cisize' ',
                '    'R1Ocisize
         end
        count = count + 1
        if count = 50 then
         do
          count = 0
          ind = 1
         end
        x = 0
    end

     x = 0
     imbed = 0
     replicate = 0
end
return
/*-------------------------------------------------------------------*/
/* Write Help Text for this exec                                     */
/*-------------------------------------------------------------------*/
Write_Help:
say '                                                                 '
say ' Description: This rexx exec is used to call the catalog search'
say '              interface.                                         '
say ' Input: filter key and catalog name (optional)                   '
say '                                                                 '
say ' Output: Data returned from the catalog search interface for     '
say '         which a parsing routine exists.                         '
say '                                                                 '
say ' Variables:                                                      '
say ' CATALOG() parm should have a catalog name                       '
say '                                                                 '
say ' Output: Data returned from the catalog search interface for     '
```

```
say '         which a parsing routine exists.                          '
say '                                                                  '
Write_Return_Codes:
say ' Condition Codes:                                                 '
say '  None                                                            '
say '                                                                  '
if exitrc > 0 then
 do
  say 'Ending with a return code of 'exitrc
  say '  '
  exit(exitrc)
 end

exit
```

B

HMC API example

This appendix contains an example REXX exec that exploits the HMC API, as discussed in Chapter 27, “Server Time Protocol alerts” on page 561.

B.1 HMC console API REXX example

Example contains sample REXX code for the HMC.

Example B-1 REXX exec using the HMC API

```
/*********************************************************************/
/* Rexx command file used to illustrate the use of the Hardware */
/* Management Console APIs. This sample will allow the user to see */
/* the objects that can be managed from the Hardware Management */
/* Console, as well as perform tasks against these objects. */
/*********************************************************************/
trace 'o';
/*********************************************************************/
/* Number of seconds that this Rexx sample will wait for API calls */
/* to complete. This may need to be changed for remote networks */
/* that require more time to return the responses. */
/*********************************************************************/
api_timeout_secs = 30;
api_timeout = api_timeout_secs * 1000;
/*********************************************************************/
/* Parse the provided arguments. No arguments are required, since */
/* we will prompt the user for them. However, they can be passed */
/* as follows: */
/* Argument #1 - target HMC's hostname or internet address */
/* Argument #2 - target HMC's SNMP community name for API request*/
/*********************************************************************/
parse arg INITBLK.TARGET INITBLK.COMMUNITY .;
'@echo off'
error = 0;
/*********************************************************************/
/* Load the OS/2 Rexx Utility functions DLL. */
/*********************************************************************/
if RxFuncQuery('SysLoadFuncs') then do
if rxfuncadd( 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs' ) then do
say 'Error trying to add OS/2 Rexx utility functions.';
error = 98;
end /* Do */
end /* Do */
/*********************************************************************/
/* Load the Hwmca Rexx API interface function DLL. */
/*********************************************************************/
if RxFuncQuery('RxHwmcaLoadFuncs') then do
if rxfuncadd( 'RxHwmcaLoadFuncs', 'ACTZSNMP', 'RxHwmcaLoadFuncs' ) then do
say 'Error trying to add the Hardware Management Console API Rexx functions.';
error = 99;
end /* Do */
end /* Do */
if error == 0 then do
call SysLoadFuncs /* Load Rexx utility functions */
call RxHwmcaLoadFuncs; /* Load HMC API functions */
call RxHwmcaDefineVars; /* Define HMC API variables */
```

```
/*****************************************************************/
/* Prompt the user for the HMC hostname or internet address. */
/*****************************************************************/
if INITBLK.TARGET = '' then do
say ' '; say 'Please enter target Hardware Management Console hostname or
internet address.';
end /* Do */
/*****************************************************************/
/* Prompt the user for the HMC community name to use. */
/*****************************************************************/
if INITBLK.COMMUNITY = '' then do
say ' '; say 'Please enter community name for target Hardware Management
Console.';
parse pull INITBLK.COMMUNITY .;
end /* Do */
/*****************************************************************/
/* This sample uses the same Initialization block, INITBLK, for */
/* all RxHwmca API calls. */
/*****************************************************************/
INITBLK.EVENTMASK = HWMCA_EVENT_COMMAND_RESPONSE
; /*****************************************************************/
/* Initialize ourselves with the HMC and tell it that we are only*/
/* interested in command response events. */
/*****************************************************************/
rc = RxHwmcaInitialize('INITBLK.',api_timeout);
if rc == HWMCA_DE_NO_ERROR then do
/*****************************************************************/
/* Get the size of the screen so we know how much room we */
/* have for outputting information. */
/*****************************************************************/
parse value SysTextScreenSize() with screen_rows screen_cols;
/*****************************************************************/
/* We are now successfully initialized with the HMC. First, */
/* lets get the name of the HMC. */
/*****************************************************************/
call get_hmc_name; nest = 0; bailout = 0;
/*****************************************************************/
/* Now we need to request the list of groups and present this */
/* to the user. */
/*****************************************************************/
if result <> ''then call show_contents HWMCA_CONSOLE_ID 'Groups';
/*****************************************************************/
/* Terminate our session with the HMC, so that it does not */
/* try and send us any more events. */
/*****************************************************************/
rc = RxHwmcaTerminate('INITBLK.',api_timeout);
call SysCls;
end /* do */
else do say 'Error' rc 'on RxHwmcaInitialize call.';
end /* do */
end /* Do */
exit
/*****************************************************************/
```

```
/* Subroutine: get_hmc_name */ /* */ /* This subroutine will request name
attribute for the HMC. */
/*********************************************************************/
get_hmc_name:
hmc_name = get_name(HWMCA_CONSOLE_ID);
return hmc_name;
```

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 701. Note that some of the documents referenced here may be available in softcopy only.

- *z/OS Distributed File Service zSeries File System Implementation z/OS V1R11,* SG24-6580
- *UNIX System Services z/OS Version 1 Release 7 Implementation*, SG24-7035
- *STP Implementation Guide*, SG24-7281

Other publications

These publications are also relevant as further information sources:

- *ServerPac: Installing Your Order* - (no order number; custom-built to your order)
- *z/OS Problem Management*, G325-2564
- *z/OS Migration*, GA22-7499
- *z/OS Introduction and Release Guide*, GA22-7502
- *z/OS License Program Specifications*, GA22-7503
- *z/OS Planning for Installation*, GA22-7504
- *z/OS Hardware Configuration Definition Planning*, GA22-7525
- *z/OS JES3 Diagnosis Reference*, GA22-7548
- *z/OS MVS Data Areas, Volume 2 (DCCB-ITZYRETC)*, GA22-7582
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589
- *z/OS UNIX System Services Planning*, GA22-7800
- *IBM WebSphere Application Server OEM Edition for z/OS Configuration Guide,* GA32-0631
- *z/OS Program Directory*, GI10-0670
- *Program Directory for z/OS Management Facility,* GI11-2886
- *z/OS JES2 Commands*, SA22-7526
- *z/OS JES2 Initialization and Tuning Reference*, SA22-7533
- *z/OS JES2 Installation Exits*, SA22-7534
- *z/OS JES2 Macros*, SA22-7536
- *z/OS JES2 Messages*, SA22-7537
- *z/OS JES3 Messages*, SA22-7552

- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS MVS Installation Exits*, SA22-7593
- *z/OS MVS JCL Reference*, SA22-7597
- *z/OS MVS Planning: Global Resource Serialization*, SA22-7600
- *z/OS MVS Planning: Operation*, SA22-7601
- *z/OS MVS Programming: MVS Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*, SA22-7609
- *z/OS MVS Authorized Assembler Service Reference EDT-IXG*, SA22-7610
- *z/OS MVS Programming: Callable Services for High Level Languages*, SA22-7613
- *z/OS MVS Setting Up a Sysplex*, SA22-7625
- *z/OS MVS System Commands,* SA22-7627
- *z/OS V1R11 MVS System Messages Volume 9*, SA22-7639
- *z/OS MVS Using the Subsystem Interface*, SA22-7642
- *z/OS SDSF Operation and Customization*, SA22-7670
- *z/OS Security Server RACROUTE Macro Reference*, SA22-7692
- *ServerPac: Using the Installation Dialog*, SA22-7815
- *IBM Health Checker for z/OS User's Guide*, SA22-7994
- *z/OS JES Application Programming*, SA23-2240
- *z/OS Common Information User's Guide*, SC33-7998
- *IBM z/OS Management Facility User's Guide,* SA38-0652
- *System z Application Programming Interfaces*, SB10-7030
- *z/OS Distributed File Service zSeries File System Administration*, SC24-5989
- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- *z/OS DFSMS Storage Administration Reference (for DFSMSdfp, DFSMSdss, DFSMShsm)*, SC26-7402
- *z/OS DFSMSdfp Utilities*, SC26-7414
- *z/OS Common Information Model User's Guide*, SC33-7998
- *z/OS DFSMShsm Managing Your Own Data*, SC35-0420
- *z/OS DFSMShsm Storage Administration*, SC35-0421
- *z/OS DFSMSdfp Advanced Services*, SC35-0428
- *z/OS Interactive System Productivity Facility (ISPF) Planning and Customizing*, SC34-4814
- *z/OS Interactive System Productivity Facility (ISPF) Messages and Codes*, SC34-4815
- *z/OS Interactive System Productivity Facility (ISPF) Services Guide*, SC34-4819
- *z/OS Interactive System Productivity Facility (ISPF) Edit and Edit Macros*, SC34-4820
- *z/OS Interactive System Productivity Facility (ISPF) Dialog Developer's and Reference*, SC34-4821
- *z/OS Interactive System Productivity Facility (ISPF) User's Guide Volume I*, SC34-4822
- *z/OS Interactive System Productivity Facility (ISPF) User's Guide Volume II*, SC34-4823

Online resources

These Web sites are also relevant as further information sources:

- The following site provides the latest information about end-of-service dates for z/OS releases.

  http://www.ibm.com/servers/eserver/zseries/zos/support/zos_eos_dates.html

- 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/

- A new tool is now available to find data sets that have a larger CA size:

  ftp.software.ibm.com

- Support Element Operations Guide for the System z10 machine:

  http://www.ibm.com/servers/resourcelink

- Find migration checks by using the functional PSP bucket HCHECKER.

  http://www14.software.ibm.com/webapp/set2/psp/srchBrocker

- View all IBM Health Checker for z/OS checks:

  http://www.ibm.com/systems/z/os/zos/hchecker/check_table.html

- IBM Ported Tools for z/OS:

  http://www-03.ibm.com/servers/eserver/zseries/zos/unix/ported/

- IBM XML Toolkit for z/OS, V1R10 continues to provide enhanced support for the XML Parser, C++ Edition and the XSLT Processor, C++ Edition:

  http://www.ibm.com/zseries/software/xml/

- Information about IBM 31-bit SDK for z/OS, Java Technology Edition Version 6, and IBM 64-bit SDK for z/OS, Java Technology Edition Version 6, is available at the zSeries Java Web site:

  http://www.ibm.com/servers/eserver/zseries/software/java/

- The most current information about 64-bit SDK for z/OS, Java 2 Technology Edition Version 5 or 31-bit SDK for z/OS, Java 2 Technology Edition Version 5 is available at the zSeries Java Web site:

  http://www.ibm.com/servers/eserver/zseries/software/java/

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 publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Redbooks

z/OS Version 1 Release 11 Implementation

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

z/OS Version 1 Release 11 Implementation

EAV volumes, HCD/HCM, HyperSwap, JES2, HiperDispatch, PFA

JES3, SDSF, Service Aids, GRS, LE, System REXX, STP

z/OS UNIX, zFS, ISPF, RRS, CIM, XCF, SMF, DCM

This IBM Redbooks publication positions the new z/OS Version 1 Release 11 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.

z/OS is a highly secure, scalable, high-performance enterprise operating system on which to build and deploy Internet- and Java-enabled applications, providing a comprehensive and diverse application execution environment.

This book describes the following new and changed functions:

- IBM z/OS Management Facility
- Allocation enhancements in z/OS V1R11
- BCPii function enhancements in z/OS V1R11
- JES2 and JES3 enhancements
- zFS file sharing enhancements
- Extended access volume enhancements
- Choosing whether to run zAAP work on zIIP processors
- System REXX enhancements in V1R11
- RRS global panel options
- Service aids enhancements in V1R11
- GRS ENQ contention notification enhancements and analysis for GRS latches
- Basic HyperSwap support enhancement
- Message Flood Automation enhancements
- Program Management new Binder IEWPARMS
- Predictive Failure Analysis (PFA)
- SMF enhancements in V1R11
- System Logger enhancements
- XCF/XES enhancements in V1R11
- AutoIPL support
- Displaying PDSE caching statistics
- ISPF enhancements
- IBM Health Checker for z/OS enhancements

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

SG24-7729-00 ISBN 073843387X