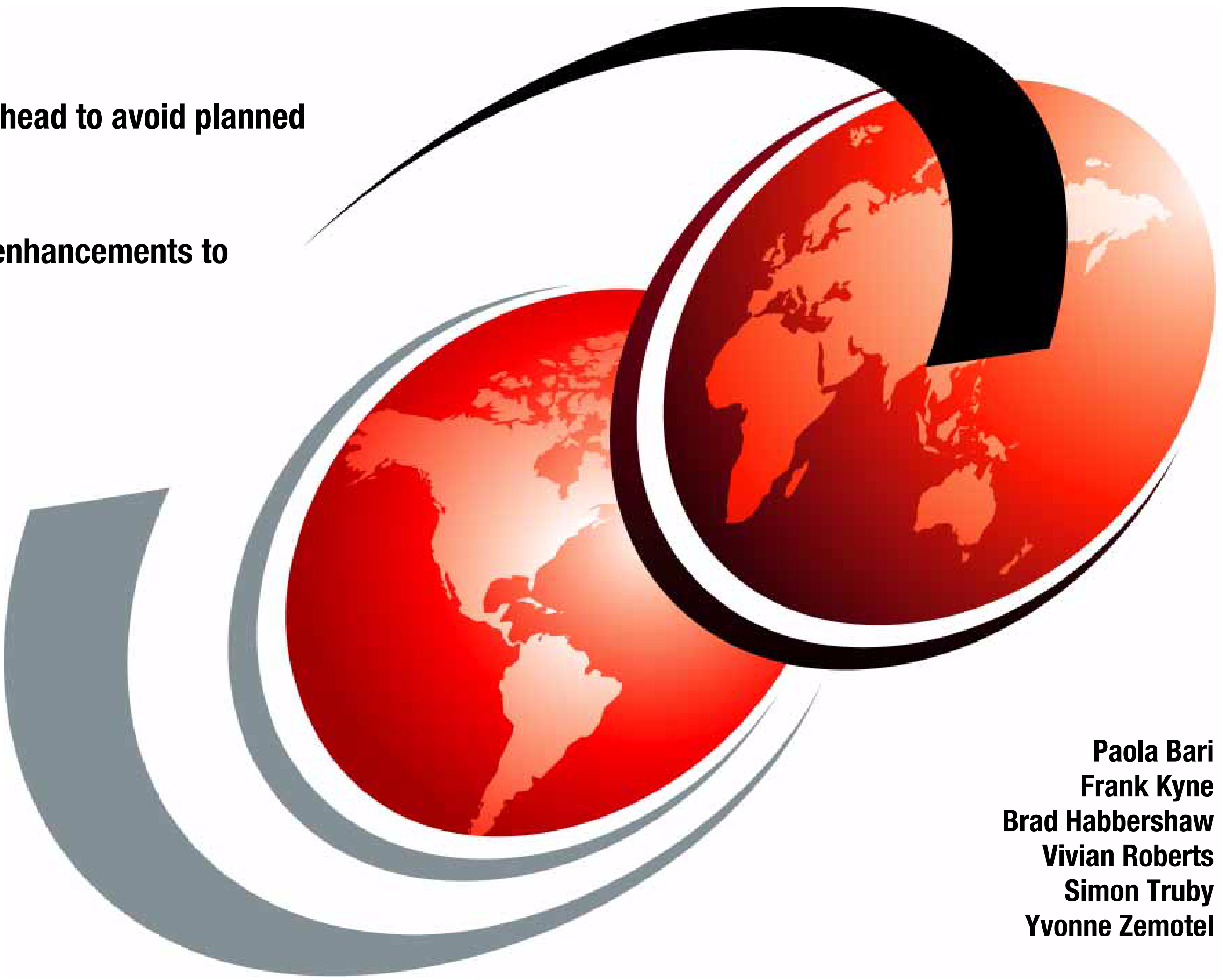
IBM

z/OS Planned Outage Avoidance Checklist

Making dynamic changes

Planning ahead to avoid planned outages

Recovery enhancements to avoid IPLs



Paola Bari
Frank Kyne
Brad Habbershaw
Vivian Roberts
Simon Truby
Yvonne Zemotel

ibm.com/redbooks

Redbooks

International Technical Support Organization

z/OS Planned Outage Avoidance Checklist

August 2006

SG24-7328-00

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

First Edition (August 2006)

This edition applies to Version 1, Release 7, Modification 0 of z/OS (product number 5647-A01).

Contents

Notices

Trademarks

Preface

One of the offshoots of the increasingly competitive global business environment is growing pressure to reduce the number of outages, both of a planned and unplanned nature. In the area of planned outages, IBM® has been steadily reducing the number of changes that require a system restart to implement. However, because of the detailed nature of these changes, many of them are not widely publicized and therefore not widely known. This IBM Redbook attempts to redress this situation by listing the changes that have been made to the operating system since OS/390® 1.3.

The team that wrote this redbook

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

Paola Bari is an Advisory Programmer at the International Technical Support Organization, Poughkeepsie Center. She has 23 years of experience as a Systems Programmer with MVS, OS/390, z/OS® and Parallel Sysplex®, including several years of experience with WebSphere® MQ and WebSphere Application Server.

Frank Kyne is a Certified IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of Parallel Sysplex. Frank is also responsible for the GDPS® product documentation. Before joining the ITSO seven years ago, Frank worked in IBM Global Services in Ireland as an MVS Systems Programmer.

Brad Habbershaw is an IT Specialist for IBM Global Services, Securities Industries Services, in IBM Canada. He has over 20 years of experience in mainframe system programming and technical support. His areas of expertise include JES3 and RACF®. He has contributed to previous IBM Redbooks® on Parallel Sysplex.

Vivian Roberts is a Senior IT Specialist with the IBM Support Center in Sydney, Australia. She has been working in the IT industry for over 25 years and has spent 21 years providing defect and non-defect support on all z/OS products.

Simon Truby is a Chief Technical Specialist with HSBC Plc in the United Kingdom. He has 17 years of experience in mainframe system programming and technical support. His areas of expertise include installation and support of many z/OS products, including UNIX® System Services (USS) and IMS.

Yvonne Zemotel is a Senior IT Specialist and Parmlib specialist with State Street Bank USA. She has 30 years of experience in mainframe application and system programming and technical support. Her areas of expertise include installation and support of many z/OS and OEM products.

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

Rich Conway
Bob Haimowitz
Paul Rogers
International Technical Support Organization, Poughkeepsie Center

Riaz Ahmad
Dave Anderson
Richard Aprile
Cy Atkinson
Laura Blodgett
Bette Brody
Barbara Bryant
Jim Caffrey
Bob Chambers
Bryan Childs
Pat Choi
Alfred Christensen
Noshir Dhondy
Scott Fagen
Richard Fine
Glenn Garrison
Frank Goberish
Evan Haruta
Gus Kassimis
Kathy Koch
Mark Langton
Nick Matsakis
Terri Menendez
Geoff Miller
Jim Mulder
Mark Nelson
Mike Phillips
David Raften
Peter Relson
Sam Reynolds
Dale Riedy
Michael Scott
Janine Sesa
Neil Shah
Ralph Sharpe
Maida Snapper
Ken Trowell
Jatinderpal Virdi
Marna Walle
Tom Wasik
David Whitney
William Yurkovic
Steve Zehner
IBM USA

Friedrich Beichter
Paul-Robert Hering
IBM Germany

Tom Bettle
William Reynolds
Sue Yarker
HSBC Plc.

Frank Byrne
David Clitherow

Roger Fowler
IBM UK

Pierre Cassier
IBM France

John Cross
Brian Gilman
IBM Canada

John Papp
State Street Bank

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about thisor other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

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

1

Introduction

This chapter introduces this Redbook and describes the layout of the remainder of the document.

1.1 Background

The modern business world is an extremely pressurized environment, and the way companies and individuals transact business now is nearly unrecognizable from just a decade ago. For more and more shoppers, for example, the first place they visit when considering a purchase is the Internet. And although in a store, shoppers will often stand in a queue for minutes while waiting to be served—on the Internet they can and often do take their business elsewhere if they are not served within seconds. Even beyond the Internet shopping world, there is relentless pressure on companies to be more efficient and nimble. And a vital component in this efficiency is the use (and therefore, the availability) of IT systems.

There is also an unprecedented move to globalization and consolidation. Previously companies were often large players in their home markets, or were large international corporations with many data centers around the world, each supporting the local market. Now we are seeing huge mergers, with mega-companies being created that operate internationally but have a small number of data centers, each supporting users all over the globe. On top of all this is the emergence of processors such as System z9®, a server that supports up to 60 LPARs.

As a result of such changes, IT departments are under constant demand to provide near-continuous application availability. For systems that serve Internet transactions or have users distributed around the globe, there is probably never a time when no one wants to use the system. And there is really no such thing as a “planned outage” at this point; an outage is only “planned” if all system users are aware of it in advance—and in the Internet world, IT departments cannot inform all their potential users in advance. So every minute of downtime, planned or otherwise, means lost revenue for their companies.

For these reasons, IT departments are looking more closely at every planned outage, trying to find ways to eliminate that downtime. IBM, in support of this strategy, has been making constant changes and improvements to its mainframe operating systems and subsystems over the years, steadily decreasing the number of situations where an IPL is required.

Such changes or improvements, however, are not typically mentioned in announcement letters or similar material. Every new release of the operating system contains so many new features, that there is no room to provide information on every “little” enhancement.

This Redbook addresses the problem by providing a comprehensive list of all enhancements to the operating system and its major components since around the OS/390 1.3 time frame that play a role in helping to avoid planned IPLs.

Note: Covering subsystem information systems is beyond the scope of this paper. Future updates may be expanded to include such information.

1.2 Role of Parallel Sysplex exploitation

Before proceeding, two points need clarification:

- It will probably never be possible to completely eliminate the need for planned system outages. For example, replacing a server or moving to a new release of the operating system will always require a system to be brought down for a short period of time.
- It is not *system* availability that users are concerned about—rather, it is *application* availability.

Most users may neither know nor care what system or platform their applications run on. Applications should be like electric service: when a switch is turned on, everyone expects light to appear. Users of electricity do not know nor necessarily care how the power is generated. Equally, system users expect to be able to use applications at all times, without having to worry about an alternative process if the normal logon process does not work. And systems programmers, while not having the capability to eliminate *all* IPLs, have the ability now to avoid having planned system outages cause outages to applications.

Given that it will always be necessary to have some number of system outages, how do you address the user requirement for continuous application availability? You can satisfy this requirement by having at least two (failure-isolated) copies of the application.

In an appropriately planned and configured Parallel Sysplex, it should rarely be necessary to have a planned system outage that will impact your application availability. The only planned events that could result in a complete lack of application availability should be a sysplex IPL (which should be rare or non-existent), or a change to the DASD that the system or application resides on (and even this can be avoided through the use of the HyperSwap™ capability in GDPS/PPRC).

The goal should be to have at least one sysplex member up and running at all times—this potentially enables you to maintain constant application availability and also provides a platform that can be used to resolve any problems that may be preventing another system from successfully IPLing.

If you are able to run copies of your critical applications on all members of the sysplex, as you could in a data-sharing Parallel Sysplex, it should be possible to maintain the availability of those applications across planned outages. Parallel Sysplex technology can help you dynamically balance your workload across the systems, automatically re-routing incoming work requests to whichever server instances are available, and steering them away from the system that is about to be shut down. However, this requires that every system have the ability to read and update all data: data sharing, in other words.

This Redbook does not specifically address Parallel Sysplex; there are other IBM Redbooks and documentation that cover that topic. However, it is important to stress that data sharing and dynamic workload balancing should be considered a prerequisite to delivering truly continuous application availability.

There is another consideration regarding Parallel Sysplex and data sharing, which is that many customers have conflicting IPL requirements:

- The desire to IPL more frequently in order to keep service levels up to date, and minimize the chance of rediscovering a known and fixed problem
- The desire to spread the IPLs as far apart as possible, to minimize the impact on end users when the system is unavailable

This dilemma is illustrated in Figure 1-1.

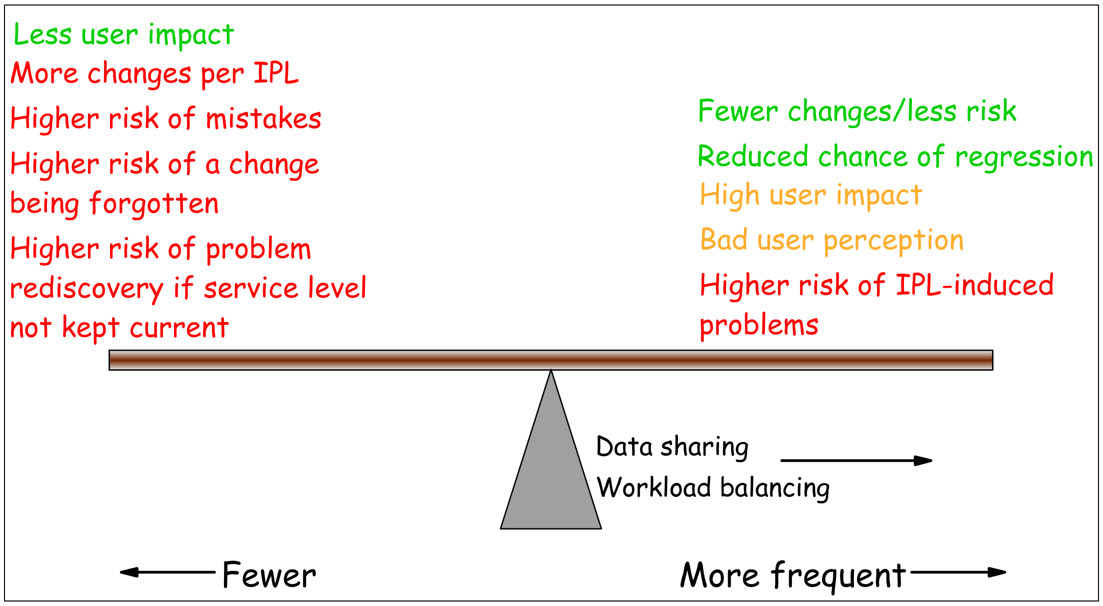

Less user impact
More changes per IPL
Higher risk of mistakes
Higher risk of a change being forgotten
Higher risk of problem rediscovery if service level not kept current
Fewer changes/less risk
Reduced chance of regression
High user impact
Bad user perception
Higher risk of IPL-induced problems
Data sharing
Workload balancing
Fewer
More frequent

Figure 1-1 IPL frequency considerations and data sharing

The advantage and disadvantages of IPLing frequently are listed on the right side of the fulcrum (the advantages are shown in green, and the disadvantages are shown in red). The advantages and disadvantages of IPLing very infrequently are listed on the left side of the fulcrum (again, the advantages are green and the disadvantages are red). The items in orange mitigate against frequent IPLs *unless* you are doing data sharing.

This graphic illustrates that if you are exploiting data sharing and workload balancing, then you have the *option* to IPL more frequently than might be acceptable otherwise. This is not saying that you necessarily *should* IPL frequently, simply that we have removed the orange items as reasons not to do so. This topic is discussed in more detail in 11.2.3, “IPL frequency” on page 127.

1.3 Layout of this Redbook

This Redbook is designed primarily for systems programmers, so the chapter contents are based on the typical division of responsibilities that is seen in most installations. For example, one chapter covers JES2, another chapter addresses hardware, while yet another deals with the network, and so on.

Also, in order to keep the paper to a manageable size and avoid having the information become obsolete, detailed implementation information is not provided. Rather, we raise awareness of what features, functions, and capabilities are available, and point you to the detailed documentation that will help you exploit whatever functions are appropriate to your installation.

Using the checklist tables

Each chapter provides a checklist, in a table format, listing the features that are discussed in that chapter. The table contains a brief description of the feature, an indicator of what release of z/OS introduced that feature[1], and a column that you can use to indicate for yourself

[1] At the time of writing, the oldest supported z/OS release is 1.4. Therefore, if the feature was introduced in z/OS 1.4 or a *previous* release of z/OS or OS/390, the release listed in the table will be 1.4.

whether you exploit that feature or not. If you find, having reviewed the table, that you are not exploiting *any* of these features, it could be an area that you might like to concentrate on.

Note: Even though the primary subject matter of this book is planned outage avoidance, we have also included references to a small number of other enhancements that we feel will be of interest to readers interested in this topic.

An example is a tip on reducing IPL times; while this does not help you avoid an IPL, it does address availability by helping you get the system up and running faster. Another example is a tip on reducing dump times: this may enable you to gather all required information on the first occurrence of a problem while at the same time reducing the impact the dump process has on service levels.

As noted, in this edition of the Redbook we do not address the subsystems. But this does not infer that the subsystems are unimportant or that there have not been enhancements in those areas. We understand that for many customers, having to restart one of the major subsystems (DB2®, for example) is tantamount to doing an IPL. The IBM development organizations are aware of this as well and have been addressing this by reducing the number of changes that require a subsystem restart. However, in order to make the paper available in a timely manner, we decided to limit the scope in this edition. If you find this document beneficial, we may update it in the future, possibly adding some or all of the subsystems at that time; your feedback is solicited:

redbook@us.ibm.com

Use your input to also let us know if there are other relevant features or functions that you feel are useful but are not covered here. Over time, we would like this document to become the portal that systems programmers use to find what planned outage avoidance features are available. The more information we can include, based in part on your feedback, the more everyone will benefit.

2

z/OS

This chapter describes changes to reduce the need for planned outages made in the following operating system components:

- Base Control Program (BCP - the MVS component of z/OS)
- Global Resource Serialization (GRS)

Note: In this and subsequent chapters, we discuss making dynamic changes to your systems. But dynamic change can have both desirable and undesirable consequences:

- Changing something dynamically can avoid an outage (desirable)
- Changing something dynamically introduces the risk that the change will be regressed when you perform your next IPL (undesirable)

In some cases, you have two options for introducing a change to the system. An example is making dynamic APF changes; there are two ways to dynamically implement a change to the APF list:

- You can use the SETPROG command and specify the data set name and all other required information on the command line.
- You can update the PROGxx member in Parmlib and then implement the change by issuing a SET PROG=xx command, pointing at the updated PROGxx member.

An *advantage* of the second option (assuming that you updated the PROGxx member used during an IPL) is that you already completed the change in Parmlib, so it would be in place at the next IPL and would not be regressed. Another advantage is that by activating from the PROGxx member you are ensuring that the syntax is correct, rather than doing an IPL and finding at that time, when the system is only half up, that you have a problem. The *disadvantage* is that you may inadvertently also pick up other changes that have been applied to the PROGxx member you use.

Which option you select is related to your confidence in your change management process. The first option may be suitable for some clients, and others may prefer the second. In any case, this is a change management decision that should be deliberately made, and then adhered to by all your support personnel.

2.1 BCP enhancements checklist

Table 2-1 contains a checklist of enhancements to the Base Control Program component of the operating system.

Table 2-1 Checklist of outage avoidance items for BCP component of z/OS and OS/390

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
Console Message Backup	1.4	Procedures and commands exist to allow recovery from a Console Message Backup situation. See 2.1.1, “Console message backup” on page 10 for more detail.	
Consoles Enhancements	1.4.2	Consoles Enhancements was introduced as a feature of z/OS 1.4 and subsequently integrated into z/OS 1.5. See 2.1.2, “Consoles Enhancements” on page 11 for more detail.	
CMDS Command	1.4	The CMDS command is used to display executing and waiting MVS commands, to delete commands that are waiting for execution, or to cancel commands that are executing. See 2.1.3, “CMDS command” on page 12 for more detail.	
Linkage Indexes (LXs)	ALL, enhanced in 1.6	A limited number of LXs are available for the system to use. When all LXs are assigned to existing address spaces, the system is unable to start a new address space. See 2.1.4, “Linkage indexes” on page 12 for more detail.	
Non-reusable address spaces	N/A	A limited number of ASIDs are available for the system to use. When all ASIDs are assigned to existing address spaces, the system is unable to start a new address space. See 2.1.5, “Non-reusable address spaces” on page 14 for more detail.	
PAGEADD Command	1.4	This command allows for the addition of local page data sets without requiring an IPL. See 2.1.6, “PAGEADD command” on page 17 for more detail.	
PAGEDEL Command	1.4	This command allows for the removal or replacement of local page data sets without requiring an IPL. See 2.1.7, “PAGEDEL command” on page 18 for more detail.	
Dynamically updating SVCs	1.4	After the IPL, the SVC table can be dynamically modified by authorized users with the SVCUPDTE macro. See 2.1.8, “SVCUPDTE macro” on page 18 for more detail.	
Reduced IPL times with large DASD configurations	1.4	APAR OA07335 introduced increased parallelism during IPL processing. Refer to 2.1.9, “Increased I/O parallelism at IPL time” on page 20 for more information.	

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
Eliminate SMF Type 19 recording	1.4	Turning off recording of SMF Type 19 records can speed up IPL processing. Refer to 2.1.10, “Reducing IPL times” on page 20 for more information.	
Dynamic APF List	1.4	It is possible to dynamically add and remove libraries from the APF list. Also, use of dynamic APF allows you to have more authorized libraries. Refer to 2.1.11, “Dynamic APF list” on page 20 for more information.	
Dynamic LNKLST	1.4	It is possible to dynamically add and remove libraries from the link list. Refer to 2.1.12, “Dynamic LNKLST” on page 21 for more information.	
Dynamic LPA	1.4	It is possible to dynamically add and remove modules from the link pack area. Refer to 2.1.13, “Dynamic LPA” on page 22 for more information.	
Dynamic Exits	1.4	Many exit points support the ability to dynamically add or modify exit routines. Refer to 2.1.14, “Dynamic Exits” on page 22 for more information.	
Exit elimination	N/A	Many exits may no longer be required, either because the business need has gone away or because the function provided by the exit has since been included in the product. Refer to 2.1.15, “Exit elimination” on page 23 for more information.	
Dynamic subsystems	1.4	It is possible to dynamically add, but not modify, subsystems. Refer to 2.1.16, “Dynamic subsystems” on page 23 for more information.	
Dynamic PPT	1.4	You can dynamically modify the Program Properties Table. Refer to 2.1.17, “Dynamic PPT” on page 24 for more information.	
IEASYMUP to update system symbols dynamically	1.6	With some restrictions, System Symbols can be dynamically added and modified. Refer to 2.1.18, “Dynamic update of System Symbols” on page 25 for more information.	
Extended Alias Support (also known as Symbolic Alias Facility)	1.4	Use this in tandem with system symbols to more easily move back and forth between subsystem releases. Refer to 2.1.19, “Extended Alias Support” on page 25 for more information.	
Monitor for hangs in structure rebuild processing	1.4	XES monitors that all structure connectors respond to a rebuild request. Any connector that does not respond will be identified by a console message. Refer to 2.1.20, “Structure rebuild hangs” on page 26 for more information.	

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
OPERLOG	1.4	Use the OPERLOG capability to have a sysplex-wide view of syslog. Refer to 2.1.21, "OPERLOG" on page 27 for more information.	
LOGREC in CF log stream	1.4	Using a CF log stream for LOGREC enables a single, sysplex-wide view of software and hardware problems and errors. Refer to 2.1.22, "LOGREC in CF log stream" on page 27 for more information.	
Update log stream attributes	1.4	It is now possible to update the attributes of a log stream while the log stream is connected. Refer to 2.1.23, "Dynamically changing log stream attributes" on page 28 for more information.	
Force disconnection or deletion of a log stream	1.7	SETLOGR FORCE command provides option to force a log stream connection, avoiding a restart of the System Logger address space. Refer to 2.1.24, "Forcing disconnection of a log stream" on page 28 for more information.	
Update REXX environment variables	1.7	It is possible to update the REXX environment variables table (IRXANCHR) without an IPL. There are also changes in this area in z/OS 1.7. Refer to 2.1.25, "REXX environment variables" on page 29 for more information.	

2.1.1 Console message backup

z/OS keeps WTO and WTOR messages in buffers in virtual storage pending delivery to all eligible consoles. In addition, the WTOR buffers each hold one WTOR message that the system has already displayed but that an operator has not responded to.

The maximum number of WTO and WTOR buffers are determined by the MLIM and RLIM parameters on the INIT statement in the CONSOLxx Parmlib member. If these parameters are not coded, the system defaults (as described in *z/OS MVS Initialization and Tuning Reference,* SA22-7592) are in effect. To avoid WTO message buffer shortages, the WTO buffer limit can be raised (using **`CONTROL M,MLIM=`**) and adjustments made to the console message deletion specifications. This command is in effect for the duration of the IPL. To avoid WTOR message buffer shortage, raise the WTOR buffer limit (**`CONTROL M,RLIM=`**) and reply to WTORs more frequently.

Procedures for responding to WTO and WTOR buffers shortages can be found in the "Responding to Console Message Backups" section of *z/OS MVS System Commands,* SA22-7627. Also, refer to the White Paper entitled *Console Performance Hints and Tips for a Parallel Sysplex Environment.*

If WTO or WTOR buffer use often reaches 80% of the limit, the limit for either or both specified at IPL might be too low to handle the message traffic in the system. The system programmer should modify the MLIM or RLIM parameter on the INIT statement in the CONSOLxx member to raise the buffer limit for the next IPL. You should also take all reasonable steps to minimize the number of consoles each message is sent to, especially attempting to minimize or eliminate the number of consoles defined with MSCOPE=*ALL.

For more information, see the following:

- IBM White Paper entitled "*Console Performance Hints and Tips for a Parallel Sysplex Environment*" available on the Web at:

  `http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130166.pdf`
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627

2.1.2 Consoles Enhancements

In 2004, IBM began to deliver components of its Consoles strategy to enhance the operator messaging architecture of z/OS. The feature focuses on minimizing the possibility of outages due to exhaustion of system resources used for messaging. The overall objective of the z/OS consoles enhancements feature is to improve system availability by enhancing the capacity and reliability of message delivery, reducing the likelihood that a consoles-related problem will result in a system IPL. To accomplish this, major changes to the message production and consumption flow help reduce the possibility of bottlenecks which can cause a backlog of undeliverable messages.

The console enhancement feature was made available as a separately orderable feature on z/OS 1.4. The FMID associated with this console restructure code is JBB7727 and was known as z/OS 1.4.2. Toleration PTFs for OS/390 2.10 and z/OS systems up to z/OS 1.4 are required as per APAR OW56244. As of z/OS 1.5, the function is rolled into the base product and is no longer an optional feature. Several tasks that were formerly best practices are now required, and some functions are no longer relevant.

Several prerequisites need to be addressed prior to implementing z/OS 1.4.2 or 1.5 and above. These are documented in the "Make updates for console enhancements" section of *z/OS V1R7.0 Migration,* GA22-7499.

IBM has announced that z/OS 1.7 is the last z/OS release that will support the one-byte console ID interface. With the advent of four-byte console IDs (in MVS SP V4.1.0), customers and vendors were encouraged to migrate away from the use of one-byte interfaces. To help prepare for the removal of this interface, a Console ID Tracking facility has been provided that will identify uses of the one-byte console ID interface in the environment.

Tip: The CNIDTRxx Parmlib member is used to control the Console ID Tracking facility. The member can be used to define a list of exclusions. A sample member for each of releases z/OS 1.4.2 through to 1.7 is available for download on the Web at:

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

More information about the consoles enhancements and the one-byte console ID interface can be found in the following:

- *IBM Health Checker for z/OS: User's Guide,* SA22-7994
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS Planning: Operations,* SA22-7601
- *z/OS V1R7.0 Migration,* GA22-7499
- IBM Redbook *z/OS Version 1 Release 5 Implementation,* SG24-6326

2.1.3 CMDS command

The `CMDS` command was introduced with z/OS 1.2 and is used to display or terminate commands that are currently active. Prior to this feature being available, it was possible for long-running commands to result in deadlock contention situations, for which an IPL was used to resolve the problem. This was most likely to occur when issuing Vary commands against a broken device. Because the deadlock can now be removed via the `CMDS` command, these IPLs are no longer necessary.

The `CMDS` command can be used to get a list of running commands and how long they have been running for, to get detailed information on each command, and to cancel running and queued commands.

There are some caveats to the use of this command:

- The command abends the TCB in the Master address space associated with the command requested to be terminated. The Master address space may have "spawned" this work into another address space. For example, an `ACTIVATE` command may still be active as a task in IOSAS after the command task has been abended with a `CMDS ABEND`. This results in the command not actually being terminated.
- Some commands remain active indefinitely, so the system will display them whenever `CMDS` is issued. For example, if any SLIP commands have been issued and SLIP traps are in effect, one SLIP command will be "executing" until all traps are deleted. This is also true for many SET commands, such as `SET SLIP` and `SET MPF`.

For more information see the following:

- *z/OS MVS System Commands,* SA22-7627

Note: Use the ABEND option with extreme caution, being careful to avoid leaving the system in an inconsistent state. Use this parameter only when all other possibilities have been explored.

2.1.4 Linkage indexes

Prior to z/OS 1.6 and z890/z990, the limit on the number of linkage indexes (LXs) was 2048. Some of the LXs are reserved as system LXs; the rest are available as non-system LXs. This total is the sum of the non-system LXs, plus the system LXs, plus the value specified (up to 512) on the NSYSLX parameter in the IEASYSxx member of Parmlib. This parameter allows an installation to specify the number of LXs (in addition to those in the system function table) to be reserved for system LXs.

If the NSYSLX parameter is omitted, the system defaults to 165 system LXs. NSYSLX may need to be specified if *either* of the following conditions is true:

- An installation runs applications that request (through the LXRES macro) more than 165 system LXs.
- An application that owns one or more system LXs fails and is restarted repeatedly. If the application does not reuse its original LX value, the supply is eventually exhausted. This condition requires an IPL to reclaim the system LXs.

If applications use more than 165 system LXs, specify the NSYSLX value a little higher than the number of system LXs used. If an application that owns one or more system LXs continues to fail, specify the NSYSLX value high enough so that, during processing, enough system LXs are available. This technique allows you to run longer between IPLs to reclaim these non-reuseable LXs, or at least to schedule the IPL at a less disruptive time.

System LXs allow service providers to connect an entry table to all address spaces. The recommendation is to not use a system LX unless the service is intended for all address spaces. Establishing a system LX makes the ASID of the address space unusable until the next IPL. z/OS sets aside part of the available LXs for use as system LXs. When the service provider connects an entry table to a system LX, the entry table is connected to all present and future address spaces.

Unlike a non-system LX, a system LX cannot be freed for reuse in releases prior to z/OS 1.6. When an address space that owns a system LX terminates, the LX becomes dormant. The system allows a dormant system LX to be reconnected to an address space different from the original owning address space. This is an important consideration for a service provider that can be terminated and then restarted. The service provider must have a way to remember the system LX it owned so that it can connect the LX to an entry table when it is restarted.

Connecting an entry table with space-switch entries through a non-system LX to a system address space or a long-running address space (such as VTAM®, CICS®, DB2, or JES) makes the ASID of the owner of the entry table non-reusable. Therefore, to avoid unnecessary loss of ASIDs, IBM recommends that the following rules are followed:

- Use system LXs only when the cross-memory service is to be used by all address spaces and the cross-memory service provider is a long-running address space.
- Avoid connecting non-system LXs to long-running address spaces.

z/OS 1.6 introduced an enhancement called the LX reuse facility. This provides additional LXs and improves reusability of LXs. It is enabled when running on a z890 or z990 processor at driver level 55 or above, with APAR OA07708 installed. With this facility enabled, the limit on the number of LXs is 32768. Current releases of DB2 and WebSphere MQ both support this new capability. In addition, RRS in z/OS 1.7 provides support for the use of reusable LXs in its exits; this support can be exploited by future releases of the products that use RRS services.

System and non-system LXs are subdivided into 12-bit LXs and 24-bit LXs. 12-bit LXs are in the range 0-2047, and 24-bit LXs are in the range 2048-32768. In support of this, the NSYSLX keyword has been changed to support two parameters—a number of 12-bit system LXs and a number of 24-bit system LXs.

Recommendation: If you are writing an application that will use LXs, always request at least a 16-bit LX when running on z/OS 1.6 or later.

Use whatever is the biggest that works for you. If your PC number is formed using some sort of “add”, a 24-bit LX will work. If the PC number is formed using some sort of “LA”, then it depends on the AMODE: if the AMODE is never 24, then 23 works; if it is possible that the AMODE will be 24, then use 16.

IBM WSC Flash 10273 explains how ASIDs and LXs are assigned and freed, and under what conditions they become unavailable for use. More importantly it explains what actions can be taken to remedy the problem without having to resort to an IPL.

Automation tip: z/OS 1.3 introduced monitoring for Linkage Index shortages, issuing messages when 85% of the available LXs are in use. We recommend adding code to your automation package to send an alert to the responsible system programmer when any of messages IEA063E, IEA065E, IEA066I, IEA070E, IEA071I, IEA072E, or IEA073I is issued.

For more information see the following:

- IBM WSC Flash10273 entitled "Non-Reusable Address Space IDs" available on the Web at:

  `http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10273`

- "Reassigning LXs when the LX Reuse Facility is enabled" and "Reassigning LXs when the LX Reuse Facility is not enabled" in the Synchronous Cross Memory Communication chapter of *z/OS V1R6.0 MVS Extended Addressability Guide,* SA22-7614.

2.1.5 Non-reusable address spaces

The system assigns an ASID to an address space when the address space is created. There is a limit on the number of ASIDs that are available for the system to assign. When all ASIDs are assigned to existing address spaces, the system is unable to start a new address space. This condition might be the result of defining too few address spaces, or having too many lost ASIDs in the system; these are known as non-reusable address spaces.

A *non-reusable address space* is one where a job that had been running in a cross-memory environment has ended. When such a job ends, the system ends the address space and marks its associated address space vector table (ASVT) entry non-reusable (unavailable) until all of the address spaces the job had cross-memory binds with have ended; this is required in order to protect the integrity of those other address spaces.

There are three parameters (MAXUSER, RSVSTRT and RSVNONR) coded in the IEASYSxx member of Parmlib that control the way the systems handles non-reusable address spaces. These parameters (which are not dynamically changeable) have a bearing on the amount of time a system can remain active before an IPL is required to reclaim the unavailable address spaces. The longer a system remains active, the more attention has to be given to monitoring the number of non-reusable address spaces.

IBM WSC Flash 10273 (referenced in 2.1.4, "Linkage indexes" on page 12) and the following section detail ways in which an installation might monitor the use of address spaces, and remedial actions that can be taken to prevent or forestall the need for an IPL to free up the non-reusable address spaces.

MAXUSER

This parameter (contained in the IEASYSxx member of Parmlib) specifies a value that, under most conditions, the system uses to limit the number of jobs and started tasks that can run concurrently during a given IPL. The number includes time sharing jobs, batch jobs, started tasks, the Master scheduler, and JES2 or JES3. MAXUSER entries can also include ASIDs that have been marked non-reusable if their total number exceeds the RSVNONR value.

This parameter is also used to allocate console control block areas in CSA that contain run-time job description data. This value is extended for started tasks by the value specified for RSVSTRT and by the value specified for RSVNONR when non-reusable ASIDs exist. In effect, this parameter limits the total number of address spaces an installation would like running at any one time.

RSVNONR

This parameter specifies the number of entries in the ASVT that are to be reserved for replacing entries that are marked non-reusable. An entry is taken from the RSVNONR "pool" to replenish MAXUSER and RSVSTRT entries, keeping them at their specified levels. If RSVNONR entries are not available, an address space that is marked non-reusable will result in the total number of available address spaces being depleted.

RSVSTRT

This parameter specifies the number of entries in the ASVT that are to be reserved for address spaces created in response to a **`START`** command (such as an initiator, the APPC address space, or the Library Look Aside address space).

The RSVSTRT entries will only be used after all the number of entries specified by MAXUSER have been used. By reserving RSVSTRT entries in the ASVT for such address spaces, IPLs that are due to no available ASVT entries for a critical address space can often be delayed. (However, it is not recommended to depend on using the RSVSTRT entries under normal operating conditions.) When planning for longer intervals between IPLs, the values set for these parameters need to be given careful consideration.

Even if all RSVNONR spare address spaces have been used, work will continue as long as capacity remains in MAXUSERs. When all RSVNONR entries are used, address spaces marked non-reuseable are no longer replaced, and the overall MAXUSER capacity will start to decrease. This will eventually lead to a need for an IPL. Eventually a shortage of available address spaces could lead to the system refusing to start new jobs or TSO users.

As stated previously, installations need to code these parameters to cope with peak usage and unexpected problems that may result in ASVT entries being marked as nonreusable. To identify appropriate values, you need to gather initial information:

- The maximum number of address spaces expected to be active at any given time.
- The depletion rate of ASVT entries per day due to the termination of address spaces holding cross-memory binds.
- The number of days expected between planned IPLs.

The maximum number of address spaces active at any given time can be identified by summing the totals returned on the **`DISPLAY A`** command, or from products like the IBM Omegamon, MXI (a product owned by Rocket Software), or other third party monitors. Determine the time when the largest number of address spaces will be active and take samples accordingly. As with RSVNONR, it is wise to code a “buffer” percentage to cater for unexpected increases in address space numbers.

Determining the depletion rate of ASVT entries is not as straightforward. There are several ways of arriving at a figure for this:

- Follow the instructions described in WSC Flash10273. That Flash documents the use of IPCS to arrive at a total number of depleted ASVTs since the last IPL.
- The MXI product contains the number of non-reuseable address spaces on the ASID panel, in the field Marked Not-Reusable.
- Code a REXX exec similar to that in Example 2-1. This exec has been tested on z/OS 1.6; however, use care regarding the ASVT offsets. Refer to *z/OS MVS Data Areas, Vol 1 (ABEP-DALT),* GA22-7581, under ASVT to check whether the offsets are correct for other releases.

Example 2-1 REXX exec to identify number of non-reuseable address spaces

```
/* REXX */
    ASVT     = C2X(STORAGE(D2X(C2D(STORAGE(10,4))+556),4))
    ASVTAAV  = C2D(STORAGE(D2X(X2D(ASVT)+480),4))
    ASVTAST  = C2D(STORAGE(D2X(X2D(ASVT)+484),4))
    ASVTANR  = C2D(STORAGE(D2X(X2D(ASVT)+488),4))
    ASVTSTRT = C2D(STORAGE(D2X(X2D(ASVT)+492),4))
    ASVTNONR = C2D(STORAGE(D2X(X2D(ASVT)+496),4))
    ASVTMAXI = C2D(STORAGE(D2X(X2D(ASVT)+500),4))
    ASVTMAXU = C2D(STORAGE(D2X(X2D(ASVT)+516),4))
```

```
SAY 'ASVT ADDR =  'ASVT
SAY ''
SAY 'MAXIMUM ADDRESS SPACE    =' ASVTMAXU
SAY 'NBR FREE SLOTS ASVT      =' ASVTAAV
SAY 'NBR FREE SLOTS START     =' ASVTAST
SAY 'NBR FREE SLOTS NON-REUSE =' ASVTANR
SAY 'IEASYSXX RSVSTRT         =' ASVTSTRT
SAY 'IEASYSXX RSVNONR         =' ASVTNONR
SAY 'ORIGINAL MAX USERS       =' ASVTMAXI
SAY ''
```

This produces a display similar to the following:

```
ASVT ADDR =  00FAB300

MAXIMUM ADDRESS SPACE    = 700
NBR FREE SLOTS ASVT      = 270
NBR FREE SLOTS START     = 100
NBR FREE SLOTS NON-REUSE = 194
IEASYSXX RSVSTRT         = 100
IEASYSXX RSVNONR         = 200
ORIGINAL MAX USERS       = 400
```

All of the above techniques return the total number depleted since the last IPL. To arrive at a daily rate:

1. Issue the **DISPLAY IPLINFO** command and work out how long, in days, the system has been active for.
2. Divide the number of depleted ASVTs by the number of days since the last IPL to arrive at an average daily depletion rate.
3. Calculate how many days the system is expected to remain active between IPLs. The RSVNONR value should (at least) equal:

   ```
   (Planned number of days between IPLs * daily depletion rate) + 5%
   ```

   The 5% is added as a buffer. Also, the IEA061E message is only triggered when just 5% of the pool remains.

This provides a reasonably accurate estimate. However, monitor the available ASVTs throughout the life of the IPL, checking for unexpected growth. If you detect such growth, take remedial actions immediately.

Following user experiences of system wait states caused by specifying a very high MAXUSER and CSCBLOC=BELOW[1], changes were made in z/OS 1.3. This introduced code to monitor the usage of RSVNONR and MAXUSER ASVT slots and issue appropriate messages when the associated pool of available slots drops below 5%. Also, the initial allocation of storage at IPL time was changed so that only a percentage of the total is allocated up front; this greatly reduces the risk of related wait states at IPL time. They are then allowed to grow as required.

Automation tip: A new message, IEA061E, was introduced with z/OS 1.3 and is issued as a warning when the number of ASIDs available for replacing non-reusable address spaces has dropped below 5% of the value specified on the RSVNONR parameter.

Add code to your automation package to send an alert to the responsible systems programmer when this message is issued.

[1] CSCBLOC=ABOVE is the default and should be used if possible.

As stated previously, a buffer should be factored into the MAXUSER and RSVNONR values. However, be realistic when coding these figures. Coding excessively large values could still eventually result in an ABEND 878 reason code 08 when applications are started after IPL.

For the RSVSTRT value, the IBM recommendation is to stay with or close to the default of 5. The reason for this recommendation is that the system only starts using this pool when all MAXUSER entries have been obtained (non-reusable plus active). This means that the system has no entries for TSO or initiator address spaces and will probably require an IPL.

For more information, refer to the following:

- Technote discussing the IEA061E message available on the Web at:
  `http://www.ibm.com/support/docview.wss?uid=isg1zTechnote0001`
- WSC Flash 10273, available on the Web at:
  `http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10273`
- *z/OS MVS Data Areas, Vol 1 (ABEP-DALT),* GA22-7581
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592 (use the -11 level of this manual; earlier releases had inaccurate information)
- “Reusing ASIDs” in the Synchronous Cross-Memory Communication chapter of *z/OS V1R6.0 MVS Extended Addressability Guide,* SA22-7614

2.1.6 PAGEADD command

If there is a higher than anticipated use of processor storage and insufficient page space has been defined, you can experience an auxiliary storage shortage, indicated by messages IRA200E or IRA201E.

Note: Use the formula contained in “Sizing Local Page Data Sets” in *z/OS MVS Initialization and Tuning Guide,* SA22-7591, to understand the recommended paging subsystem capacity. To maintain optimal performance, do not run your page data sets more than 30% full in normal operation.

If you encounter the situation where you are running out of auxiliary storage, one way to address this is to allocate more page space. However, once you are in an auxiliary storage shortage situation, it may not be possible to submit any jobs to define a new page data set. Therefore, you should have a number of spare page data sets defined in advance.

As long as you have spare page data sets defined, you can use the **`PAGEADD`** command to add auxiliary storage space (local page data sets), up to the maximum number as specified on the PAGTOTL parameter in IEASYSxx, without the need to IPL. Because the PAGTOTL value cannot be changed without an IPL, it is very important to set it to a larger value than the number of page data sets you currently have. Note that, in z/OS 1.5, the default PAGTOTL value increased from five to 40. There is no cost in specifying too large a value, but specifying too small a value could result in an IPL.

Note: Some automation products provide the ability to issue the **`PAGEADD`** command on your behalf, using spare page data sets that you have previously defined to the product.

The page data sets added remain available to the system until the next IPL with the CLPA (clear link pack area) option, IPL with the clear virtual I/O (CVIO) option, or you issue a **`PAGEDEL`** command.

Automation tip: Messages IRA200E, IRA201E, or ILR009E indicate a shortage, or potential shortage, of auxiliary storage. Add code to your automation package to issue a `PAGEADD` command to add spare page data sets and to send an alert to the responsible systems programmer when any of these messages are issued.

For more information see the following:

- *z/OS MVS System Commands,* SA22-7627
- *z/OS MVS Initialization and Tuning Guide,* SA22-7591

2.1.7 PAGEDEL command

If you wish to remove a page data set that contains I/O errors, or you want to de-install the device containing the data set, you can use the `PAGEDEL` command. The `PAGEDEL` command allows you to drain and delete local page data sets without the need to IPL.

Be aware that the `PAGEDEL` command requires x'500' bytes in SQA (below the line) for every cylinder in the page data set being drained that contains valid data. This storage is required for the full amount of time it takes for the command to complete. Therefore, if you foresee the use of this command, ensure that you have defined SQA to be sufficiently large.

For more information see the following:

- *z/OS MVS System Commands,* SA22-7627
- *z/OS MVS System Messages, Vol 7 (IEB-IEE),* SA22-7637

Notes on PAGEDEL:

- `PAGEDEL` cannot be used to delete or drain the PLPA, common, or the last local page data sets.
- If a `PAGEDEL` command is entered while another `PAGEDEL` command is already in progress, the system issues a message and rejects the command.
- The system rejects a `PAGEDEL` command that decreases the amount of auxiliary storage below a fixed percentage of the previously-available auxiliary storage.
- To identify the page data sets the system is currently using or the status of the `PAGEDEL` command, issue the `DISPLAY ASM` command.

2.1.8 SVCUPDTE macro

There are two aspects to adding or replacing an SVC:

- The code must be made available to the system.
- The SVC table must be updated with information about the current program.

Making the code available

The Dynamic LPA function, described in 2.1.13, "Dynamic LPA" on page 22, can be used to make the code available. This can even be used for nucleus-resident SVCs as long as they do not contain any external references and you specify that the module should be page-fixed.

Updating the SVC table

This is handled by the SVCUPDTE service. Authorized users can use the SVCUPDTE macro to dynamically replace or delete SVC table entries, or return the SVC number of a routine at a specified entry point, thus avoiding unnecessary IPLs. Callers who use this service are responsible for providing recovery. Be aware that improper deletion or replacement of system-provided SVC routines causes unpredictable results and might terminate the system.

The resource name, SYSZSVC TABLE, is available as the operand of an ENQ or DEQ macro, to be used when serializing the execution of a program that uses the SVCUPDTE macro.

For example, authorized subsystems such as VTAM can alter the SVC table when the subsystem starts, and then restore the table when the subsystem terminates. For additional flexibility, the EPNAME and EXTRACT parameters of the SVCUPDTE macro allow an authorized user to dynamically associate SVC numbers with entry points of SVC routines. A sample program to call SVCUPDTE is shown in Example 2-2.

Example 2-2 Sample program to invoke SVCUPDTE

```
SVCUPDTE TITLE 'PROGRAM TO DYNAMICALLY UPDATE THE SYSTEM SVC TABLE'
*
SVCUP    CSECT
SVCUP    AMODE 31
SVCUP    RMODE ANY
         USING SVCUP,12
*
         SAVE  (14,12),,&SYSDATE-&SYSTIME-SVCDC
         LR    12,15              BASE.
         LR    14,13              SAVE HI-SAVE PTR.
         LA    13,SAVE            POINT AT LO-SAVE.
         ST    14,4(,13)          CHAIN LO- &..
         ST    13,8(,14)          .HI-SAVE.
         L     2,X'10'(,0)        POINT AT CVT.
         TM    X'74'(2),X'80'     TEST, WHETHER..
         BZ    SVCUPA             .NOT RUNNUNG MVS/XA.
         LA    2,SVCUPA           SET..
         O     2,HBITOW           .31-BIT..
         BSM   0,2                ADDR MODE.
SVCUPA   DS    0H
*
         ENQ   (ENQ_MAJ_NAME,ENQ_MIN_NAME,E,,SYSTEM),RNL=NO
*
         MODESET KEY=ZERO,MODE=SUP
*
         SVCUPDTE 216,REPLACE,TYPE=3,EPNAME=DFHCSVC
*
         MODESET KEY=NZERO,MODE=PROB
*
         DEQ   (ENQ_MAJ_NAME,ENQ_MIN_NAME,,SYSTEM),RNL=NO
*
*
         SR    15,15
         L     13,4(,13)
         RETURN (14,12),RC=(15)
*
ENQ_MAJ_NAME DC CL8'SYSZSVC'
ENQ_MIN_NAME DC C'TABLE'
*
         DS    0F
HBITOW   DC    X'80000000'
SAVE     DC    18F'0'
*      THE FOLLOWING MACROS MUST BE INCLUDED IN THE SOURCE PROGRAM
*      CVT    - TO MAP THE FIELD CVTNUCLU
*      IHAPSA - TO SUPPLY CVT BASE
*      NUCLKUP- TO FIND THE SVC UPDATE SERVICE ENTRY POINT (IEAVESTU)*
         PRINT NOGEN
         IHAPSA
         IKJTCB
         CVT   DSECT=YES
         END   SVCUP
```

An SVC update recording table is maintained in parallel with the SVC table. This table provides a record of changes to the SVC table. Entries are created whenever a change is made to the SVC table with IEASVCxx Parmlib member statements or the SVCUPDTE macro.

For more information see the following:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS Authorized Assembler Services Guide,* SA22-7608
- *z/OS MVS Authorized Assembler Services Reference SET-WTO,* SA22-7612
- *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST),* GA22-7585

2.1.9 Increased I/O parallelism at IPL time

Building the dynamic path arrays at IPL time used to be a serial process. For customers with very large numbers of devices, or if the devices are a long distance from the server, this processing could potentially take many minutes.

APAR OA07335, available back to z/OS 1.4, changed this process to make it run in parallel. Some customers have experienced a reduction of up to 75% in that part of IPL processing following application of this APAR.

Install this APAR on z/OS 1.4 or 1.5 (it is included in the base of z/OS 1.6). Customers with non-IBM disk subsystems should refer to the APAR text for additional relevant information.

2.1.10 Reducing IPL times

SMF Type 19 records contain information about DASD volumes that are online at IPL time. If you have a large number of online DASD devices, this could add up to a number of minutes of processing time.

If you do not use SMF Type 19 records for anything, turn them off in the SMFPRMxx member. This will eliminate the processing associated with collecting this information and potentially reduce your IPL times.

For more information about controlling SMF, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Management Facilities (SMF),* SA22-7630

2.1.11 Dynamic APF list

Libraries that are to be APF-authorized can be defined in the IEAAPFxx member of Parmlib, or in one of the PROGxx members of Parmlib. Libraries defined in the IEAAPF member are in what is known as the *static APF list* (limited to a maximum of 255 libraries). Libraries defined in the PROGxx member, or by using the **`SETPROG APF`** command, are in the *dynamic APF list*, which does not have a limit on the number of members. All customers should use the PROGxx mechanism to define the APF libraries.

There are two ways to dynamically add a library to the dynamic APF list:

- Update the PROGxx member and issue a **`SET PROG=xx`** command.
- Define the library name and attributes directly on the command line using the **`SETPROG APF,ADD,...`** command.

You cannot modify an existing APF entry, but if you move an APF-authorized library from one volume to another, for example, you can set up a new definition for the new location, and delete the definition for the old volume after the move is complete.

For more information about the use of dynamic APF, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627
- A presentation about the dynamic capabilities of z/OS is available on the Web:

  `http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1876`

2.1.12 Dynamic LNKLST

System LNKLST is a way to make programs available to address spaces for execution without those address spaces having to know (for example, by specifying on a JOBLIB or STEPLIB) which library the program resides in.

Typically, dynamic updates to the LNKLST are made for one of the following reasons:

- You wish to install a new product, typically one that has not been used on this system before. You would use the LNKLST DEFINE/ADD/ACTIVATE keywords in this case.
- You need to remove a library from the LNKLST in order to enable some processing to take place against it. You would use the LNKLST DELETE keyword for this.
- You must work with uncataloged data sets of the same name as data sets in the LNKLST. You would use the LNKLST UNALLOCATE/ALLOCATE keywords for this.

Similarly to APF libraries, there are two ways you can define a library as being in LNKLST: you can define it in the LNKLSTxx member of Parmlib, or you can define the LNKLST sets in one of the PROGxx members of Parmlib. All customers should use the PROGxx mechanism to define the LNKLST libraries.

When you use the PROGxx member, you define what is known as a LNKLST set. When address spaces start, they receive a copy of the LNKLST set in effect at that time. Normally, they will continue to use that set for the life of the address space.

If you want to dynamically change the LNKLST libraries, you would create a new LNKLST set (using either the **`SET PROG=xx`** or **`SETPROG LNKLST`** commands). When you subsequently activate the new LNKLST set, any address spaces that start execution *after* that time will use the new set. But any address spaces that are already executing will continue to use the LNKLST set that was in effect when they started. Normally this is fine, because the intent of the dynamic LNKLST capability is to let you install new products without an IPL, and normally the only address spaces that will use that new function are those that are started after you have activated the new LNKLST set.

However, never allow alteration of the *contents* of the related data sets while they are in an active LNKLST. For Dynamic LNKLST sets, remove the data set using the LNKLST DELETE parameter (LNKLST UNALLOCATE does not accomplish this task). Then, any changes (compress, linkedit/bind, delete, or rename members) desired may be safely executed once you are certain that no users exist for the members you are affecting. There are additional considerations relating to PDSE libraries in the LNKLST. Refer to 4.1.7, “DFSMS Restartable PDSE address space” on page 54 for more information.

Important: While it is possible to force an executing address space to start using the new LNKLST set rather than the one in place when it started, this is a very risky process and could result in affected address spaces abending.

The risk is partly related to what the affected address spaces are doing at the instant you issue the command—so having the command work on a test system is no guarantee that the same command will be successful on a production system.

For more information about the use of dynamic LNKLST, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627
- A presentation about the dynamic capabilities of z/OS, available on the Web:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1876

2.1.13 Dynamic LPA

Modules (and aliases) residing in libraries listed in the LPALST member of Parmlib will be loaded into LPA or ELPA at IPL time when a CLPA is specified. Dynamic LPA provides the ability to either add completely new modules into LPA, or to logically replace existing LPA modules, without an IPL. Whereas the dynamic APF and dynamic LNKLST functions operate at the library level, the dynamic LPA capability operates at the individual module level.

Modules added to LPA dynamically will actually be placed in CSA or ECSA, so if you plan to exploit this capability, it is important to plan to have sufficient free space in CSA and ECSA. When you issue the **`SETPROG LPA`** command to add a module dynamically, you can optionally specify that a certain amount of free space in CSA and ECSA (specified in bytes, KB, or MB) must still be available after the add—if the add would result in less than that amount being available, the add request will not be processed. Note that if you do an ADD multiple times for the same module (to test changed versions, for example), each ADD will add a new copy to CSA or ECSA - it does *not* replace the previous version.

Important: When you add modules to LPA dynamically, you must specify the module names or a name pattern on the **`SETPROG LPA`** command. If any of the modules have an alias, that alias will not be automatically be loaded; you *must* specify all alias names in addition to the actual module names.

For more information on dynamic LPA, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627
- A presentation about the dynamic capabilities of z/OS is available on the Web:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1876

2.1.14 Dynamic Exits

Normally, when a product contains exit points, the code for those exits is loaded at IPL time or when the product is initialized. This means that implementing changes to those exits is a disruptive process.

To remove the impact of such changes, z/OS provides the Dynamic Exits capability. For products that support this capability, the exit code can be changed and reloaded without a restart of the system or subsystem. For an overview of exits that have been defined to the Dynamic Exits facility, review “Dynamic Exits Facility” in *z/OS MVS Installation Exits,* SA22-7593.

Through the CSVDYNEX macro, you can define exits and control their use just as the system does when it offers installation exits. This macro is described in “Using Dynamic Exits Services” in *z/OS MVS Authorized Assembler Services Guide,* SA22-7608. To obtain a list of products and exit points that support this capability, issue the D PROG,EXITS command.

For more information about the z/OS Dynamic Exit capability, refer to:

- *z/OS MVS Authorized Assembler Services Guide,* SA22-7608
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS Installation Exits,* SA22-7593
- *z/OS MVS System Commands,* SA22-7627
- A presentation about the dynamic capabilities of z/OS is available on the Web:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1876

2.1.15 Exit elimination

Exits enable you to modify some aspect of the behavior of a product, to match specific requirements you may have. This can improve the usability or value of a product to your installation. However, the downside of providing your own exits is that they then must be maintained, updated, and tested every time you upgrade your system.

For example, it is not unusual to come across exits that are regularly maintained and tested, yet no one knows if they are still even required or not. Maintaining such exits is a time-consuming process, usually requiring knowledge of Assembler. It elongates the time required to move to new releases of the operating system. And updates to the exits may require a subsystem or system outage if they need to be updated.

For these reasons, critically review every exit each time you install a new release of the related component. Maybe the function the exit delivers is no longer required. Or maybe the product has been enhanced to deliver the required function without having to provide an exit. Some installations go as far as omitting any exits for which they are unable to identify a business owner every time they upgrade the system; if no one complains, the exit presumably was not required. And if someone does complain, the business need often disappears when they are asked to cover the cost of upgrading and testing the exit for the new release.

2.1.16 Dynamic subsystems

Subsystems are normally defined in the IEFSSN member of Parmlib. However, if you are installing a new product, or adding a new subsystem for an existing product (DB2, for example), it is possibly to add the subsystem dynamically using the Dynamic Subsystem support in z/OS.

The **SETSSI** command allows you to add, activate, or deactivate a subsystem. You can specify essentially the same information on the **SETSSI** command as you can include in the IEFSSN member. However, you cannot control the position of the subsystem on the subsystem interface (this is achieved based on the relative placement of the defining statement in the IEFSSN member). You also cannot specify that a new subsystem is to be the Primary subsystem, but it really does not make sense to try to add a new Primary subsystem except at IPL time anyway.

If the subsystem you wish to add uses an initialization routine that is located in a library that you added to LNKLST using the dynamic LNKLST support, you *must* refresh the LNKLST set used by the Master Scheduler before you try to add the subsystem. This is achieved using the following command:

```
SETPROG LNKLST,UPDATE,JOB=*MASTER*
```

Note the warnings relating to the use of the UPDATE command in 2.1.12, “Dynamic LNKLST” on page 21 before issuing this command.

Tip: You cannot delete a subsystem dynamically, nor can you modify an existing subsystem definition. So, if you make a mistake on the **SETSSI ADD** command, you cannot undo that mistake until the dynamic subsystem is removed at the next IPL.

There is no command to allow you to update the IEFSSN member and load the definitions from the newly updated member. Therefore, it is especially important to remember to reflect the subsystem definition back into the IEFSSN member that will be used for the next IPL.

In addition, note that the **SETSSI ACTIVATE** or **DEACTIVATE** commands can only be used with subsystems that were added dynamically or were defined in the IEFSSN member using the keyword (rather than the old positional) format.

For more information about the z/OS dynamic subsystem capability, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627
- A presentation about the dynamic capabilities of z/OS is available on the Web:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1876

2.1.17 Dynamic PPT

Additions to the IBM-supplied Program Properties Table are defined in the SCHEDxx member of Parmlib. The IBM-supplied PPT entries are listed in the SCHED description in *z/OS MVS Initialization and Tuning Reference,* SA22-7592.

Products that require a PPT entry should furnish the statements that need to be added to the SCHEDxx member. To add these entries dynamically, update the SCHEDxx member as appropriate and the issue the **SET SCH=xx** command to activate the PPT definitions from that member.

Note that other system attributes defined in the SCHEDxx member, such as the size of the Master Trace table, are *not* read when you issue the **SET SCH** command. Only the parameters that define PPT entries are processed.

For more information about the z/OS dynamic PPT capability, refer to:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627

2.1.18 Dynamic update of System Symbols

System symbols provide an extremely powerful way to effectively manage a sysplex environment. You can use symbols to minimize the number of definitions required to describe multiple systems. For example, you could have a single IEASYSxx member for all sysplex members, but specify unique page data set names as follows:

```
....
PAGE=(PAGE.&SYSNAME..PLPA,
      PAGE.&SYSNAME..COMMON,
      PAGE.&SYSNAME..LOCAL1,L),
....
```

System symbols can be used in Parmlib definitions, started task JCL, and many other places (for a detailed discussion about the use of system symbols, refer to IBM Redbook *Parallel Sysplex - Managing Software for Availability,* SG24-5451). The intelligent use of system symbols significantly reduces the number of separate definitions that must be maintained in a sysplex environment, and therefore reduces the workload to maintain the sysplex systems.

One drawback of system symbols, however, is that there is no fully supported mechanism to change the value of the system symbol without an IPL. However, as of z/OS 1.6, IBM provides a member called IEASYMUP in SYS1.SAMPLIB. This sample program provides the ability to add or change system symbols dynamically. More information about IEASYMUP, including its use and restrictions, is contained in Appendix B, “IEASYMUP” on page 133.

2.1.19 Extended Alias Support

Moving to a new release of software, especially for a major subsystem, can be a very time-consuming process. A significant part of that time is taken renaming data sets or updating JCL and processes to reflect the data set names of the new release. This is especially important if you need to back out the change, because the subsystem will be unavailable until all these changes can be regressed.

A function introduced in OS/390 2.7 called Extended Alias Support provides the ability to define a catalog entry containing a system symbol (it is really a DFSMS feature, but this is a more logical place to mention it).

The following example shows how this might work:

- SYS1.V1R3M0.LOADLIB is a software data set.
- Users reference this data set through alias SYS1.LOADLIB, which was defined using the SYMBOLICRELATE keyword on the DEFINE ALIAS command. Using this keyword, the following code is used to define this data set:

  ```
  DEFINE ALIAS (NAME(SYS1.LOADLIB) -
        SYMBOLICRELATE(‘SYS1.&PRODVR..LOADLIB’))
  ```
- Initially, the symbol &PRODVR would be set to V1R3M0 on all systems. When the new release is ready to be tested and released, the &PRODVR symbol would be changed on a system-by-system basis to V1R4M0 and the applications affected would be restarted. Thus, a shared catalog can be used, and users on two systems can use the same data set name—but you can have each user get a different data set. This technique can help to avoid IPLs that would be needed to deploy software changes.

Note: Every time a symbol is changed using IEASYMUP, it is imperative to update the corresponding IEASYMxx member. If this is not done, an IPL would regress the change.

When using this facility, care should be taken when choosing the real data set names that are used for the product libraries. For example, if a library is currently called SYS1.LOADLIB, and it is decided to move to SYS1.V1R3M0.LOADLIB and SYS1.V1R4M0.LOADLIB using a SYMBOLICRELATE of SYS1.&PRODVR..LOADLIB, then the original SYS1.LOADLIB library would be picked up if the system symbol &PRODVR was not defined. This can provide a safety net during the migration, but care needs to be taken to avoid inadvertently picking up the wrong library at a later time. Delete or rename the SYS1.LOADLIB data set after the migration.

Parmlib considerations for Symbolic Alias Facility

Some of the data sets you wish to use this facility with may need to reside in LPA, LNKLST, or be APF authorized. This methodology can be used successfully, even if the product is using LNKLST, LPALST (as long as they are placed in the Master Catalog), or APF authorized libraries.

Data sets that you use symbolic aliases with may be in LNKLST or LPA. However, because these members are processed very early in the IPL process, before all the Catalog facilities are available, you cannot use the symbolic alias name to define the data sets to LNKLST or LPA at IPL time. Rather, the PROGxx and LPALSTxx members must contain the fully-qualified data set names as follows:

```
LNKLST DEFINE NAME(LNKLST1)
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.LINKLIB) ATTOP
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.MIGLIB)
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.CSSLIB)
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.&PRODVR..LOADLIB)
LNKLST ACTIVATE NAME(LNKLST1)
```

LPA components fall into two categories.

1. Libraries referenced as part of non-dynamic LPA processing must use the data set coded with the symbolic name in a similar manner to LNKLST.
2. Libraries loaded into dynamic LPA using the PROG member or SETPROG command which can use either the data set name containing the symbol (as above) or the catalog alias.

Data sets requiring APF authorization must use fully qualified data set names because APF processing does not use the master catalog, so there is no way for it to pick up the alias. The recommend way to define these data sets to APF is to hardcode both fully-qualified library names (for example SYS1.V1R3M0.LOADLIB and SYS1.V1R4M0.LOADLIB) in the PROGxx or IEAAPFxx member.

For more information on the Extended Alias Support (also referred to as the Symbolic Alias Facility), refer to:

- “Extended Alias Support” in *z/OS DFSMS Managing Catalogs,* SC26-7409
- IBM Redbook *Parallel Sysplex - Managing Software for Availability,* SG24-5451

2.1.20 Structure rebuild hangs

Any time a CF structure is to be rebuilt, the initiating system sends a message to all connectors of that structure, informing them that the structure is about to be rebuilt. However, the rebuild cannot proceed until all connectors have responded that they have seen the message and are aware of what is about to take place.

Prior to a new function known as External Hang Detect, if one of the connectors did not respond, the rebuild would simply hang, with no external indication of the reason for the hang. The response to this was often that people would IPL the smallest system in the sysplex to see if that relieved the problem. If not, they would try the next largest system, then the next largest, and so on until they happened across the “right” system.

In OS/390 2.8, a new function called External Hang Detect was delivered. This function in XES monitors every incoming rebuild request, and issues a message if the address space that is the target of the request has not responded within two minutes. In this case, messages IXL040E and IXL041E are issued, identifying the structure and the offending address space. This provides you with the information you need to address the problem, and removing the need to IPL systems to free up the hang.

Automation tip: Add code to your automation package to send an alert to the responsible systems programmer when these message are issued.

2.1.21 OPERLOG

In a sysplex environment, there is a high level of interaction between the systems. One example is the CF structure rebuild process. This process consists of a number of steps, each of which can potentially run on a different member of the sysplex. So, the command to rebuild a structure could be issued on one system, but the messages indicating why the rebuild failed could be issued on a different member. Therefore, it is especially important in a sysplex environment that the operator interface provides a view of messages across all members of the sysplex.

One way to achieve this view is to use the MVS OPERLOG function. This involves sending a copy of the syslog messages to a CF log stream. This log stream can then be viewed from SDSF.

While OPERLOG does not provide the same levels of performance as normal SDSF SYSLOG processing, it is possible to have syslog messages sent to both SYSLOG and OPERLOG. This gives you the option of using SYSLOG for normal operations, but still having a consolidated sysplex view (OPERLOG) available to do problem determination. The additional cost of writing messages to both SYSLOG and OPERLOG is miniscule. We have encountered a number of customer IPLs that could have been avoided if the operators had access to and used such a sysplex-wide view of activities.

For more information on implementing and using OPERLOG, refer to:

- IBM Redbook *S/390 Parallel Sysplex: Resource Sharing,* SG24-5666
- IBM Redbook *Systems Programmers Guide to: z/OS System Logger,* SG24-6898

2.1.22 LOGREC in CF log stream

A much underused system resource is the information saved in the SYS1.LOGREC data set. This data set contains a great deal of valuable information that will help you identify and debug system software or hardware problems.

The default mode of operation is that each member of a sysplex will have its own LOGREC data set. However, in a sysplex environment, problems on one member of the sysplex are often caused by (or at least, related to) problems on other members of the sysplex. In this situation, it would be very helpful to have a single, sysplex-wide, repository of LOGREC information.

You can achieve this by sending the LOGREC records to a CF log stream. In this case, there is a single log stream for all members of the sysplex. This can help you identify and quickly address potential problems before they escalate to the point of bringing a system down.

Another advantage of using a log stream (as opposed to the SYS1.LOGREC data set) is that you are less likely to run out of space in the log stream, meaning that valuable records will not be overwritten before you see them.

For more information on setting up LOGREC to use a CF log stream, refer to:

- IBM Redbook *S/390 Parallel Sysplex: Resource Sharing,* SG24-5666
- IBM Redbook *Systems Programmers Guide to: z/OS System Logger,* SG24-6898

2.1.23 Dynamically changing log stream attributes

It used to be the case that it was not possible to change the definition of a log stream while any address space was connected to that log stream. The IXCMIAPU job to update the log stream definitions could only be run when no one was connected to the log stream. This was especially disruptive for large CICS users who could have hundreds or thousands of log streams.

With z/OS 1.3 and higher, it is possible to schedule the update of nearly all the attributes of a log stream while the log stream is connected. This requires that the LOGR CDS is formatted with the SMDUPLEX keyword (which requires z/OS 1.2 or later on all systems in the sysplex). You can determine the CDS format level using the **`D XCF,C,TYPE=LOGR`** command. The SMDUPLEX keyword enables many new System Logger functions in addition to support for System Managed CF Duplexing, and we recommend that all sysplexes where all systems are running higher than z/OS 1.2 should format their LOGR CDSs with this keyword.

For more information refer to the following:

- The IBM Redbook *Systems Programmers Guide to: z/OS System Logger,* SG24-6898
- *z/OS MVS Setting Up a Sysplex,* SA22-7625

2.1.24 Forcing disconnection of a log stream

It is possible, in some error situations, to end up in a situation where the address space that was connected to a log stream has ended, but the System Logger information relating to the connection has not been fully cleaned up. As a result, when you restart the address space, it is unable to connect to the log stream because System Logger thinks it is already connected. Prior to z/OS 1.7, the only way to address this situation was to restart the System Logger address space or to IPL that system.

In z/OS 1.7, two new options have been added to the **`SETLOGR`** system command. These options allow you to force disconnection of a connection that System Logger thinks is still in place, or to force the deletion of a log stream from the System Logger CDS. These commands should obviously be used with care. However, in the appropriate circumstances, they can save an IPL or a restart of the System Logger address space (which is tantamount to an IPL for many large Logger users).

For more information refer to the following:

- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- *z/OS MVS System Commands,* SA22-7627

Note: z/OS 1.8, which has only been previewed at the time of writing, contains a new function to permit the renaming of a log stream. This will allow you to save an existing log stream and its contents, for diagnostic purposes for example, while still being able to restart the application with a new empty log stream with the name it expects.

2.1.25 REXX environment variables

The REXX environment variable table contains the number of REXX environments. Prior to z/OS 1.6, the default was 40. However, this is the number allowed per *address space*, not per system, as many people believe. If this number of 40 is insufficient, you can:

- Update the IRXANCHR module in SYS1.LINKLIB and do a LNKLST refresh, or
- Create a private version of IRXANCHR with a larger value and point to that in the STEPLIB of the address spaces that require the larger number of environments.

In z/OS 1.6, a change was made to the number of ENVBLOCKs allowed by the default IRXANCHR module shipped by TSO/E. The intent was to increase the likelihood that the default number would be sufficient for almost all customer needs. The default has now been increased by a factor of 10, from 40 up to 401. If you still need to increase the value, there is a sample SMP/E usermod provided in the IRXTSMPE member of SYS1.SAMPLIB.

For more information see the following:

- *z/OS TSO/E REXX Reference*, SA22-7790

2.2 Global Resource Serialization enhancements checklist

Table 2-2 contains a checklist of enhancements to the Global Resource Serialization (GRS) component of z/OS.

Table 2-2 Checklist of outage avoidance items for GRS component of z/OS and OS/390

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
SETGRS command	1.4	This command allows dynamic modification of certain GRS parameters. See 2.2.1, “Dynamically changing GRS configuration” on page 30 for more details.	
Dynamic Resource Name List (RNL) change	1.4	Use the `SET GRSRNL=` command to change RNLs dynamically, removing the need to IPL a system. See “Dynamic RNL change” on page 30 for more details.	
Dynamic GRS exits	1.4	A number of exits are available that can be refreshed dynamically and augment GRS processing. See 2.2.2, “Dynamic GRS exits” on page 33 for more details.	
ANALYZE command	1.4	This command displays an analysis of system contention. See 2.2.3, “ANALYZE command” on page 33 for more details.	
Enhanced Contention Analysis Service (ISGECA)	1.5	The ISGECA macro is used to obtain waiter and blocker information for GRS component-managed resources. See 2.2.4, “Enhanced Contention Analysis Service (ISGECA)” on page 34 for more details.	
Contention Management	1.4	Helps relieve potential storage shortages due to inappropriate dispatching priorities of Event Notification (ENF) listeners. See 2.2.5, “Contention management” on page 35 for more details.	

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
Contention Notification System (CNS) Enhancement	1.7+	This is a GRS CNS enhancement to allow customers to move the CNS function to a system of their choice. Retrofitted to z/OS 1.7 by APAR OA11382. See 2.2.6, “Contention Notification System enhancement” on page 35 for more details.	
SVC Dump options	1.4	APAR OA06591 make tasks dispatchable earlier, and added the ability to have more granularity in specifying what GRS information should be in an SVC dump. See “GRSQ information” on page 119 for more details.	
IBM HealthChecker checks	1.7	The IBM HealthChecker for z/OS is an operating system component that checks the current active z/OS and sysplex settings and definitions for an image and compares their values to either those suggested by IBM or defined by the customer. See 2.2.7, “IBM Health Checker” on page 36 for more details.	

2.2.1 Dynamically changing GRS configuration

In the appropriate configuration, it is possible to change the following attributes of GRS dynamically:

- Change the mode from Ring to Star (but not back to Ring)
- Change the Resource Name Lists (RNLs) used by the members of the complex
- Change the RESMIL and TOLINT values
- Enable or disable the SYNCHRES feature

GRS Star

As long as all the members of the GRS are members of the same sysplex and the appropriate configuration is in place, you can use the **SETGRS MODE=STAR** command to dynamically move the sysplex from Ring mode to Star mode. Any installation that has two Coupling Facilities and more than one z/OS system in the sysplex should migrate to a GRS Star configuration.

Important: Backing out requires a *sysplex* IPL from an IEASYSxx member coded with the GRS=START, JOIN, or TRYJOIN option to implement Ring mode.

However, we are only aware of one customer that migrated to Star mode that had to revert to Ring mode.

Dynamic RNL change

The GRS RNLs contain the specifications for how resources are to be serialized by GRS. In order to guarantee data integrity, it is vital that all systems in the sysplex adhere to the same set of serialization rules at all times.

The **SET GRSRNL=** command can be used to make dynamic RNL changes without the need to IPL.

If configured in a Ring complex that includes systems that are not members of the sysplex, the non-sysplex systems must be removed from the GRS complex before initiating the change. After the change completes, the non-sysplex systems can be IPLed back into the

GRS complex. Plan the use of RNLs carefully when operating a mixed complex to avoid unnecessary IPLs.

If GRS is in Star mode, then all members of the GRS complex must, by definition, be in the same sysplex. Therefore, you should always be able to update the RNLs dynamically when in Star mode.

The RNLs are specified in the GRSRNLxx member referred to by the GRSRNL= parameter in the IEASYSxx member of Parmlib. If the RNLs used by the system during initialization do not match those being used by the other members of the complex, GRS issues message ISG312W and the system is put into an x'0A3' wait state.

To change the RNLs currently being used by GRS, set up the GRSRNLxx Parmlib members with the new RNLs and issue the **`SET GRSRNL`** command on a system that has access to those members. The new RNLs are then communicated to all systems in the complex. Although you only need to issue the command on one system to implement the change, keep in mind that every system must be updated to point to the updated GRSRNLxx Parmlib members in order to avoid an x'0A3' wait state at the next IPL.

Note: Before an RNL change can complete, special processing may need to be performed if any resources affected by the change are currently in use. These resources are known as “affected resources”. Jobs issuing new requests for these resources are suspended until the RNL change is complete across all members of the complex. These are known as “suspended jobs” and message ISG210E is issued on each system in the complex that has one or more suspended jobs.

If any job currently holds one or more of the affected resources, the change is delayed until all of the affected resources are freed. Jobs holding an affected resource (and thereby delaying the RNL change) are called “delaying jobs”. When jobs are holding affected resources and delaying the change, the following messages are issued on whichever console originated the RNL change:

```
ISG219E   RNL CHANGE WAITING FOR RESOURCES TO BE FREED.
          TO LIST DELAYING JOBS, USE ROUTE SYSNAME,DISPLAY GRS,DELAY.
          TO LIST SUSPENDED JOBS, USE ROUTE SYSNAME,DISPLAY GRS,SUSPEND.
ISG220D   REPLY C TO CANCEL RNL CHANGE COMMAND, OR S FOR SUMMARY OF RNL CHANGE PROGRESS.
```

If this occurs there are actions that can be taken to allow the change to proceed:

- Issue the **`DISPLAY GRS,DELAY`** operator command. It lists the jobs that hold affected resources and are causing the change to be delayed. When all the jobs listed release the affected resources, the RNL change completes.

  There might be instances where the operator must either cancel the RNL change or cancel jobs that hold the affected resources. Two common holders of resources are the CATALOG address space and LLA.

  The CATALOG address space can be recycled (see 4.1.2, “Freeing resources held by the Catalog Address Space” on page 50 for further details) to overcome this situation and various actions can be taken with regard to LLA, ranging from individual data set removal to stopping and starting LLA. Irrespective of the holder, careful consideration should be given to the actions needed to overcome the contention.

- Issue the **`DISPLAY GRS,SUSPEND`** operator command. It lists the jobs that are being suspended due to the RNL change. The jobs listed remain suspended until the RNL change completes, or until the RNL change is cancelled.

- Replying to message ISG220D with an `S` produces a summary of the RNL change progress. This summary indicates the number of jobs on each system that are delaying or are suspended by the RNL change.

  Replying to message ISG220D with a `C` causes the RNL change to be cancelled. If the operator chooses not to respond to message ISG220D with a C, the change will take place when all delaying jobs release the affected resources.

For more information see the following:

- *z/OS MVS System Commands,* SA22-7627
- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS System Messages, Vol 9 (IGF-IWM),* SA22-7639

Notes:

- After a change completes, the GRSRNLxx Parmlib member must be updated on every system in the sysplex to preserve the changes beyond the next IPL.
- Migration of a GRS Ring complex to a Star complex cannot take place while an RNL change is occurring.
- Test GRSRNLxx carefully to ensure that it does not include syntax errors. In SYS1.SAMPLIB, IBM supplies a syntax checker that can detect such errors. Member SPPINST of SYS1.SAMPLIB contains information about using the syntax checker. To minimize the testing, create a single GRSRNLxx member, test it, and then copy it to the Parmlib of all systems.

SYNCHRES

The Synchronous RESERVE (SYNCHRES) feature was added to GRS in OS/390 2.7. The SYNCHRES option allows an installation to specify whether the system should obtain a hardware RESERVE for a device prior to returning control to the program after an ISGENQ or RESERVE request.

If SYNCHRES(NO) is specified, the RESERVE request is appended to the first subsequent I/O request to that resource. If there is a delay between when the RESERVE is issued and when the first I/O is initiated, there is a window when another work unit could manage to get the RESERVE for the target device. This introduces the possibility of a “deadly embrace”, which was sometimes addressed by IPLing the system in question. For this reason, you should specify SYNCHRES(YES).

Prior to z/OS1.6, the default for SYNCHRES was NO. This was changed in z/OS 1.6 to YES. The SYNCHRES feature is enabled or disabled independently on each system.

The SYNCHRES feature can be activated through either the GRSCNFxx Parmlib member or the **`SETGRS`** operator command.

While the use of this feature is not mandatory, it is recommended to prevent deadlocks when reserving volumes.

Notes:

- Starting with z/OS 1.6, the default value for SYNCHRES is YES. This is a change from previous releases of z/OS, where the default value for SYNCHRES is NO.
- SYNCHRES=YES might increase RESERVE request time.
- When SYNCHRES=YES, callers of the RESERVE macro might encounter an ABEND with ABEND code 738, reason code 0001 if the reserve channel command word (CCW) fails because of an I/O failure device.

For more information see the following:

- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS System Commands,* SA22-7627

2.2.2 Dynamic GRS exits

z/OS 1.2 introduced two new GRS exits, ISGNQXIT and ISGNQXITFAST, which are called prior to RNL processing. These exit routines affect the way ISGENQ, ENQ, DEQ, and RESERVE requests are handled. They replace the ISGGREX0 interface. Unlike ISGGREX0, they can be refreshed without an IPL.

Note: Use care when reloading these exits because, unlike in RNL processing, GRS does not enforce serialization integrity across exit reloads. This introduces the possibility of a data integrity exposure during the window until all systems are using the new level of the exit.

For best performance, use ISGNQXITFAST rather than ISGNQXIT. The ISGNQXIT exit should only be used if the ISGNQXITFAST exit cannot be used. Under no circumstances should both exits be used, as both will be called on every ENQ, resulting in degraded system performance.

IBM introduced two further dynamic exits in z/OS 1.4 plus APAR OW53323. These exits, ISGCNFXITSYSTEM (for SYSTEM-scope resources) and ISGCNFXITSYSPLEX (for SYSTEMS-scope resources) provide a way for the installation to suppress specific ENF 51 signals that GRS would otherwise issue to notify programs of contention. See 2.2.5, "Contention management" on page 35 for implications of excessive ENF 51 notifications.

For more information see the following:

- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS Installation Exits,* SA22-7593

2.2.3 ANALYZE command

Contention is often first detected when one user complains that a job is not moving. You display the job and find that it is waiting on a data set. So you display the data set and find that it is held by another job. You display that job and find that it is hung, waiting on a different data set. And on and on you go until eventually you reach the top of the chain. Although this process works, it is very time-consuming.

To make this process more efficient, the GRS ANALYZE command was introduced in OS/390 2.8. This command provides a sysplex-wide list of the units of work involved in contention for GRS-managed resources.

This display can focus on the units of work that are waiting for the resources, or those holding the resources. In addition, the installation can display dependencies between requesters of GRS-managed resources. The scope of the analysis is based on the input specified by the command issuer, and can be:

- The entire sysplex
- One system
- One address space
- One task

The default scope for the analysis is the entire sysplex.

The ability of the GRS ANALYZE command to provide sysplex-wide information means that in deadlock scenarios, an informed decision as to which system or entity is causing the problem can be made and appropriate action taken (cancelling a task, for example). Prior to having the ability to easily diagnose complex contention problems, customers often IPLed a system in the hope of resolving the problem. Now, such an IPL can be avoided.

For more information see the following:

- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS System Commands,* SA22-7627

2.2.4 Enhanced Contention Analysis Service (ISGECA)

An API equivalent to the `D GRS,ANALYZE` command was introduced with z/OS 1.5. Known invokers of this function include System Automation 2.1 and above, and z/OS msys for Operations.

Customers can write their own utility programs utilizing the ISGECA macro to obtain waiter and blocker information for GRS-managed resources. Used in a similar manner to the GRS ANALYZE command, it can provide valuable information when deadlocks occur that enable the contention to be addressed without the need to IPL.

ISGECA returns one of two types of information for a relevant GRS resource:

- WAITER: the longest-waiting unit of work for that resource, and the top (longest) blocking unit of work for that waiter. General information about the resource and the numbers of resource owners and waiters is also reported.
- BLOCKER: the longest -locking unit of work for that resource. General information about the resource and the number of resource owners and waiters is also reported.

A GRS resource is considered relevant to an ISGECA request if that resource currently has waiters and blockers associated with it. GRS resource waiter/blocker information can be obtained for a specific system within the current sysplex, or for all the systems operating in the current sysplex.

ISGECA reports on relevant resources as they are encountered in the system's GRS resource management data infrastructure. The order of reported resources in the output area is unpredictable, and implies no suggestion of one resource having greater waiter/blocker considerations than any other reported on resource.

For more information see the following:

- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS Authorized Assembler Services Reference ENF-IXG,* SA22-7610

2.2.5 Contention management

Resource contention can result in less important work holding resources that are needed by more important work. To help prevent this, ENQ and Latch contention processing issue an SRM ENQHOLD SYSEVENT to promote the work holding the resource. The installation uses the enqueue residence value (ERV) on the IEAOPTxx Parmlib member to specify the number of CPU service units that an address space or an enclave is to be promoted for. During this time, the work unit runs with increased dispatching priority.

The default ERV value is 500 service units. However, on a 2094-701 there are nearly 30000 service units per second, meaning that the default ERV value will result in the work unit being promoted for just 1/600th of a second. Therefore, in order to increase the likelihood of the work unit releasing the resource within the promotion period, increase the ERV value to a larger value like 10000.

Also related to resource contention is the enqueue contention reporting provided by products like RMF. GRS issues an Event Notification (ENF51) to provide resource contention information. ENF processing schedules an SRB to the listener’s address space, which branches to the listener’s exit routine. RMF (and some non-IBM products) listen for the ENF51 event. If RMF is set to a low dispatching priority in a busy system, the SRBs can build up while waiting to be dispatched. Because the SRBs reside in SQA, this can result in storage shortages.

Two exits, ISGCNFXITSYSTEM and ISGCNFXITSYSPLEX, can be used to filter the resources that contention is reported for, thereby relieving this situation. See 2.2.2, “Dynamic GRS exits” on page 33 for further details.

Note: Run RMF at a high dispatching priority to avoid encountering these storage shortages.

2.2.6 Contention Notification System enhancement

Resource contention can result in poor system performance, and long-term contention can indicate that programs are being “starved” or even that a deadlock condition exists. GRS provides operator commands and APIs such as ISGECA, GQSCAN, and ISGQUERY to query for contention information. It also issues ENF 51 signals to notify monitoring programs to track contention for resources.

In GRS Ring mode, the passing of the GRS RSA message around the ring results in each system in the GRS complex knowing about the complex-wide ENQs. This provides each system with the information it needs to produce a sysplex-wide enqueue contention report.

However, in Star mode, each system only knows the ENQs that are issued by itself. In order to ensure that every system has a sysplex-wide view of the contention information, all ENQ contention events are sent to one system (known as the Contention Notification System, or CNS), which then issues a sysplex-wide ENF 51.

The first system in the sysplex to be IPLed will be the CNS. The CNS role can become CPU-intensive such that it dominates a particular image. Alternatively, the image it resides on may not be receiving enough CP resource to be able to keep up with the incoming requests.

This can result in delays or XCF buffer problems in other systems in the sysplex. There may be particular systems that naturally generate more contention due to their workloads, and that would benefit by having the CNS run locally to remove cross-system communication to a remote CNS.

Prior to an enhancement in z/OS 1.8, the only way to move the CNS role was to IPL the system that is currently the CNS. This enhancement will allow you to control which is the CNS system by using a GRS system command.

The enhancement will be implemented with z/OS 1.8 and rolled back to z/OS 1.7 via an enhancement APAR OA11382. Note that this APAR must be installed on *all* systems in the sysplex before the command can be used. If the sysplex contains systems running at a level lower than z/OS 1.6, the command cannot be issued until all systems have been upgraded.

This enhancement provides two availability benefits:

- System outages caused by delays or XCF buffer shortages may be avoided.
- It is no longer necessary to IPL a system to move the CNS function.

The syntax of the new command is:

```
SETGRS CNS=<sysname>,[np]
```

For more information see the following:

- APAR OA11382
- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600

2.2.7 IBM Health Checker

The objective of the IBM Health Checker for z/OS is to identify potential problems before they affect system availability or cause outages.

Customize the IBM Health Checker for z/OS and run it on a regular basis to provide early detection of potential problems. The level of Health Checker shipped with z/OS 1.7 (and available for download for releases back to z/OS 1.4) includes checks specifically for GRS. Run these checks on a regular basis, and act on any exceptions that may be reported.

For more information see the following:

- “Using IBM Health Checker for z/OS and Sysplex” in the introduction chapter of *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *IBM Health Checker for z/OS: User’s Guide,* SA22-7994

3

Communications Server

This chapter describes changes to reduce the need for planned outages or subsystem restarts made in the Communications Server component of OS/390 and z/OS.

3.1 Communications Server enhancements checklist

Table 3-1 contains a checklist of availability-related features in Communications Server, with a brief description and information about which release of z/OS the feature was delivered in.

Table 3-1 Communications Server checklist

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
VTAM Maintenance without re-IPLing MVS	1.4	It is possible to activate most VTAM service without an IPL. See 3.1.1, "VTAM Maintenance without re-IPL" on page 39.	
SMF exits	1.4	TCP/IP SMF user exits are no longer supported. See 3.1.2, "SMF exits" on page 39.	
VTAM and TCP/IP startup recommendations	1.4	Changing the VTAM Enterprise Extender HOSTNAME dynamically. See 3.1.3, "VTAM and TCP/IP startup recommendations" on page 39.	
TCP/IP CTRACE	1.5	CTRACE buffer size can now be changed using `TRACE CT` command. See 3.1.4, "TCP/IP CTRACE enhancements" on page 40.	
Activating RESOLVER changes	1.4	Dynamic updates to RESOLVER using `MODIFY RESOLVER,REFRESH` command. See 3.1.5, "RESOLVER address space" on page 40.	
TCP/IP Dynamic Reconfiguration	1.5	`VARY TCPIP,,OBEYFILE` command. See 3.1.6, "TCP/IP Dynamic Reconfiguration" on page 41.	
TCP/IP Dynamic VIPA (DVIPA)	1.4	DVIPA function improves VIPA takeover. See 3.1.7, "TCP/IP Dynamic Virtual IP Addresses (DVIPA)" on page 42.	
Telnet in its own address Space	1.6	Telnet can be started as part of the TCP/IP stack, or it can run in its own address space separate from the stack. See 3.1.8, "Telnet in its own address space" on page 43.	
VMCF and TNF restartable address spaces	1.4	You can configure Virtual Machine Communication Facility (VMCF) and TNF in two different ways: as restartable subsystems, or as non-restartable subsystems. See 3.1.9, "Virtual Machine Communication Facility (VMCF) and Terminal Notification Facility (TNF) restartable address spaces" on page 44.	

3.1.1 VTAM Maintenance without re-IPL

In OS/390 releases prior to OS/390 2.6, some VTAM modules resided in SYS1.LPALIB. Some other VTAM modules resided in the SYS1.NUCLEUS. To apply maintenance to either type of module, you had to IPL the system.

With Communications Server for OS/390 2.6, most of the modules that formerly resided in LPALIB and NUCLEUS now reside in a new data set called SYS1.SISTCLIB. During VTAM initialization, OS/390 now loads these modules into CSA and ECSA. Maintenance can be applied to these modules without a system IPL; however, you must restart VTAM to pick up the new service.

Note: Some modules associated with TSO/VTAM and VTAM common management information protocol (CMIP) services remain in LPALIB, and continue to be loaded into PLPA/EPLPA storage. Some new modules introduced in OS/390 2.6—those associated with common storage manager (CSM)—also reside in LPALIB. You must re-IPL the system to activate maintenance to these modules. When an IPL or VTAM restart is necessary, the PTF HOLDDATA instructions will contain this information.

For more information see:

- Chapter 4, “VTAM Maintenance Without re-IPLing MVS” in *z/OS Communications Server: SNA Planning and Migration Guide,* SC31-8622

3.1.2 SMF exits

CS for OS/390 2.5 removed support for SMF user exits in TCP/IP. Users with existing TCP/IP SMF exits were recommended to migrate their exits to use the standard SMF exit facilities (IEFU83, IEFU84, and IEFU85). Using standard SMF exit facilities provides the following benefits:

- Support for dynamic updating of the exits
- Support for all executing environments under which the TCP/IP stack is executing (SRB mode, Reentrant, Cross-memory mode, and so on)
- Avoidance of duplicate SMF exit processes

Note: FTP server SMF exits are still supported; however, we strongly recommend that customers migrate to the new Type 119 SMF records, and the FTP server exits are not supported for these.

For more information see:

- Chapter 4, “SMF Exits” in *z/OS Communications Server: IP Planning and Migration Guide,* SC31-8512 for OS/390 2.5
- Chapter 13, “SMFEXIT” in *z/OS Communications Server: IP Configuration Reference,* SC31-8776

3.1.3 VTAM and TCP/IP startup recommendations

The VTAM Enterprise Extender (EE) HOSTNAME parameter can be modified only if NODETYPE is specified during VTAM START processing.

While you can modify the EE HOSTNAME value dynamically, and displays of VTAM start options will show the new value immediately, the new value will not be *used* until all Enterprise

Extender lines, whose GROUP definition statements do not have EE HOSTNAME explicitly coded, are inactive. Any subsequent line activation from the Enterprise Extender XCA major node, whose GROUP definition statements do not have EE HOSTNAME explicitly coded, will make use of the new EE HOSTNAME start option value.

The IPADDR start option, if it is in effect at the time when the **`MODIFY VTAMOPTS,HOSTNAME=hostname`** is specified, will be reset (that is, set to a value of 0.0.0.0) as part of the MODIFY processing. The value NONE can be used to clear the setting of the EE HOSTNAME start option.

The EE HOSTNAME and IPADDR cannot be modified using one **`MODIFY VTAMOPTS`** command. If both start options are specified on the same **`MODIFY`** command, they will both be ignored and message IST1917I will be generated.

Also, as of OS/390 2.8, TCP/IP shares its Data Link Controls (DLCs) with VTAM. Therefore, usage of DLCs requires VTAM to be started before TCP/IP. If your VTAM goes down, TCP will detect this and remove itself from the IP sysplex. When restarting VTAM, TCP will automatically reconnect to the IP sysplex. A restart of TCP/IP does *no*t require a restart of VTAM.

For more information see:

- “MODIFY VTAMOPTS” in *z/OS Communications Server: SNA Operation,* SC31-8779
- “TCP/IP and VTAM Considerations” in *z/OS Communications Server: IP Planning and Migration Guide,* SC31-8512

3.1.4 TCP/IP CTRACE enhancements

After initialization, you must use the **`TRACE CT`** command to change the component trace options. Modifying options with the **`TRACE CT`** command can be done with or without the Parmlib member. As of z/OS 1.5 you can dynamically modify the component trace buffer size for the SYSTCPDA, SYSTCPIP, and SYSTCPIS components.

For more information see:

- “Modifying options with the TRACE CT command” in *z/OS Communications Server: IP Diagnosis Guide,* SC31-8782

3.1.5 RESOLVER address space

Prior to z/OS 1.2, the RESOLVER function was implemented as part of the various socket Application Programming Interfaces (APIs) available on the z/OS platform. As a result, multiple versions of the RESOLVER function were available, one for each type of socket API supporting RESOLVER calls. All of these RESOLVER libraries were quite similar in their support of RESOLVER functions, but had slight differences from administrative and configuration perspectives.

In z/OS 1.2, the various RESOLVER libraries supported by the TCP/IP and LE APIs are now consolidated into a single RESOLVER component. This allows consistent name resolution processing across all applications using the TCP/IP and LE socket APIs. The new consolidated RESOLVER is automatically enabled on z/OS 1.2 and requires a new system address space that is automatically started during UNIX System Services initialization.

The consolidated RESOLVER offers several enhancements over previous releases:

- You can now specify TCPIP.DATA statements that will be used regardless of the application's environment or the socket API the application is using. This allows installations to specify TCPIP.DATA statements in single location for the entire operating system image and prevent end users from being able to override these specifications. This support is provided by a new resolver setup statement, GLOBALTCPIPDATA.

- You can now specify the final search location of where TCPIP.DATA statements are found by using a new resolver setup statement, DEFAULTTCPIPDATA.
- The RESOLVER address space now allows changes to the TCPIP.data file in long-running jobs. You can refresh the RESOLVER address space from the operator's console using the **`MODIFY RESOLVER,REFRESH`** command. REFRESH causes applications to have their TCPIP.DATA information, including local host tables (for example, etc/hosts, etc/ipnodes, HOSTS.SITEINFO, HOSTS.ADDRINFO, or ETC.IPNODES information), updated on their next RESOLVER request after the REFRESH occurs.

  TCPIP.DATA statements control how (and if) the RESOLVER uses name servers.

When z/OS UNIX starts the RESOLVER address space, it starts it with SUB=MSTR so that JES is not required to be active. This means that JES services are not available to the RESOLVER address space. Therefore, no DD cards with SYSOUT can be used.

For more information see:

- “MODIFY command—RESOLVER Address Space” in *z/OS Communications Server IP System Administrator’s Commands,* SC31-8781
- “Running a Started Task Under the Master Subsystem” in *z/OS V1R7.0 MVS JCL Reference,* SA22-7597

3.1.6 TCP/IP Dynamic Reconfiguration

If you wish to make changes to the TCP/IP configuration, you can use the **`VARY TCPIP,,OBEYFILE`** command to make temporary dynamic changes to the system operation and network configuration without stopping and restarting the TCP/IP address space.

You can dynamically change many of the TCP/IP configuration options defined in the PROFILE.TCPIP data set. To do this, put the changed configuration statements in a separate data set and process it with the **`VARY TCPIP,,OBEYFILE`** command.

In addition to changes to the stack, there are three sets of parameters in the `PROFILE` data set for Telnet that you can edit and customize. Changes to these parameters can be activated using the **`OBEYFILE`** command. Because the TN3270 server functions are more typically in a separate address space, specify that address space name when issuing the **`OBEYFILE`** command.

The statements involved are:

- The stand-alone PORT statement with the INTCLIEN keyword (optional)
- TELNETPARMS (INTERNALCLIENTPARMS) information block for defining PROFILE statements relating to Telnet port setup
- The BEGINVTAM information block, for defining PROFILE statements relating to the VTAM interface.

Changes activated via the **`OBEYFILE`** command are in effect until the TCP/IP cataloged procedure is started again or until another **`VARY OBEYFILE`** overrides them. You can maintain different data sets that contain a subset of the TCP/IP configuration statements and activate them while TCP/IP is running.

Note: If you attempt to edit PROFILE.TCPIP while TCP/IP is active, and PROFILE.TCPIP is defined in the TCP/IP PROC as a sequential data set, the message: `data set in use` might be displayed. To avoid this, specify FREE=CLOSE in the TCP/IP JCL as follows:

```
//PROFILE  DD DISP=SHR,DSNAME=TCPIP.PROFILE.TCPIP,FREE=CLOSE
```

This allows you to edit the profile while TCP/IP is active. Typically, when TCP/IP starts, it keeps the PROFILE allocated and does not release the allocation until the end of the step (in this case, the end of the job). If you specify FREE=CLOSE, the release occurs once the data set is read. MVS releases the enqueue on the PROFILE, which allows you to edit it.

If the PROFILE is a member of a PDS, FREE=CLOSE is not needed.

For more information see:

- “VARY TCPIP,,OBEYFILE” in *z/OS Communications Server: IP Configuration Reference,* SC31-8776
- “Configuration overview” in *z/OS Communications Server: IP Configuration Guide,* SC31-8775
- “VARY TCPIP” in *z/OS Communications Server IP System Administrator’s Commands,* SC31-8781

3.1.7 TCP/IP Dynamic Virtual IP Addresses (DVIPA)

The capability of VIPA takeover is improved with the introduction of Dynamic Virtual IP Addresses (DVIPA) and Distributed Dynamic Virtual IP Address (distributed DVIPA). The DVIPA function improves VIPA takeover by allowing systems programmers to plan for system outages and provide for backup systems to take over without operator intervention or external automation.

The distributed DVIPA function extends the availability benefits by allowing the connections for a single DVIPA to be serviced by applications on several stacks. This adds the benefit of limiting the scope of an application or stack failure, while also providing enhanced workload balancing.

In general, z/OS configured with VIPA provides the following advantages:

- Automatic and transparent recovery from device and adapter failure. When a device (for example, a channel-attached router) or adapter (for example, an OSA-Express adapter) fails, if there is another device or link that provides the alternate paths to the destination:
  - IP will detect the failure, find an alternate path for each network, and route outbound traffic to hosts and routers on those networks via alternate paths.
  - Inbound and outbound traffic will not need to reestablish the active TCP connections that were using the failed device.
  - For connection requests originating at a z/OS TCP/IP stack, tolerance of device and adapter failures can be achieved by using source VIPA addressing.
- Recovery from z/OS TCP/IP stack failure (where an alternate z/OS TCP/IP stack has the necessary redundancy). Assuming that an alternate stack is defined as a backup, the use of VIPAs enables the backup stack to activate the VIPA address of the failed stack. Connections on the failed primary stack will be disrupted, but they can be reestablished on the backup using the same IP address as they were using previously. In addition, the

temporarily reassigned VIPA address can be restored to the primary stack after the failure has been resolved.

- Limited scope of a stack or application failure. If a DVIPA is distributed among several stacks, the failure of only one stack affects only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.
- Enhanced workload management through distribution of connection requests. With a single DVIPA being serviced by multiple stacks, connection requests and associated workloads can be spread across multiple z/OS images according to Workload Manager (WLM) and Service Level Agreement policies (for example, the TCP/IP Quality of Service agent).
- Allows the non-disruptive movement of an application server to another stack so that workload can be drained from a system in preparation for a planned outage.

DVIPA takeover is possible when a DVIPA is configured as active (using VIPADEFINE) on one stack and as backup (using VIPABACKUP) on another stack within the sysplex. When the stack on which the DVIPA is active terminates or leaves the sysplex group, the backup stack automatically activates the DVIPA and notifies the routing daemon.

Note: In order for DVIPA takeover to be useful, the applications that service the DVIPA addresses must be available on the backup stacks. In the absence of the application, the DVIPA will be active, but client connections to the application will still fail. If OMPROUTE is used, configure GLOBALCONFIG SYSPLEXMONITOR DELAYJOIN. This causes DVIPA takeover to be delayed until OMPROUTE is active and able to advertise DVIPAs on the takeover stack. Note that dynamic routing is *not* required in conjunction with DVIPA.

Starting with z/OS 1.6, a new set of autonomic functions are added to the TCP/IP sysplex support. These functions (which must be explicitly enabled using the new SYSPLEXMONITOR option on the GLOBALCONFIG parameter) allow TCP/IP to automatically proactively react to problems in the sysplex. For example, if one TCP/IP finds that a critical resource is being depleted, it may remove itself from the TCP/IP sysplex group. This causes work to be routed to the other members of the sysplex and potentially avoids a crash of this TCP/IP (which could result if the resource shortage was not addressed).

Starting in z/OS 1.8, Communications Server will provide checks that run under the z/OS Health Checker infrastructure.

For more information see:

- *z/OS Communications Server: New Function Summary,* GC31-8771
- “Introduction to VIPAs and Using Dynamic VIPAs (DVIPAs)” in *z/OS Communications Server: IP Configuration Guide,* SC31-8775

3.1.8 Telnet in its own address space

As of z/OS 1.6, you can optionally run Telnet in its own address space, separate from the TCP/IP stack. Prior to this, Telnet was started as part of the TCP/IP stack. The advantages of running Telnet in its own address space include the following:

- Telnet priority can be set to a different priority than that of TCP/IP.
- Telnet can be stopped and restarted without stopping TCP/IP.
- Separating Telnet and TCP/IP makes problem diagnosis easier.
- You can start multiple instances of Telnet.

Note that the default continues to be to run Telnet in the TCP/IP address space, so if you wish to exploit this capability, you must explicitly update the PORT statement for TELNET to specify NOAUTOLOG, and you must set up the required JCL for the TELNET address space. IBM has announced that the ability run Telnet in the TCP/IP stack address space will be removed in a future release of z/OS.

For more information see:

- “Telnet in its own address space” in *z/OS Communications Server: IP Configuration Guide,* SC31-8775

3.1.9 Virtual Machine Communication Facility (VMCF) and Terminal Notification Facility (TNF) restartable address spaces

Virtual Machine Communication Facility (VMCF) and Terminal Notification Facility (TNF) are two address spaces that manage inter-address space communication for TCP/IP. These address spaces contain the names of the VMCF and TNF users. From OS/390 2.6, these address spaces are optionally restartable, enabling you to modify VMCF and TNF tables without having to re-IPL MVS. Configuring VMCF and TNF as restartable subsystems has the following advantages:

- Error detection is provided when the subsystems do not seem to be initializing properly.
- You can change the system name on the restart.
- Commands are available to remove users from internal tables, display current users, and terminate the subsystem.

If you choose to use restartable VMCF and TNF, follow these steps:

1. Update your IEFSSNxx member in Parmlib with the TNF and VMCF subsystems.
2. Add procedure EZAZSSI to your system PROCLIB.
3. Start VMCF and TNF using the procedure EZAZSSI before starting TCP/IP.

A restartable VMCF and TNF configuration provides better availability and is recommended.

Note: These functions, if used improperly, can cause unpredictable side effects for TCP/IP and its server and client applications. These side effects can include hangs, server restart loops, and abends. They could also require the restart of TCP/IP and its applications as well as the restart of VMCF, TNF, or both. The following are some common VMCF problems:

- VMCF or TNF fail to initialize with an 0C4 abend.

  This is probably an installation problem; check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.

- Abends 0D5 and 0D6 after removing a user with REMOVE.

  This probably occurs because the application is still running and using VMCF. To remove users from VMCF or TNF, first terminate the affected user.

- VMCF or TNF do not respond to commands.

  This probably occurs because one or both of the non-restartable versions of VMCF or TNF are still active. To get them to respond to commands, stop all VMCF/TNF users, FORCE ARM VMCF and TNF, then use EZAZSSI to restart.

- VMCF or TNF cannot be stopped.

  This probably occurs because users still exist in the VMCF and TNF lists. Use the **`MODIFY VMCF,DISPLAY,NAME=*`** and **`MODIFY TNF,DISPLAY,NAME=*`** commands to identify users who are still active. Then either cancel those users or remove them from the lists using the **`MODIFY VMCF,REMOVE`** and **`MODIFY TNF,REMOVE`** commands.

- Address Space Identifiers (ASIDs) become non-reusable when VMCF and TNF address spaces are stopped or restarted.

  As VMCF and TNF address spaces provide PC-entered services that must be accessible to all address spaces, they each obtain a system LX. This causes the ASIDs associated with these address spaces to be non-reusable when these address spaces are terminated. If VMCF and TNF are terminated enough times, then all available ASIDs could be exhausted, preventing the creation of new address spaces on the system. In this case, an IPL will be required.

For more information see:

- “Planning your installation and migration, Configure VMCF and TNF” in *z/OS Communications Server: IP Configuration Guide,* SC31-8775

4

DFSMS

This chapter describes changes to reduce the need for planned outages made in the DFSMS component of z/OS.

4.1 DFSMS enhancements checklist

Table 4-1 contains a checklist of enhancements to the DFSMS components of the operating system.

Table 4-1 Checklist of outage avoidance items for BCP component of z/OS and OS/390

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
DFSMS CATALOG Abnormal End of Service Task	1.5	The **`MODIFY CATALOG,FORCE`** command allows you to abnormally end a service task even if it is in recall. See 4.1.1, “DFSMS CATALOG Abnormal End of Service Task” on page 49.	
Freeing resources held by Catalog Address Space	1.4	The **`MODIFY CATALOG,RESTART`**, **`CLOSE`**, and **`UNALLOCATE`** commands can be used to release resources held by Catalog Address Space and to resolve numerous CATALOG-related problems. See 4.1.2, “Freeing resources held by the Catalog Address Space” on page 50.	
DFSMS SYMREC	1.4	New DFSMS symptom records written to LOGREC. See 4.1.3, “DFSMS SYMREC creation” on page 51.	
Dynamically changing CATALOG attributes set at IPL time	1.4	The **`MODIFY CATALOG`** command allows you to alter a number of attributes that are initialized at IPL time. See 4.1.4, “Dynamically changing CATALOG attributes set at IPL time” on page 52.	
DFSMS CATALOG GDS setting	1.5	You can change the GDS setting for reclaim processing as needed without having to IPL. See 4.1.5, “DFSMS CATALOG GDS setting” on page 52.	
DFSMS CATALOG usage threshold	1.4	The **`MODIFY CATALOG,NOTIFYEXTENT(xxx)`** command can be used to set the CATALOG notify extent threshold. See 4.1.6, “DFSMS CATALOG usage threshold” on page 53.	
Restartable DFSMS PDSE address space	1.6	One of the PDSE address spaces is now restartable. See 4.1.7, “DFSMS Restartable PDSE address space” on page 54.	
DFSMS PDSE ANALYSIS and FREELATCH commands	1.4	The **`VARY SMS,PDSE,ANALYSIS`** and **`VARY SMS,PDSE,FREELATCH`** commands can help you identify and resolve PDSE problems. See 4.1.8, “DFSMS PDSE ANALYSIS and FREELATCH commands” on page 56.	
DFSMS PDSE issues with high CPU	1.6	APAR OA11068 changes the default to not enable PDSE caching when in z/Architecture® mode. See 4.1.9, “DFSMS PDSE issues with high CPU” on page 56.	

Features/Enhancements	Introduced in z/OS release	Description	Exploited?
DFSMS non-SMS-managed PDSE	1.4	It is no longer a requirement for PDSEs to be SMS-managed. See 4.1.10, "DFSMS non-SMS-managed PDSEs" on page 57.	
DFSMShsm Common Recall Queue	1.4	It is possible to have a common recall queue for DFSMShsm running on multiple z/OS systems. See 4.1.11, "DFSMShsm Common Recall Queue" on page 58	
DFSMShsm secondary host promotion	1.4	If the current primary HSM is stopped or fails, an alternate HSM can take over the primary responsibilities. See 4.1.12, "DFSMShsm secondary host promotion" on page 58.	
Various space-related enhancements in SMS	1.4	Numerous enhancements were provided to improve automated reaction to data set space problems. See 4.1.13, "DFSMS space problem enhancements" on page 59.	
Rename ENQed data set	1.4	OS/390 2.10 added the ability to rename an ENQed duplicate data set - see STGADMIN.DPDSRN RACF profile. See 4.1.14, "Rename ENQed data sets" on page 59.	
OAM Sysplex support	1.4	OAM sysplex support allows you to take down the system that owns an OAM library without that impacting the applications using that library. See 4.1.15, "Object Access Method (OAM) sysplex support" on page 60.	

4.1.1 DFSMS CATALOG Abnormal End of Service Task

Within the CATALOG address space (CAS), there are service tasks that perform services on behalf of users. These service tasks on occasion may become hung. Originally, the DFSMS **MODIFY CATALOG** command allowed users to end a service task abnormally using the END or ABEND commands.

However, if the active service task was waiting for processing in the user address space to proceed, the system did not abnormally end the task. When a service task is waiting for processing in the user address space to finish, it is in a state defined as *recall* in catalog terminology. The **MODIFY CATALOG,END(id),FORCE** command allows you to end abnormally the service task even if it is in recall.

You would normally become aware of a hung service task as a result of an ENQ being held by CAS for an abnormally long time. The appropriate action to take depends on how you want to proceed. The considerations, and the related CATALOG commands, are described in "Ending a Catalog Request Task" in *z/OS DFSMS Managing Catalogs,* SC26-7409.

For more information see:

- Chapter 8, “Modify Catalog Command Syntax” in *z/OS DFSMS Managing Catalogs,* SC26-7409.
- Chapter 5, “Termination of service task in recall” in IBM Redbook *z/OS V1R3 and V1R5 DFSMS Technical Guide,* SG24-6979.

4.1.2 Freeing resources held by the Catalog Address Space

If you need to free a resource held by CATALOG, there are two situations: either the catalog you wish to work on is open in the Catalog Address Space for normal processing, or there is some problem within the Catalog Address Space, resulting in the resource being held for longer than normal. In this section we address the normal situation first, and then discuss the error situation.

In normal processing, the Catalog Address Space will allocate and open catalogs until the maximum number of concurrently open catalogs (the default is 1024) is reached. If you wish to perform some action on a catalog that is open to the Catalog Address Space, you can use the **`MODIFY CATALOG,UNALLOCATE`** command to release the ENQ.

If you simply wish to change the attributes of a catalog (for example, its share options), issue the **`MODIFY CATALOG,CLOSE`** command and then make your change; the next time the catalog is opened by the Catalog Address Space, the new attributes will be loaded. The **`MODIFY CATALOG,CLOSE`** command allows you to close an Integrated Catalog Facility (ICF) catalog dynamically, without affecting any existing allocations. Thus, catalog attributes can be changed without cancelling jobs or performing a system IPL.

Closing or unallocating a catalog might also be useful when the control blocks for a catalog become corrupted. When the catalog is opened again, new control blocks are built for the catalog. The rebuilding of the control blocks is transparent to users of the catalog.

The Catalog Address Space is designed with internal checks that allow it to identify, and subsequently rebuild, damaged catalog control blocks. However, not all problems can be automatically identified. If you find that attempts to access a particular catalog are resulting in recurrent abnormal endings, rebuild the control blocks in the Catalog Address Space. This should be done using the **`CLOSE`** or **`UNALLOCATE`** parameters. If the damaged control blocks are for a VVDS, use **`VCLOSE`** or **`VUNALLOCATE`**.

Each of these parameters causes the control blocks to be released and rebuilt on a subsequent request that tries to access the catalog. Which parameter you use depends on whether you want the catalog unallocated and closed (**`UNALLOCATE`**), or closed but left allocated (**`CLOSE`**). When rebuilding the control blocks for a VVDS, use **`VCLOSE`** whenever possible. With **`VCLOSE`**, you specify the particular VVDS whose control blocks you suspect are in error. **`VUNALLOCATE`** can also be used, but this parameter unallocates *all* VVDSs. You cannot unallocate a specific VVDS.

The **`UNALLOCATE`** parameter can be used to unallocate all catalogs at once. It can also be used to unallocate a catalog on a volume that you need to vary offline. Use **`VUNALLOCATE`** if you need to also unallocate a VVDS to get a volume offline.

If the catalog resource is being held for a reason other than normal processing, the **`MODIFY CATALOG,RESTART`** command can be used to resolve the following problems:

- Inability to vary a volume containing a catalog offline, and a **`MODIFY CATALOG,CLOSE`** command is not successful in releasing the volume.

- ABENDs in the Catalog Address Space that relates to lack of storage (such as x'878' ABENDs).
- ABENDs in the Catalog Address Space that might indicate damage to control blocks (such as repeated 0C4 ABENDs at the same location).
- ENQ lockouts, particularly on the SYSIGGV2 resource, when the `MODIFY CATALOG,ABEND(id)` command will not remove the offending task.
- Activating catalog maintenance to load module IGG0CLX0 to correct a problem when an IPL is not necessary or feasible.

A cancel or other abnormal termination of a job while processing a catalog request may not complete cleanup processing in the Catalog Address Space due to an ENQ lockout. This could cause other users who need the same catalog to be delayed until the situation is cleared.

The condition can be easily cleared by using `D GRS,C` in conjunction with `MODIFY CATALOG,LIST` to identify the catalog service task that holds the ENQ that is blocking other tasks, and then issuing a `MODIFY CATALOG,ABEND(id)` command to terminate that service task. Alternatively, a `MODIFY CATALOG,RESTART` will correct the error non-disruptively. See APAR OA02940 for further details.

Note: Do not use `RESTART` to refresh catalog or VVDS control blocks or to change catalog characteristics—other catalog `MODIFY` commands are provided to accomplish this on a catalog-by-catalog basis. There is a risk that the Catalog Address Space restart might fail for some unanticipated reason. If this occurs, it will be necessary to IPL the system to recover the address space. However, a restart failure is a very unlikely occurrence.

It is also possible that a catalog request that is currently being processed cannot be properly retried after being interrupted by these commands. For example, this could occur with a DEFINE of a VSAM data set that is partially completed at the time of the restart. It is recommended that you try to quiesce system activity as much as feasible before doing a restart, or at least attempt to minimize the use of catalogs at the time the command is issued.

For more information refer to:

- Chapter 8, "Modify Catalog Command Syntax" in *z/OS DFSMS Managing Catalogs,* SC26-7409.

4.1.3 DFSMS SYMREC creation

There are times when the catalog return and reason code are not enough to identify a problem. SYMREC records on these occasions may provide more specific data about the problem and allow a diagnosis of the cause. In z/OS 1.3, SYMREC infrastructure was added to CAS. This should reduce the amount of time required to debug the problem. These SYMREC records are written to LOGREC.

Prior to this enhancement in z/OS 1.3, these records were written as part of the SVC dump when one is generated by CATALOG. By writing the SYMREC record to just LOGREC instead, it will save some time and processing. This feature is enabled by default in z/OS 1.5, although it can be enabled and disabled using the `MODIFY CATALOG,ENABLE/DISABLE(SYMREC)` command.

For more information, refer to:

- Chapter 14, “Detail edit report for a software record” in *z/OS MVS Diagnosis: Tools and Service Aids,* GA22-7589
- Chapter 8, “Modify Catalog Command Syntax” in *z/OS DFSMS Managing Catalogs,* SC26-7409

4.1.4 Dynamically changing CATALOG attributes set at IPL time

The **MODIFY CATALOG** command allows you to alter a number of attributes that are initialized at IPL time. Many of these attributes are set by the SYSCATLG member of SYS1.NUCLEUS, or preferably using the SYSCAT parameter in the LOADxx member, but can be overridden by the **MODIFY** command without an IPL.

The attributes that are defined in either the SYSCATLG member or the SYSCAT parameter are:

- Master catalog name
- Volume serial number of the volume containing the Master catalog
- The number of qualification levels for catalog aliases
- The Catalog Address Space service task lower limit
- Whether SYS% to SYS1 conversion is active

The first two items require an IPL to change. However, all the other attributes can be changed dynamically using the **MODIFY CATALOG** command.

The number of qualification levels for catalog aliases can be changed using the **MODIFY CATALOG,ALIASLEVEL** command. Obviously this command should be used with care as it could potentially result in a different catalog being used to catalog or find a data set.

The service task lower limit can be changed dynamically using the **MODIFY CATALOG,TASKMAX** command. Refer to “Changing the Maximum Number of Catalogs and Tasks in CAS” in *z/OS DFSMS Managing Catalogs,* SC26-7409.

The use of SYS% to SYS1 conversion can be controlled using the **MODIFY CATALOG,{SYS%ON|SYS%OFF}** command.

For more information see:

- Chapter 8, “Modify Catalog Command Syntax” in *z/OS DFSMS Managing Catalogs,* SC26-7409.

4.1.5 DFSMS CATALOG GDS setting

In some situations, a new generation (+1) of an SMS-managed GDS data set is created by a job, but is not rolled in. Any subsequent job that attempts to create a new (+1) generation will cause SMS to automatically go into a process that is called *GDS reclaim processing*. GDS reclaim processing, in effect, reuses a GDS in a deferred roll-in state.

This reuse can overwrite and destroy the data in a new generation, created by the first job, that is left in a deferred roll-in state. If a subsequent job reclaims it, the data in it will be overwritten, resulting in data loss. The automatic GDS reclaim processing is generally the correct action if the original abending job is rerun. However, there is a likelihood of data loss if a different job using that data set is run before the failed job is rerun. Some installations prefer this automatic GDS reclaim processing not to occur.

DFSMS in z/OS 1.5 introduced an enhancement that allows you to turn off automatic GDS reclaim processing. This is done with a new keyword GDS_RECLAIM in the IGDSMSxx member. The GDS_RECLAIM keyword may be set to YES or NO. The default value is YES.

You can change the GDS setting for reclaim processing as needed without having to IPL the system by specifying the keyword setting for GDS_RECLAIM(NO) in the IGDSMSxx member of Parmlib. You can then use the `SET SMS=xx` command to change the IGDSMSxx member being used. If you do not wish to use this method, you can alternatively use the `SETSMS GDS_RECLAIM {YES|NO}` command to change the value.

If you turn off reclaim processing, manual means must be used to delete the generation, rename it, or roll it in.

Notes:

- The setting of this parameter is independent on each system. Changing the setting and issuing the SETSMS command only affects the system the command is issued on. The changed value is not retained across an IPL unless you update the IGDSMSxx member that will be used at the next IPL. Installations that have different levels of DFSMS or different settings for this function on different systems should not share GDSes across those systems.
- The `SET SMS` command initializes SMS parameters and starts SMS if it has been defined but not started at IPL time, and changes SMS parameters if SMS is already running. The `SETSMS` command changes SMS parameters only when SMS is running.

For more information see:

- Chapter 2, “Using Catalogs with the Storage Management Subsystem” in *z/OS DFSMS Managing Catalogs,* SC26-7409.

4.1.6 DFSMS CATALOG usage threshold

Previously, when a catalog filled up the last extent, an error occurred on the next attempt to add a new record to the catalog. Those errors caused serious problems ranging from failed applications to a complete sysplex shutdown being required. In order to give you advance warning of a potential catalog-full condition, DFSMS now includes a function that issues a warning message (IEC361I) any time the highest extent of a catalog reaches a specified threshold.

You can specify the threshold using the DFSMS `MODIFY CATALOG,NOTIFYEXTENT(xxx)` command. The threshold applies to all catalogs; you cannot specify a different threshold for different catalogs. However, while not recommended, you can disable the function completely by setting the threshold to 0 or 100. The default threshold setting is 80%.

This warning message reduces the likelihood of system outages because catalogs otherwise could become full without warning. We recommend that you have a documented and tested procedure in place for handling a catalog-full situation. In particular, you should closely monitor your Master catalog because addressing space problems in that catalog can be so disruptive to the whole sysplex.

For more information, refer to:

- Chapter 8, “Modify Catalog Command Syntax” in *z/OS DFSMS Managing Catalogs,* SC26-7409
- “Full Catalog Recovery Procedure” in *z/OS DFSMS Managing Catalogs,* SC26-7409

4.1.7 DFSMS Restartable PDSE address space

Tip: If you make significant use of PDSE data sets, we strongly recommend that you read the current version of IBM Redbook *Partitioned Data Set Extended Usage Guide,* SG24-6106-01. This book contains information that is vital if you are to successfully manage and get the best value from a large number of PDSE data sets.

In the past, it was necessary to re-IPL one or possibly even more systems to resolve a hang condition, deadlock condition, or an out-of-storage condition in the Partitioned Data Set Extended (PDSE) address space.

DFSMS in z/OS 1.3 introduced a new address space, SMSPDSE, and moved a lot of PDSE-related control blocks from common storage and the SMXC and SYSBMAS address spaces into this new address space. Then in z/OS 1.6, a new, restartable, address space, SMSPDSE1, was introduced. The two address spaces are used as follows:

- SMSPDSE - this address space continues to be non-restartable. All processing that took place in the SMXC and SYSBMAS system address spaces now occurs in SMSPDSE.
  - This reduces excessive ECSA usage by moving control blocks from common storage into the SMSPDSE address space.
  - It reduces re-IPLs due to system hangs (in failure or CANCEL situations).
  - It provides storage administrators with tools for monitoring and diagnosis (for example, determining which systems are using a particular PDSE).
- SMSPDSE1 - this new address space is restartable.
  - This new address space allows you to recover from many PDSE problems without having to re-IPL a system or systems.
  - It helps you determine which systems require a re-IPL or repair to resolve PDSE hangs and failures.

SMSPDSE1 is a restartable address space that provides connections to, and processes requests for, those PDSE data sets that are not part of the global connections associated with SMSPDSE. Global connections are those made to PDSE data sets that are in the LNKLIST data set concatenation.

Note that the default will continue to be to have just a single address space (SMSPDSE); to create the SMSPDSE1 address space in a sysplex coupled systems environment, you must explicitly specify the following options in the IGDSMSxx member of Parmlib:

```
PDSESHARING(EXTENDED|NORMAL) Specify PDSESHARING(EXTENDED)
PDSE_RESTARTABLE_AS(YES|NO) Specify PDSE_RESTARTABLE_AS(YES)
```

You can move from PDSESHARING NORMAL mode to EXTENDED mode dynamically. However, you cannot move from EXTENDED back to NORMAL mode without a sysplex IPL. Therefore we recommend that you carry out sufficient testing of this function in a test sysplex prior to implementing it in a production sysplex.

Note: APAR OA07897 implemented a change relating to starting SMSPDSE1 for the first time. Previously, this could be achieved by changing the IGDSMSxx member, activating the change, and issuing a command to start the SMSPDSE1 address space. However, after the APAR, you cannot start this address space with a command unless the address space had been previously running. You must now update the IGDSMS member to specify that you want this new address space, and then IPL.

LNKLST and LLA considerations

PDSE data sets may be used in the LINKLIST, and the LLA will manage their directories in much the same way as for a standard PDS. This means that for data sets that are not changed for the life of an IPL, there are no significant considerations. However, when a data set is in the LINKLIST, and the data set might be modified from another system in the GRS complex, it is essential that the GRS parameters are correctly set up across all members of the complex. Failure to ensure that the GRS parameters are correctly set up could result in damage to the PDSE.

When PDSESHARING(EXTENDED) is specified, it is assumed that GRS is correctly set up and that all systems sharing the PDSEs are in the same sysplex. We recommend that PDSESHARING(EXTENDED) be specified even if the SMSPDSE1 address space is not being used because EXTENDED enables users to share read and write access to PDSEs across systems in the sysplex. Remember, however, that you can only use this function if you are certain that the PDSE will only be used by members of the sysplex.

Important: Adhere to the following PDSE sharing rules to avoid data corruption:

- If you specified PDSESHARING(NORMAL), do not share PDSE data sets beyond the scope of the GRS complex.
- If you specified PDSESHARING(EXTENDED), do not share the PDSE data sets beyond the scope of the sysplex.

We recommend that PDSE_RESTARTABLE(YES) be specified to obtain the advantages available with the restartable SMSPDSE1 address space. PDSESHARING(EXTENDED) must be specified to be able to implement SMSPDSE1. However, because PDSESHARING(EXTENDED) cannot be used if you share the PDSE outside the sysplex, that means that you also cannot use the restartable SMSPDSE1 address space if you share PDSEs outside the sysplex.

Management of the LNKLST members that are allocated at IPL time or using the SETPROG LNKLST command is done by the SMSPDSE address space, which is not restartable. As a result, any problems relating to PDSE data sets in the LNKLST concatenation could result in a requirement for an IPL.

There is also a restriction relating to deleting a PDSE that was in the LNKLST. Connections are made to PDSEs in the LNKLST (these are called global connections) and these are not released until a subsequent IPL, even if the LNKLST is updated to remove the PDSE library. Therefore, you cannot delete a PDSE that was associated with the system LNKLST. If you wish to delete such a PDSE, then ensure that it is not included in your PROGxx LNKLST specification for your next IPL, and delete it after that subsequent IPL. You *can* rename a PDSE that was in the system LNKLST; however, the global connections to that newly-renamed PDSE remain, and as mentioned, you cannot delete it until a subsequent IPL.

Notes:

- If neither of the PDSE_RESTARTABLE_AS or PDSESHARING parameters are specified, or are allowed to default, then only the non-restartable address space SMSPDSE will be initiated and activated.
- As soon as you implement extended PDSESHARING, ensure that all members of the sysplex have their IGDSMS members updated accordingly.
- You can implement the extended PDSE sharing dynamically with an IPL, but reverting to normal mode would require a sysplex IPL. Implementing the restartable SMSPDSE1 address space for the first time requires an IPL of each system where you use this capability.
- Before restarting the SMSPDSE1 address space, you should be aware this action could result in failures of currently running jobs and TSO sessions that are accessing PDSEs.

For more information, refer to:

- Chapter 15, "Restartable PDSE1 address space" in *z/OS DFSMSdfp Diagnosis,* GY27-7618
- IBM Redbook *Partitioned Data Set Extended Usage Guide,* SG24-6106-01

4.1.8 DFSMS PDSE ANALYSIS and FREELATCH commands

Two new commands were added to help in the analysis and repair of some PDSE errors:

- `VARY SMS,PDSE|PDSE1,ANALYSIS<,DSNAME(..)<,VOLSER(..)>><,RETRIES(..)>`
- `VARY SMS,PDSE|PDSE1,FREELATCH(........,...,........)`

Issue the **ANALYSIS** command only when you suspect that one or more users or jobs are having problems accessing a PDSE.

If you suspect that a PDSE is failing, issue the **ANALYSIS** command from each system where the PDSE has been reported as failing. Do not issue the **FREELATCH** command without preceding it with an **ANALYSIS** command.

Note: If the **FREELATCH** command is used to release a latch held by a process that was still running, it could result in the breakage of the SMSPDSE or SMSPDSE1 address spaces. The latch is not released unless it is held by the ASID, tcbaddr, indicated in the command. The latch is released only if it is held by the same user for each of the retries.

For more information see:

- Chapter 15, "Freelatch command and Analysis command" in *z/OS DFSMSdfp Diagnosis,* GY27-7618.
- IBM Redbook *Partitioned Data Set Extended Usage Guide,* SG24-6106-01

4.1.9 DFSMS PDSE issues with high CPU

APAR OA11068 changes the default to *not* enable caching for z/Architecture, by changing the default to HSP_SIZE=0 for z/Architecture systems. This parameter specifies the size of the Hiperspace™ (in megabytes) that is used for the SMSPDSE and the restartable SMSPDSE1 address spaces.

The installation can enable caching by changing the HSP_SIZE to a non-zero value. This requires an IPL for all releases except z/OS 1.6 or later, in which it is possible to restart the PDSE restartable address space if this feature is enabled.

For z/OS 1.6 each of the two address spaces, SMSPDSE and SMSPDSE1, need their own specification for HSP_SIZE.

Note: Although the impact of inappropriate specification of the Hiperspace size is less than before the SMSPDSE and SMSPDSE1 address spaces were implemented, care is still required to avoid overloading the real and auxiliary storage managers.

4.1.10 DFSMS non-SMS-managed PDSEs

Although SMS-management is beneficial and recommended for most user data sets, it is no longer a requirement for PDSEs. You can define PDSEs on non-SMS-managed volumes. This helps to simplify certain PDSE maintenance procedures and also provides for PDS and PDSE common processing. In particular, this change eases the maintenance of multiple z/OS images:

- You do not need to catalog non-SMS-managed PDSEs. This makes it easier for PDSEs to reside on system residence volumes. You can also clone the volume since the catalog requirement is removed. Keep in mind, however, that DFSMShsm cannot process uncataloged PDSEs.
- You can also use indirect cataloging and extended indirect cataloging.
- Because of the removal of the catalog requirement, you can now have more than one PDSE with the same name which enables you to concurrently access multiple PDSEs with the same data set name.
- You can use DFSMSdss to logically dump and restore the SYSRES volume containing PDSEs.

This function is available by PTF for DFSMS/MVS 1.4 and DFSMS/MVS 1.5 systems. Additional PTFs are required for BCP Allocation, TSO/E, and ISPF for full function support.

To determine if a lower-level system can use non-SMS-managed PDSEs, you can test the new DFAUPDSE bit in the DFA (pointed to by CVTDFA and mapped by the IHADFA macro). This indicates whether full support is available for creating, deleting and copying PDSEs. It does not guarantee, however, that the correct TSO/E and ISPF function is installed.

On sharing systems that do not have the equivalent function installed, you can expect the following:

- Non-SMS-managed PDSEs can be opened and modified after they are created on a system that has the function.
- ISPF and ISMF will not display correct data set properties.
- DFSMSdss and DFSMShsm cannot process the data sets.

Note:

1. DFSMShsm cannot process uncataloged PDSE data sets.
2. If SMS is inactive when you are allocating a PDSE using the TSO/E ALLOCATE command with the LIKE operand, you must add DSNTYPE(LIBRARY) to the ALLOCATE command, or a PDS will be created instead.

For more information refer to:

- “Summary of Changes” in *z/OS DFSMS Migration,* GC26-7398

4.1.11 DFSMShsm Common Recall Queue

Prior to z/OS 1.3, every HSM maintained its own, in-storage, queue of data set recall requests. This meant that if HSM, or the system it is running, on goes down before a recall request has been processed, the request is lost and must be reissued when HSM is restarted.

Also, because each HSM could only process its own requests, each HSM needed physical access to any resources required by the recall request. So for example, if data set A.B was migrated to tape X12345, the system that is recalling that data set would need access to both an appropriate tape drive and also to that specific tape.

In z/OS 1.3, HSM introduced the option of having a single HSM recall queue that could be shared between multiple HSMs. This means that if tape X12345 is currently mounted for a recall on SYSB, and a user on SYSA issues a recall for a data set on that tape, then the recall request could be processed by SYSB, avoiding the need for SYSA to have access to any tape resource, and potentially eliminating an additional tape mount.

More importantly, because there are now two copies of each recall queue (one in the local storage of the requesting HSM, and the other in the Coupling Facility structure), it is possible to restart a HSM or the system it is running on without losing any queued requests. In fact, you could even do a complete sysplex IPL, and as long as the CF is not restarted, the queued requests will still be there waiting to be processed when the HSMs are restarted.

There are other performance and load balancing benefits of Common Recall Queue, but these are the availability-related benefits of this feature.

For more information about HSM Common Recall Queue, refer to:

- IBM Redbook *z/OS V1R3 and V1R5 DFSMS Technical Guide,* SG24-6979
- “Common Recall Queue” in *DFSMShsm Storage Administration Guide,* SC35-0421

4.1.12 DFSMShsm secondary host promotion

There are certain functions that are only carried out by the HSM host that has been specified to be the primary host. If that host is down for some reason, those functions will not be performed.

OS/390 2.7 introduced the HSM secondary host promotion capability. Using this function, one or more alternative HSMs can be defined as potential backups in case the primary HSM is stopped or fails. HSM uses XCF communication to find out when the primary goes away. Each of the alternates will race to automatically take over as the new primary. Whichever one “wins” will carry out the primary responsibilities until the primary is restarted, at which point it automatically takes back the primary host role.

For more information about how to implement this function, refer to:

- “Secondary Host Promotion” in *DFSMShsm Implementation and Customization Guide,* SC35-0418
- IBM Redbook *DFSMS/MVS 1.5 Presentation Guide,* SG24-5336

4.1.13 DFSMS space problem enhancements

DFSMS in z/OS 1.3 and z/OS 1.5 added a number of enhancements in the area of handling and avoiding data set space-related problems. This section only lists these enhancements. For more details or implementation information, refer to the appropriate product manual.

- Dynamic Volume Count (DVC)

  DVC is a value that you can specify for a data class. If a data class contains a DVC value, and this value is greater than the current volume count for a data set, then the DVC is used to allow the data set to extend to additional volumes.

- Extend storage groups

  The extend storage group is a feature of the pool storage group type designed to be used for secondary allocation when there is insufficient space available in the primary storage group. Without this support, extend processing can only select volumes from within the same storage group as the initial allocation.

- Overflow storage groups

  Overflow storage groups are a new type of pool storage group that are used for the initial data set creation when there is insufficient space available in the primary storage group for the first extent of a data set. If there is an overflow storage group defined and the ACS™ routines allow the use of the overflow storage group for this data set, then a data set will be directed to the overflow storage group if all volumes in the primary storage groups are over their high allocation threshold.

- Automation assistance

  Previously, the routing of all SMS messages issued during data set allocation had been altered to only send messages to the joblog of the task allocating the data set. DFSMS in z/OS 1.3 will route SMS messages related to volume selection failures to syslog. DFSMS in z/OS 1.5 extends this by routing messages to syslog when SMS End-Of-Volume (EOV) processing is unable to allocate space on a new volume.

  DFSMS in z/OS 1.5 also adds the ability to issue console messages when a storage group exceeds its high threshold utilization.

- Multi-tiered storage groups

  SMS conventional volume selection will select the best-performing volume from any of the supplied storage groups. When multi-tiered storage groups are enabled by the storage administrator, SMS will honor the storage group sequence order as listed in the ACS storage group SET statement when selecting volumes for new allocations. SMS will attempt to allocate using a volume in the first listed group prior to attempting allocation in the next sequential and subsequent storage groups.

4.1.14 Rename ENQed data sets

In OS/390 2.10, DFSMS added the ability to rename ENQed data sets. This is a facility that is frequently required by systems programmers who may be working on a copy of a system data set, where system address spaces have an ENQ against the "live" version of the data set, but the systems programmer needs to delete and reallocate the copy. Because the ENQ is only on the data set name and not the data set name/volume serial number combination, it is not normally possible to proceed with the delete request.

Many installations created their own authorized routines to bypass the ENQ processing. However, DFSMS now provides an IBM-supported mechanism for achieving this objective. If you define a RACF FACILITY class profile called STGADMIN.DPDSRN, and grant the systems programmer the requisite level of access, it will be possible to rename these ENQed data sets. This can potentially avoid the disruption that may be associated with removing the ENQ from the live version of the data set.

For more information on this function, refer to:

- "DADSM rename of duplicate data sets" in IBM Redbook *DFSMS Release 10 Technical Update,* SG24-6120

4.1.15 Object Access Method (OAM) sysplex support

DFSMS 1.5 (OS/390 2.7) introduced sysplex support for Object Access Method. Previously, the OAM library could only be online and accessible to one system in the sysplex. This support provided the ability to use OAM function shipping between members of the OAMPlex. So a job running on SYSB could access an OAM object in a library that was owned by OAM on SYSA.

Further, if SYSA goes down for some reason (for a planned IPL, for example), one of the other members of the OAMPlex would automatically take over ownership of the library, informing the remaining members of the sysplex that requests for objects in that library should now be shipped to this OAM. As a result, it is possible to have an outage, planned or unplanned, of the system that owns the OAM library *without* that resulting in an outage of the applications that access that data.

For more information on implementing and using the OAM sysplex support, refer to:

- IBM Redbook *DFSMS/MVS 1.5 Presentation Guide,* SG24-5336
- *DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support,* SC35-0426

5 Time change

This chapter discusses changes and recommendations that need to be reviewed for the time changes in the spring and fall for most z/OS software components. The clock typically moves forward one hour for the spring time change, and back one hour for the fall time change.

In this chapter, we use the term Universal Time, Coordinated (UTC), which replaces the previously used term Greenwich Mean Time (GMT). The difference between the two is that UTC is adjusted for leap seconds.

For a detailed discussion about the role of the Sysplex Timer® in relation to time changes, we highly recommend IBM Redbook *S/390 Time Management and IBM 9037 Sysplex Timer,* SG24-2070.

5.1 Time change

Traditionally, most installations would IPL their systems to adjust for the semi-annual clock change. However, because outages have become more difficult to schedule, installations are now giving consideration to whether it is actually necessary to IPL. The concerns are more apparent in the fall time change due to the potential of duplicate time stamped records or automated timer events that may occur twice. In the spring, the potential of missing log data or missed automated timer events is not as critical, but still needs to be reviewed.

5.1.1 Adjusting z/OS time

IBM recommends that the Time of Day (TOD) clock should always be set to the UTC value. UTC is the one time value that remains constant across the globe. The TOD clock is set either via the HMC for single systems, or preferably via the Sysplex Timer for sysplex systems. This results in a simpler procedure to schedule a change to the local time offset twice a year.

Conversely, if the TOD clock is set equal to local time, then you will be required to shut down and IPL to make both spring and fall time changes (except, of course, if your local time never changes, or your time zone is the UTC time zone). If you are in this scenario, you may consider graduated changes to the offset. Otherwise, the outage to make the change will be at least as long as the offset you have currently to UTC.

If you do not have a Sysplex Timer, or if you specified ETRZONE=NO in your CLOCKxx member, you can use the new `SET TIMEZONE` command (introduced with z/OS 1.7) to make a daylight saving change. Use this command instead of the `SET CLOCK` command that people would traditionally have used to adjust the time in configurations like this.

Note: In order to avoid an IPL during the time changes, your software products and applications should be reviewed for UTC and local time dependency. Each product may need to be recycled or kept down for an hour to avoid duplication of time stamps or apparent gaps in logs. If the number of these products is more than can be easily managed, then an IPL may be the preferred option.

Review all releases of products on your system to determine support. Some require them to be down or recycled during the time change. See Table 5-1 for a list of z/OS components and their respective time change considerations.

Note that all the major IBM subsystems (CICS, DB2, IMS, and so on) use the UTC time in their logs. Therefore, as long as the UTC value is not changed, the subsystems themselves should have no problem, regardless of changes to the local offset.

However, if the applications running in those subsystems use local rather than UTC time, you need to determine whether they can handle the local time changing dynamically.

Table 5-1 z/OS components and time change considerations

Component	Time change considerations
EREP and SYSLOG	1 hour of repeated time stamps in the fall and 1 missing hour in the spring.
RMF	There will be a time gap in the spring and potential loss of data in the fall. For information about how to avoid the loss of RMF data in the fall, refer to 5.1.2, “RMF” on page 63.

Component	Time change considerations
C/C++ and Language Environment® for MVS	Review and remove the usage of the usermod EDCLLOCL from SCEESAMP. See 5.1.3, “C/C++ and Language Environment for MVS” on page 64 for further information.
Integrated Call Level Interface (ICLI) as a started task	This is a USS application that uses USS system calls to retrieve the current time. If the ICLI server is a started task and you use UTC for log time stamps, then the started task will need to be recycled with the new TZ parm in iclienv. For more information, see *SAP R/3 on DB2 UDB for OS/390 and z/OS Planning Guide*.
UNIX System Services	The time zone for USS is based on environment variable TZ and will need to be reviewed. For more information, see 5.1.4, “UNIX System Services” on page 64.
Tivoli® Workload Scheduler (TWS) - formerly OPC	As of TWS 8.2, the controller and tracker no longer need to be restarted when the local time offset changes. See APAR PK06007 for more information. Previous releases of TWS require a recycle of TWS during the time change.
IBM Tivoli Omegamon products	Action is required for all the Omegamon started tasks. See 5.1.5, “IBM Tivoli Omegamon” on page 64 for details.
CICS	Supported as of CICS/TS 1.3.0. Requires an `EXEC CICS RESETTIME` to synchronize its local time to the MVS TOD clock.
DB2	DB2 supports the time change. However, applications that run on DB2 may be affected if they use local time rather than UTC time.
IMS	IMS V6 and above support the time change. However, some IMS applications may be affected if they use local time.

For more information on handling the time change, refer to:

- IBM Redbook *S/390 Time Management and IBM 9037 Sysplex Timer,* SG24-2070
- *IBM 9037 Sysplex Timer and System/390 Time Management,* GG66-3264 (which is only available by ordering as hardcopy at the time of writing)

5.1.2 RMF

There are some considerations for RMF, depending on whether the local time is being moved forward or backward:

- In the spring, there is a gap in local time where no data appears to be collected.
- In the fall, data is collected twice with the same local stamp. For the RMF SMF records, this simply means that you will have records with duplicate timestamps. However, for RMF Monitor III, the data will be overwritten.

  So if this data is essential for monitoring and you want to avoid any loss of data, the data set needs to be removed before the time changes. This can be done by using:

  ```
  RMFGAT (DS(DEL(dsname))
  ```

  The data is archived and can be used with the Monitor III reporter when allocated as RMFDS00 in a TSO session. For more details, see “*Data Set allocation*” in *z/OS V1R6.0 RMF User’s Guide,* GC33-7059.

5.1.3 C/C++ and Language Environment for MVS

Review and remove the usage of the usermod EDCLLOCL from CEE.SCEESAMP as referenced in *z/OS: C/C++ Language Environment Customization,* SA22-7564.

Important: Do *not* install the EDCLLOCL usermod. The default C/C++ locale (EDC$370) will by default obtain the time zone difference from UTC from the system. If your C/C++ application requires a different time zone than the one obtained from the system, you can use the tzset() and the TZ environment variable described in *z/OS: C/C++ Run-Time Library Reference***.**

In previous releases, the EDCLLOCL usermod would have been used to change the C/C++ locale time information for your site. It would also need to be updated if the weeks for the time change weekend were modified.

5.1.4 UNIX System Services

All commands assume that times stored in the file system and returned by the operating system are stored using UTC. The mapping from the universal reference time to local time is specified by the TZ (time zone) environment variable found in /etc/profile.

The syntax of the parameter is:

```
TZ= standardHH[:MM[:SS]] [daylight[HH[:MM[:SS:]]]
[,startdate[/starttime],enddate[/endtime]]]
```

where startdate is of the form Jnnn for Julian date or Mm.w.d for month/week/day.

An example of the TZ parameter is:

```
TZ=EDT5EST,M3.5.0/2,M11.1.0/2
```

This translates to a time change at 2:00 am on the 5th, or last, Sunday in March and another change on the first Sunday in November.

Possible action required: After recent U.S. Legislation, the default values may not be appropriate in your installation. The values default to M4.1.0/2:00:00,M10.5.0/2:00:00. This translates to the standard American Daylight Saving Time rules starting at 02:00 am on the first Sunday in April, and ending at 02:00 am on the last Sunday in October.

5.1.5 IBM Tivoli Omegamon

The following actions must be taken for any IBM Tivoli Omegamon products during the time changes.

All Omegamon historical collector started tasks must be recycled in the spring and then kept down for at least 1 hour in the fall. This is to avoid VSAM errors due to duplicated time records.

All other Omegamon started tasks must have their time reset via the following command:

```
   MODIFY OMSTCx, TIME RESET
```

6

Security Server

This chapter describes changes to the Security Server component of z/OS that reduce the need for planned outages.

6.1 Security Server enhancements checklist

Table 6-1 contains a checklist of IPL avoidance features in Security Server with a brief description and which release of z/OS introduced it.

Table 6-1 Security Server checklist

Features/Enhancements	Introduced in z/OS release	Description	Completed
RACF DATABASE SWITCH	1.4	**RVARY SWITCH** command usage. See 6.1.1, “RACF DATABASE SWITCH” on page 66.	
RACF Dynamic CDT support	1.6	The Dynamic CDT provides the means to add, change, and delete installation-defined classes. See 6.1.2, “RACF Dynamic Class Descriptor Table (CDT) support” on page 67.	
RACF Dynamic Template enhancements	1.5	Enhancements to Dynamic Template in z/OS 1.5 reduce system outages and simplify the installation of any future RACF releases. See 6.1.3, “RACF Dynamic Template enhancements” on page 68.	
RACF Router Table removal	1.6	New support to eliminate the need to update it. See 6.1.4, “RACF Dynamic Router Table removal” on page 69.	
RACF STARTED class	1.4	Use the STARTED class to add or modify RACF identities for started procedures. See 6.1.5, “RACF STARTED Class” on page 69.	
Restarting the RACF subsystem	1.4	Use the **RESTART** operator command to restart a specified function in the RACF subsystem. See 6.1.6, “Restarting the RACF subsystem” on page 70.	
Stopping the RACF subsystem address space	1.4	If you are using RRSF, the RACF STOP command allows you to stop the RACF subsystem address space with a minimal loss of remote requests. See 6.1.7, “Stopping the RACF subsystem address space” on page 70.	
Changing the size of the RACF CF structures	1.4	RACF does not support the use of SETXCF ALTER to change the size of its CF structures, and issuing a REBUILD against an active structure can result in abends in RACF requests. But there is a way to dynamically change the structure sizes. See 6.1.8, “Changing size of RACF CF structures” on page 70.	
RACF RNL Health Check	1.6	Health Checker reports on incorrectly specified RACF RNLs. See 6.1.9, “RACF RNL Health Check” on page 71.	

6.1.1 RACF DATABASE SWITCH

The **RVARY SWITCH** deactivates and deallocates the current primary database and causes RACF to use the backup copy as the new primary without having to re-IPL your system. This command also allows you to restore the RACF database from a copy on tape, or to recatalog the database on another volume. The backup database must be specified in the data set name table (ICHRDSNT). The **RVARY ACTIVE** command is used to reactivate a database, followed by **RVARY SWITCH**, which reallocates the database.

If you deactivate the primary RACF database data set and delete or uncatalog it, and then replace it with a new data set, the new data set must be cataloged and have the same name as the original database before you can activate it. To avoid the single point of failure associated with running just a single RACF database, repair the original primary and reactivate it at the earliest opportunity.

When an **`RVARY SWITCH`** command is issued, the database buffers are also switched. When RACF is *not* enabled for sysplex communication, it associates a set of buffers with the new primary database (the original backup database) and disassociates the buffers from the original primary database (the new backup database).

When RACF *is* enabled for sysplex communication and an **`RVARY SWITCH`** command is issued, RACF switches the buffers by associating the larger set of buffers with the new primary database (the original backup database) and the smaller set of buffers with the new backup database (the original primary database). After the **`RVARY SWITCH`**, the Coupling Facility structures associated with the original primary are associated with the new primary, and the ones associated with the original backup are associated with the new backup.

Notes:

- While you can deactivate and then make changes to either the primary or backup RACF databases, you *cannot* change the names of the databases (defined in the ICHRDSNT module) without an IPL.
- If an I/O error occurs on the RACF database, causing the device to be varied offline, RACF automatically issues an **`RVARY SWITCH`** command to switch to the backup database (assuming the backup is active and on a device that has not been varied offline).
- Before deleting or recataloging a database, you must first deactivate the database by issuing either the **`RVARY INACTIVE`** command or the **`RVARY SWITCH`** command.
- The **`RVARY SWITCH`** command will not activate an inactive database.

For more information see:

- “RVARY SWITCH command” in *z/OS Security Server RACF Security Administrator’s Guide,* SA22-7683.

6.1.2 RACF Dynamic Class Descriptor Table (CDT) support

The CDT contains information that directs the processing of general resources. RACF references the Class Descriptor Table whenever it receives a resource class name other than DATASET, USER, or GROUP.

The Class Descriptor Table has two parts:

- The static Class Descriptor Table

  If you make a change to the static Class Descriptor Table, you must re-IPL to have the change take effect on your system.

  The CDT consists of two load modules, ICHRRCDX and ICHRRCDE. ICHRRCDX contains the class entries supplied by IBM. Each class supplied by IBM is a CSECT in load module ICHRRCDX.

  Note: Do *not* delete or modify any of the class entries in ICHRRCDX. ICHRRCDE is an optional module that contains installation-defined class entries. You define classes in ICHRRCDE using the ICHERCDE macro.

- The dynamic Class Descriptor Table

  As of z/OS 1.6, the dynamic CDT provides the ability to add, change, and delete installation-defined classes in the CDT without IPLing the system. Rather than having to assemble and link a module containing the CDT information, a new RACF class, CDT, is used to hold the definitions of the installation-defined classes. This optional portion of the Class Descriptor Table contains installation-defined class entries built from the CDT general resource class.

  You can define classes in the dynamic Class Descriptor Table using RDEFINE and RALTER commands. If you make a change to the dynamic Class Descriptor Table, you must issue the **`SETROPTS RACLIST(CDT) REFRESH`** command to activate the change.
- The RACF Web site provides a REXX exec to translate your installation-defined CDT into a series of RDEFINE CDT commands to facilitate this migration. Look for this download at:

  `http://www.ibm.com/servers/eserver/zseries/zos/racf/goodies.html`

Notes:

- If you have applications or vendor products that use dynamic classes, they must use the **`RACROUTE REQUEST=STAT`** interface to process information for dynamic classes (for example, to check if a class is active). If your application or vendor product uses the RACSTAT macro or the RCVTCDTP pointer in the RCVT control block to locate a dynamic class, it will not work properly.
- IBM-defined classes in ICHRRCDX are not dynamic. If updates are made to this module, an IPL is still required.
- We recommend to migrate your static installation-defined classes to the dynamic CDT.
- Any updates made to the RACF Data Set Names Table (ICHRDSNT) or the RACF Range Table (ICHRRNG) require a sysplex IPL.

For more information see:

- “Administering the Dynamic Class Descriptor Table” in *z/OS Security Server RACF Security Administrator’s Guide,* SA22-7683
- “ICHERCDE macro” in *z/OS Security Server RACF Macros and Interfaces,* SA22-7682
- “The Class Descriptor Table” in *z/OS Security Server RACF System Programmer’s Guide,* SA22-7681

6.1.3 RACF Dynamic Template enhancements

Enhancements to the use of RACF database templates in z/OS 1.5 reduce system outages and simplify the installation of future RACF releases or PTFs that provide new versions of the templates. This includes the following support:

- Change the templates to contain a level indicator that makes it possible to compare two sets of templates and to determine easily which level of template has a later set of definitions.
- During initialization at IPL time, RACF will determine whether the database has the right level of templates or not. If not, RACF will ignore the templates on the database and automatically use the proper set of templates from a new CSECT, IRRTEMP2.
- When a PTF that changes the templates is applied to a live system, you can run IRRMIN00 and have RACF recognize the new templates without an IPL.

- Prevent installing a downlevel set of templates onto a RACF database.
- Prevent complete reinitialization of a RACF database if that database is in use on the system where IRRMIN00 is run.

Continue to run IRRMIN00 with PARM=UPDATE against each of your RACF data sets when upgrading RACF releases or when applying PTFs that update the templates. Failure to do so will result in a warning message from RACF at IPL time, and possible error processing from RACF Database Unload, IRRUT200 or BLKUPD, although the normal mainline RACF functions will work properly. Additionally, it is possible that some vendor utilities will not function correctly without the PARM=UPDATE processing occurring.

Additionally, these enhancements introduce a new required migration task.

For more information see:

- “Dynamic Template” in *z/OS Security Server RACF Migration V1R5,* GA22-7690.

6.1.4 RACF Dynamic Router Table removal

As of z/OS 1.6, the IBM-supplied portion of the RACF router table (ICHRFR0X) is removed, eliminating the need to provide an entry in the installation-defined router table (ICHRFR01) for every installation-defined class. Most installation-defined classes will not require any change to ICHRFR01. In previous releases, ICHRFR01 had to be updated (assembler code, assembly, link edit, and IPL) when an installation-defined class was being added.

For more information see:

- “The RACF router table” in *z/OS Security Server RACF System Programmer’s Guide,* SA22-7681.

6.1.5 RACF STARTED Class

As of MVS/ESA 5.1, it is no longer necessary to define Started Task names in the ICHRIN03 module. Instead, you should use the STARTED class to add or modify RACF identities for started procedures without having to re-IPL the system. You can modify the security definitions for started procedures dynamically using the RDEFINE, RALTER, and RLIST commands.

In effect, the STARTED class provides a dynamic started procedures table. You must still have a started procedures table (ICHRIN03) because RACF cannot be initialized if ICHRIN03 is not present, but you can use the default one provided with z/OS.

For performance and ease of migration, you may consider defining JES as a started procedure with the trusted attribute.

Note: When the STARTED class is active, RACF uses it before using the started procedures table, ICHRIN03. It overrides any matching entries in ICHRIN03.

For more information see:

- “Using Started procedures” in *z/OS Security Server RACF Security Administrator’s Guide,* SA22-7683.

6.1.6 Restarting the RACF subsystem

Tip: The RACF subsystem is a prerequisite for a number of the “newer” RACF capabilities. In addition to that, however, it also provides the ability to issue RACF commands from the console, which can be very useful if a RACF change inadvertently impacts your ability to logon to TSO. For this reason, everyone should start this subsystem.

You can use the `RESTART` command to recover from failures when the RACF subsystem is unable to recover automatically. You can also use the `RESTART` command to activate maintenance to one of the load modules associated with a restartable function. When you restart the associated function, the updated copy of the load module is made available, without the need to re-IPL or reinitialize the RACF subsystem address space. The exception is when you specify the NODE keyword, in which case no modules are reloaded. Refer to “Restarting a function in the RACF subsystem” in *z/OS Security Server RACF System Programmer’s Guide,* SA22-7681, for a mapping of load module names to functions.

The RACF address space is started as SUB=MSTR, which allows it to start even if JES is not started.

For more information see:

- “The RACF subsystem” in *z/OS Security Server RACF System Programmer’s Guide,* SA22-7681

6.1.7 Stopping the RACF subsystem address space

Like other system address spaces that are needed for basic system operation, the RACF subsystem address space runs non-cancellable. If you are not using the RACF remote sharing facility (RRSF), there is no need to stop the address space before system shutdown. Subsequent IPL or MVS START command processing rebuilds the address space for proper system usage.

However, if you *are* using RRSF, the RACF `STOP` command allows you to stop the RACF subsystem address space with a minimal loss of remote requests. Stopping the RACF subsystem address space any other way (for example, using FORCE) is not recommended, and might cause requests to be lost or the VSAM files for workspace data sets to be damaged.

If an RRSF node allows directed commands, password synchronization, or automatic direction, it is important that you stop the address space with the RACF `STOP` command during a system shutdown, after all users have logged off and all batch jobs have completed.

It is also important that you not stop (or FORCE) the address space while users and jobs are active. If you do, updates could be lost (resulting in passwords or profiles not being synchronized) and output from directed commands could be lost.

For more information see:

- “Stopping the RACF subsystem address space” in *z/OS Security Server RACF System Programmer’s Guide,* SA22-7681

6.1.8 Changing size of RACF CF structures

The RACF Coupling Facility structures hold the RACF database buffers, and can deliver significantly improved performance for RACF requests.

To determine the correct size for the RACF structures, use the CFSizer tool, available on the Web at:

http://www.ibm.com/servers/eserver/zseries/cfsizer/racf.html

However, RACF does not support the ability to alter the structure size using the **SETXCF START,ALTER** command. To increase the structure size, the structure size in the CFRM policy must be changed, the new policy started, and a **SETXCF START,REBUILD** command issued against the structure in question.

Keep in mind that rebuilding the structure while RACF is storing its buffers in the structure can result in RACF database requests failing. To get around this situation, temporarily switch to non-datasharing mode using the **RVARY NODATASHARE** command, and then issue the REBUILD command. When the REBUILD completes successfully (it should only take a few seconds), switch back to data sharing mode using the **RVARY DATASHARE** command.

For more information on the use of the RACF structures, refer to:

- IBM Redbook *S/390 Parallel Sysplex: Resource Sharing,* SG24-5666
- *z/OS Security Server RACF System Programmer's Guide,* SA22-7681

6.1.9 RACF RNL Health Check

IBM Health Checker for z/OS evaluates whether the RACF ENQ names are in either the installation system exclusion resource name list (SERNL) or the system inclusion resource name list (SIRNL), and examines the security characteristics of several system-critical data sets. The output of this check is a list of these data sets with exceptions flagged.

The RACF service team has debugged several customer problems and outages and found that the problem or outage was caused by a customer's RNL incorrectly changing the scope of a RACF serialization request.

Note: Converting RACF SYSTEMS ENQs to SYSTEM ENQs can result in corruption of the RACF database, potentially leading to system outages.

In general, you can apply service to existing Health Checker checks without an IPL. If you wish to add a new check, you need to restart the Health Checker address space. For more information see:

- "RACF checks (IBMRACF)" in *IBM Health Checker for z/OS: User's Guide,* SA22-7994

6.2 Security Server changes that still require an IPL

There are some changes that are of such a significant nature that they can only be affected by scheduling a concurrent (that is, *not* a rolling) shutdown and re-IPL of *all* the systems sharing the RACF databases. There are two Security Server changes that fall into this category, as described here.

Changes to the Data Set Names Table

This module defines the names of the RACF primary and backup databases. Obviously it is vital that all systems that are sharing the RACF database have a common view of the databases they are to use. To enable this common view, if you want to change the Data Set Name Table, you must stop all the systems that are sharing the set of databases, update the module, and restart all those systems. If only a subset of the systems in the sysplex are sharing the database, only those systems must be IPLed.

Changes to the RACF Range Table

This module specifies the keys that will be associated with each of the RACF databases. Obviously it is vital that all systems that are sharing the RACF database have a common view of which database data set they should use to retrieve or insert a given profile. To enable this common view, if you want to change the Range Table, you must stop all the systems that are sharing the set of databases, update the module, and restart all those systems. If only a subset of the systems in the sysplex are sharing the database, only those systems must be IPLed.

Because of the impact of changing the Range Table, we strongly recommend regular monitoring of the utilization (both from a space perspective and a performance perspective) of the RACF database data sets in order to have plenty of advance notice of the need to increase the number of RACF database data sets.

7

JES2

This chapter discusses the features and capabilities in JES2 that contribute to reducing the number of planned outages.

More information can be found in the JES2 product manuals. Attributes that cannot be changed dynamically are also noted for completeness of documentation.

For more information on any of the JES2 parameters, refer to the following JES2 manuals for your release of z/OS:

- *z/OS JES2 Initialization and Tuning Guide,* SA22-7532
- *z/OS JES2 Initialization and Tuning Reference,* SA22-7533
- *z/OS JES2 Commands,* SA22-7526

Note:

There are a number of useful SDSF modifications with z/OS 1.7 to support the JES2 Health Monitor and the associated $J commands.

In addition, there is a new Resource Monitor (RM) panel that shows information about JES2 resources, such as JOEs, JQEs and BERTs. From the panel, users can identify resource shortages and view the history for resources.

There are enhancements to the Lines and Nodes panels to exploit new support in z/OS 1.7 for JES2 NJE connections over TCP/IP.

For more information on any SDSF function or change, see *z/OS SDSF Operation and Customization,* SA22-7670.

7.1 Modifying JES2

You can change JES2 initialization parameter settings through six methods that vary in their impact to the JES2 member. The following list provides the hierarchy, beginning with the easiest and least disruptive method, and ending with the most difficult or disruptive method for the JES2 member:

1. **Operator command**

   JES2 provides commands that permit you to dynamically change nearly all the JES2 attributes that are defined in the JES2PARM member. This is the least disruptive way to make a change to JES2. However, you must remember to reflect your change back into the relevant JES2PARM member in order to ensure that the change is not lost the next time JES2 is started.

2. **Hot start**

   A hot start is performed when JES2 abnormally terminates (or is stopped using the $PJES2,ABEND command) and is subsequently restarted using PARM=WARM, and JES2 determines that an IPL has not occurred prior to restart.

   You can use a hot start when JES2 has stopped, but other jobs and started tasks have continued to function and have not experienced problems. When you hot start JES2, all address spaces continue to execute as if JES2 had never terminated. Functional subsystems will continue to process spooled output as long as possible before the hot start, and JES2 will reconnect to the functional subsystem (FSS) during the hot start. Hot starts do not affect other members in a multi-access spool configuration.

   Note that many parameters in the JES2PARM deck are ignored by JES2 when coming up with a hot start. For more information, refer to *z/OS JES2 Initialization and Tuning Reference,* SA22-7533.

3. **Quick start**

   JES2 performs a quick start when you start JES2 with PARM=WARM after a $PJES2 was used to stop JES2 on this member of the MAS. A quick start is also performed following an all-member warm start and JES2 determines that the job queue and job output table do not need to be updated. In this case, the member being started is not the first member being started in the MAS.

   Unlike a hot start, all parameters in the JES2PARM deck are processed during a quick start.

4. **Single-member warm start**

   A single-member warm start is performed when JES2 is started for the first time following an IPL and PARM=WARM is specified and other members of the configuration are active. The warm-starting member joins the active configuration and recovers only work in process on that member when it failed or was stopped.

5. **All-member warm start**

   An all-member warm start is performed if a warm start is specified by the operator and JES2 determines that no other members of the configuration are active, or there is only one member in the configuration. All in-process work in the MAS will be recovered. After an all-member warm start, other members entering the configuration for the first time will perform a quick start.

6. **Cold start**

   On a restart of a JES2 member, you should implement a cold start with caution, because all job data previously on the spool volumes is lost. No other member in a multi-access spool configuration can be active during a cold start, or JES2 will end with an error message.

   If you request a cold start of JES2 or have specified the FORMAT start option, an IPL must precede the execution of the MVS START(S JES2) command unless JES2 was previously stopped by a $P JES2 operator command.

Table 7-1 lists the statements used to initialize JES2, as well as whether each can be changed dynamically, and if so, how they can be changed dynamically.

Table 7-1 JES2 INIT statements

INIT statements	Available in z/OS	Dynamic	Description
APPL(jxxxxxxx))	1.4	Yes	Defines an SNA NJE application to JES2.
BADTRACK	1.4	Hot start	Specifies an address or range of addresses of defective spool volume tracks that JES2 is not to use.
BUFDEF	1.4	Yes	Defines the local JES2 buffers to be created.
CKPTDEF	1.4	Yes	Defines the JES2 checkpoint data set(s) and the checkpointing mode.
CKPTSPACE	1.4	Yes	Defines how much additional space is available for expanding the JES2 checkpoint record.
COMPACT	1.4	Hot start	Defines a compaction table for use in remote terminal communications.
CONDEF	1.4	Yes	Defines the JES2 console communication environment.
CONNECT	1.4	Yes	Specifies a static connection between the nodes identified.
DEBUG	1.4	Yes	Specifies whether debugging information is to be gathered by JES2 during its operation for use in testing.
DESTDEF	1.4	Yes	Defines how JES2 processing interprets and displays both job and SYSOUT destinations.
DESTID(jxxxxxxx)	1.4	Yes	Defines a destination name (mostly for end-user use, as on a JCL statement) for a remote terminal, another NJE node, or a local device.
D MODULE(jxxxxxxx)	1.4	Display only	Displays diagnostic information for specified JES2 assembly modules and installation exit assembly modules.
ESTBYTE	1.4	Yes	Specifies, in thousands of bytes, the default estimated output (SYSOUT) for a job at which the `BYTES EXCEEDED` message is issued, and the subsequent action taken.
ESTIME	1.4	Yes	Specifies, in minutes, the default elapsed estimated execution time for a job, the interval at which the `TIME EXCEEDED` message is issued, and whether the elapsed time job monitor feature is supported.
ESTLNCT	1.4	Yes	Specifies, in thousands of lines, the default estimated print line output for a job, the interval at which the `LINES EXCEEDED` message is issued, and the subsequent action taken.

INIT statements	Available in z/OS	Dynamic	Description
ESTPAGE	1.4	Yes	Specifies the default estimated page output (in logical pages) for a job, the interval at which the `PAGES EXCEEDED` message is issued, and the subsequent action taken.
ESTPUN	1.4	Yes	Specifies, in number of cards, the default estimated punch card output for a job, the interval at which the `CARDS EXCEEDED` message is issued, and the subsequent action taken.
EXIT(nnn)	1.4	Yes	Associates the exit points defined in JES2 with installation exit routines.
FSS(accccccc)	1.4	Yes	Specifies the functional subsystem for printers that are supported by an FSS (for example Print Services Facility™).
INCLUDE	1.4	Yes	Allows new initialization data sets to be processed.
INIT(nnn)	1.4	Yes	Specifies the characteristics of a JES2 logical initiator.
INITDEF	1.4	Single member warm start	Specifies the number of JES2 logical initiators to be defined.
INTRDR	1.4	Yes	Specifies the characteristics of all JES2 internal readers.
JOBCLASS(v) | STC | TSU	1.4	Yes	Specifies the characteristics associated with job classes, started tasks, and time sharing users.
JOBDEF	1.4	Yes	Specifies the characteristics that are assigned to jobs that enter the JES2 member.
JOBPRTY(n)	1.4	Yes	Specifies the relationship between job scheduling priorities and job execution time.
LINE(nnnn) - BSC	z/OS 1.4 Modified in 1.7	Yes	The LINE(nnnnn) statement specifies the characteristics of one teleprocessing line to be used during remote or network job entry (for BSC NJE and RJE terminals).
LINE(nnnnn) - NJE/RJE/ SNA	z/OS 1.4 Modified in 1.7	Yes	The LINE(nnnnn) statement specifies the characteristics of one teleprocessing line or logical line (for SNA RJE terminals) to be used during remote or network job entry.
LINE(nnnnn) - TCP/IP	z/OS 1.7	Yes	The LINE(nnnnn) statement specifies the characteristics of one teleprocessing line or logical line to be used during remote or network job entry.
L(nnnn).JT(m)	1.4	Yes	Specifies the characteristics for a job transmitter on an NJE line.
L(nnnn).ST(m)	1.4	Yes	Specifies the characteristics for a SYSOUT transmitter on a line defined for network job entry.
LOADMOD(jxxxxxxx)	1.4	Yes	Specifies the name of a load module of installation exit routines to be loaded.
LOGON(n)	1.4	Yes	Identifies JES2 as an application program to VTAM.
MASDEF	1.4	Yes	Defines the JES2 multi-access spool configuration.
MEMBER(n)	1.4	Single member warm start	Defines the members of a JES2 multi-access spool configuration.

INIT statements	Available in z/OS	Dynamic	Description
NAME	1.4	Yes	Specifies the module or control section to be modified through subsequent VER and REP initialization statements.
NETACCT	1.4	Single member warm start	Specifies a network account number and an associated local account number.
NETSRV(nnn)	1.7	Yes	Defines NJE over TCP/IP server address space.
NJEDEF	1.4	Yes	Defines the network job entry characteristics of this JES2 node.
NODE(nnnn)	1.4	Yes	Specifies the characteristics of the node to be defined.
OFF(n).JR	1.4	Yes	Specifies the characteristics of the offload job receiver associated with an individual offload device.
OFF(n).JT	1.4	Yes	Specifies the characteristics of the offload job transmitter associated with an individual offload device.
OFF(n).SR	1.4	Yes	Specifies the characteristics of the offload SYSOUT receiver associated with an individual offload device.
OFF(n).ST	1.4	Yes	Specifies the characteristics of the offload SYSOUT transmitter associated with an individual offload device.
OFFLOAD(n)	1.4	Yes	Specifies the characteristics of the logical offload device.
OPTSDEF	1.4	Hot start or all-member warm start	Defines the options that are currently in effect.
OUTCLASS(v)	1.4	Yes	Specifies the SYSOUT class characteristics for one or all output classes.
OUTDEF	1.4	Yes	Defines the job output characteristics of the JES2 member.
OUTPRTY(n)	1.4	Yes	Defines the association between the job output scheduling priorities and the quantity (records or pages) of output.
PCEDEF	1.4	Hot start	Specifies the definition for various JES2 processes.
PRINTDEF	1.4	Yes	Defines the JES2 print environment.
PROCLIB	1.4	Yes	Ensures that data sets specified can be allocated.
PRT(nnnn)	1.4	Yes	Specifies the characteristics of a local printer.
PRT(nnnn) - FSS-only	1.4 - 1.7	Yes	Specifies the characteristics of an FSS-only local printer.
PUNCHDEF	1.4	Single member warm start	Defines the JES2 punch environment.
PUN(nn)	1.4	Yes	Specifies the characteristics of a local card punch.
R(nnnn).PR(m)	1.4	Yes	Specifies the characteristics of a remote printer.
R(nnnn).PU(m)	1.4	Yes	Specifies the characteristics of a remote punch.
R(nnnn).RD(m)	1.4	Yes	Specifies the characteristics of a remote card reader.

INIT statements	Available in z/OS	Dynamic	Description
RDR(nn)	1.4	Yes	Specifies the characteristics of a local card reader.
RECVOPTS(type)	1.4	Yes	Specifies the error rate below which the operator will not be involved in the recovery process.
REDIRect(vvvvvvvv)	1.4	Yes	Specifies where JES2 directs the response to certain display commands entered at a console.
REP	1.4	Yes	Specifies replacement patches for JES2 modules during initialization.
REQJOBID	1.4	Yes	Describes attributes to be assigned to Request Jobid address spaces.
RMT(nnnn) BSC	1.4-1.7	Yes	Specifies the characteristics of a BSC remote terminal.
RMT(nnnn) SNA	1.4-1.7	Yes	Specifies the characteristics of a SNA remote terminal.
SMFDEF	1.4	Yes	Specifies the system management facilities (SMF) buffers to JES2.
SOCKET(vvvvvvvv)	1.7	Yes	Defines an IP address and port for NJE/TCP and the associated NJE node.
SPOOLDEF	1.4	Yes	Defines the JES2 spool environment.
SSI(nnn)	1.4	Yes	Specifies the characteristics associated with individual subsystem interface definitions.
SUBTDEF	1.4	Hot start	Specifies the number of general purpose subtasks you wish JES2 to attach during initialization.
TPDEF	1.4	Yes	Defines the JES2 teleprocessing environment.
TRACE(n)	1.4	Yes	Specifies whether a specific trace ID(s) is to be started.
TRACEDEF	1.4	Yes	Defines the JES2 trace environment.
VER	1.4	Yes	Specifies verification of replacement patches for JES2 modules during initialization.
ZAPJOB	1.4	Yes	Remove job structure from job queue.

There are very few remaining parameters in JES2 that cannot be changed either dynamically by JES2 commands or through hot, single-system, or all-system warm starts. Table 7-2 contains a list of JES2 INIT statements and the related parameters that still require a cold start.

Table 7-2 JES2 statements requiring cold starts

Init statement	Related parm or statement	Comment	Description
JOBDEF	JOBNUM	Cold start required to *decrease* the number of jobs.	Specifies the maximum number of jobs that can be in the JES2 job queue at any given time.
NJEDEF	NODENUM	Cold start is required to *decrease* the value.	Specifies the maximum number (1-32767) of nodes in the NJE network to which this member belongs, or a value greater than or equal to the highest numbered node in your system.
NJEDEF	OWNNODE	Cold start is required to change the OWNNODE number for a node.	Specifies the number (1-32767) of this node, where nnnn is an integer between 1 and the value specified in the NODENUM= parameter.
OUTDEF	JOENUM	Cold start is required to *decrease* the value.	Specifies the number (must be no greater than 500000) of job output elements (JOEs) to be generated.
SPOOLDEF	BUFSIZE	Cold start is required to either increase *or* decrease this value.	Specifies the size (1944-3992) in bytes of each JES2 buffer. If the value specified is not a multiple of 8, it is rounded up automatically.
SPOOLDEF	DSNAME	Cold start is required to change the name of the spool data sets.	Specifies a 1- to 44-character name (hyphens (-) may be included for any character except the first) which is to be used as the data set name of the JES2 spool data set.
SPOOLDEF	SPOOLNUM	If $ACTIVATE has been issued, SPOOLNUM can be increase with $TSPOOLDEF command. If the command has not been issued, a cold start is required to increase this value.	Specifies the number (1-253) of JES2 spool volumes.
SPOOLDEF	TGSIZE	Cold start to *decrease* the TGSIZE.	Specifies the number (1-255) of JES2 buffers to be contained in a track group.
SPOOLDEF	TGSPACE	Cold start to *decrease.*	WARN= specifies the percentage (0-100) of use of track groups at which the operator will be alerted through message $HASP050 JES2 RESOURCE SHORTAGE.
SPOOLDEF	TRKCELL	Cold start to change the track cell size.	Specifies the size (1-120) of a track cell in terms of spool buffers.
SPOOLDEF	VOLUME	Cold start to change the spool volume prefix.	Specifies the 4- to 5-character prefix assigned to JES2's spool volumes.

7.2 Dynamic PROCLIB

One of the more common causes of a JES2 restart is to update the PROCLIB concatenation being used by JES2. The introduction of dynamic PROCLIB support in z/OS 1.2 means that PROCLIB concatenations can be added, modified, and deleted without restarting JES2.

The PROCLIB(xxxxxxxx) statement in the JES2INIT deck defines a dynamic PROCLIB concatenation to be used during conversion processing for jobs on this member. These concatenations can be added, updated, or deleted through operator commands.

Note: PROCLIB statement processing only ensures that the data sets specified can be allocated. It does *not* ensure that they actually exist or can be opened and used as a PROCLIB data set. That processing occurs when the PROCLIB is used by a job during conversion processing.

You can optionally specify the UNCONDITIONAL parameter, which tells JES2 to ignore any invalid or broken data set in the PROCLIB concatenation. With static PROCLIB support, you would need to restart JES2 with a modified PROCLIB concatenation if one of the PROCLIB data sets became damaged or inaccessible.

The $ADD PROCLIB JES2 command can be used to define a dynamic PROCLIB concatenation to be used during conversion processing for jobs on this member. Dynamic PROCLIB can override PROCxx DDs in the JES2 start PROC, but cannot alter or delete them.

$DEL PROCLIB(xxxxxxxx) can be used to delete a dynamic PROCLIB concatenation.

Table 7-3 lists and describes parameters that can be changed after IPL with a **`$T PROCLIB`** command.

Table 7-3 PROCLIB statement parameter list

Parameter	Related parm or statement	Command	Description
xxxxxxxx	N/A	$T PROCLIB	Specifies the 1- to 8-character PROCLIB DD name being defined.
DD(nnn)	N/A	$T PROCLIB	Specifies up to 255 data sets to be concatenated to this PROCLIB DD name.
DSName	N/A	$T PROCLIB	Specifies a 1- to 44-character data set name which JES2 will include in this PROCLIB concatenation.
UNIT	N/A	$T PROCLIB	If the PROCLIB data set to be used is not cataloged, then you must specify the unit information for the device containing the data set.
VOLser	N/A	$T PROCLIB	If the PROCLIB data set to be used is not cataloged, then this specifies a 1- to 6-character volume serial number on which the data set resides.
UNCONDitional I CONDitional	N/A	$T PROCLIB	Specifies what action JES2 should take if one of the data sets cannot be allocated.

Parameter	Related parm or statement	Command	Description
NAME	N/A	$T PROCLIB	Intended mostly for the $T command, NAME= allows the name of a PROCLIB concatenation to be changed.

7.3 ZAPJOB

Use ZAPJOB to remove all traces of a job structure from the JES2 job queue. It is intended to be used in situations where a job cannot be removed using normal JES2 commands or by a JES2 restart. Before using ZAPJOB for a job, ensure that the job and any output it might have created are not active in any JES2 process or device.

The ZAPJOB initialization statement can only be entered from CONSOLE mode. This prevents you from forgetting a ZAPJOB statement that you placed into the initialization data, thereby impacting jobs on future JES2 starts. See *z/OS JES2 Initialization and Tuning Guide,* SA22-7532 for information on how to enter CONSOLE mode during initialization.

Table 7-4 lists and describes parameters that can be specified on the `$ZAPJOB` command.

Table 7-4 ZAPJOB statement parameter list

Parameter	Related parm or statement	Command	Description
JOBNAME	N/A	$ZAPJOB	The name of the job to be zapped.
JOBID	N/A	$ZAPJOB	The JES2 JOBID of the job to be zapped.
JOBKEY	N/A	$ZAPJOB	The hexadecimal job key that is associated with the job.
JQEINDEX	N/A	$ZAPJOB	The index of the JQE to be zapped.
JQEOFF	N/A	$ZAPJOB	The offset of the JQE to be zapped.

Important: If used improperly, the ZAPJOB initialization statement can cause JES2 abends, including abends on multiple systems. Review the entire ZAPJOB documentation before using this initialization statement.

When specifying a job to zap, IBM suggests specifying as many operands as are known for that job. This reduces the possibility of accidentally zapping the wrong job because of a typing error.

Zapping a job that is active can lead to abends and the loss of a PCE until JES2 is restarted. ZAPJOB will not correct queue errors.

7.4 Testing JES2 exits

Many customers have JES2 exits. The testing of changes to the exit can be disruptive because JES2 must be restarted (at least) to load a new version of the exit code.

Use a secondary JES2 (also known as a Poly-JES) to test changes to JES2 exits. Testing can be further facilitated by using a sample JES2 Exit 5 that is available on the CBT tape. This exit provides two new JES2 commands, called $ADDEXIT and $REPEXIT, which can be used to dynamically add and replace JES2 user exits without restarting JES2. Note that JES2 must be restarted once to load the new JES2 commands. Then, after this restart, changed exits can be dynamically reloaded with no further JES2 restarts required.

7.5 Spool fencing and affinities

The normal mode of operation of JES2 is to allow all jobs to allocate all their track groups across all available spool volumes. This balances the performance and space utilization across the volumes, but results in many jobs being associated with each spool volume. If you wish to stop and remove that volume subsequently, the actual drain process can take a long time because so many jobs are involved.

An alternative is to use *spool fencing*, whereby a job or job class can be limited to using a subset of the volumes. If you have a lot of churn in your DASD farm and find that you frequently move or replace spool volumes, you may be interested in exploring this capability. Refer to “SPOOL partitioning” in *z/OS JES2 Initialization and Tuning Guide,* SA22-7532, for more information about this topic.

There is a similar but different function that you may be interested in if you have multiple sites, with primary DASD in each site. This function allows you to specify an affinity between a given system or set of systems (as opposed to a job or job class) and a SPOOL volume. This gives you the ability to limit the systems in each site to using the DASD that are in the same site, protecting your JES2 environment from connectivity failures or planned site outages.

Whereas the association between a *job* and a SPOOL volume is specified via the FENCE parameter in JES2PARM, the only way to associate a *system* with a SPOOL volume is by using the $TSPOOL command after JES2 has initialized.

7.6 Handling sysout files for long-running tasks

It is not uncommon to find started tasks that are started when the system is IPLed, and run until it is shut down. One characteristic that can limit how long a started task can run for is when the started task continually sends output to a sysout data set without closing that data set to allow the file to be processed. In this situation, the output from the task could potentially be consuming a lot of space in the spool; however, the file cannot be processed until it is closed.

To address this situation, JES2 provides support for intermittently closing off these spool files. The JOBCLASS JESLOG statement supports the SPIN keyword, providing the ability to indicate that the file should be closed on a time basis (at a given time every day, or every so-many hours) or based on the number of lines of output produced.

You can also use the JESLOG keyword with the START command when starting a started task or started job.

This keyword applies to every started task. If you are unable to apply this process to all started tasks, an alternate is to use the SEGMENT keyword on the SYSOUT DD statement. This is not as flexible as SPIN, because it only supports segmentation based on amount of output and not on time. However, it may be a viable solution if you only wish to apply this to a small number of started tasks.

7.7 JES2 Health Monitor

The JES2 Health Monitor introduced in z/OS 1.4 is a self-starting, self-diagnostic service aid that allows JES2 to monitor very severe performance problems within the JES2 address space. It runs in its own address space, jesxMON (where jesx is the specific subsystem name). The health monitor automatically starts when JES2 is initialized, and terminates when JES2 terminates cleanly, such as in response to a $P JES2 command.

Do not consider the health monitor to be a “performance monitor” in the traditional sense. Instead, think of it as an overall subsystem status reporter. The health monitor samples JES2 processing, collects data, and reports situations where JES2 is not responding to commands, and you cannot easily diagnose the problem. Such situations can be as basic as a command that is taking an unexpected amount of time to complete, a legitimate “bug” in JES2, an exit routine problem, or other code running in the JES2 address space.

Note: The JES2 Health Monitor is not a part of the z/OS Health Checker introduced into z/OS in z/OS 1.7, and does not interact with the Health Checker.

JES2 provides a set of $J commands to allow you to both control the monitor itself, and request JES2 to report the data it has collected for the operator to view. In z/OS 1.7, SDSF has been enhanced to display information from the health monitor and to issue $J commands. See *z/OS SDSF Operation and Customization,* SA22-7670, for more detailed information.

The $J commands and their function are listed in Table 7-5. Become familiar with these commands and their functions in advance of any problems, so that you can use them effectively should a JES2 problem arise.

Table 7-5 JES2 Health Monitor ($J) commands

Command	Description
$JDDETAIL	This command should be used to determine if a resource problem is impacting JES2. The first part of the display contains statistics on the major JES2 resources. The second part displays JES2 CPU sampling statistics. High sample counts with a status other than ACTIVE or IDLE might indicate a problem. The final section is the MVS wait table. The main task should not enter long MVS WAITs after initialization. Typically, waits indicate some resource that was not immediately available to JES2.
$JDHISTORY	At the beginning of every hour, JES2 saves the current statistics and resets the count. Use this command to review the saved statistics. The monitor obtains and displays up to 72 hours of samples, based on how long the monitor address space has been running. This command allows subscripts for resources.
$JDJES	$JDJES displays information about conditions that the monitor is tracking that are not considered problems, as well as the data displayed in response to $JDSTATUS.
$JDMONITOR	This command displays the status of all the tasks that make up the monitor, as well as information on all the modules in the monitor. If the monitor is not functioning properly, this command can provide information to assist its diagnosis.
$JDSTATUS	This is the primary monitor command that you should use to diagnose problems with JES2. This command displays all current alerts and any notices that are outstanding.
$JSTOP	You might need to stop the monitor address space because it is not functioning properly, or because you want to reset the time frame for the statistics that the monitor has been maintaining.

7.8 JES2 commands not related to JES2 Init statements

Table 7-6 lists some JES2 commands that may be helpful in diagnosing and addressing JES2 problems, potentially saving you from a JES2 outage.

Table 7-6 JES2 commands

Command	Function	Description
$ACTIVATE	To activate a particular level of JES2 processing. This command is not included in z/OS 1.7	This command comes and goes depending upon the release of JES2 and whether or not it is needed. Specifying “R4” activates the compatibility mode of JES2 processing. This allows pre-z/OS 1.2 members to coexist in the MAS with z/OS 1.2 members. In this mode, certain z/OS 1.2 functions are not available. Specifying “Z2” activates the z/OS 1.2 full–function level of JES2 processing. This enables larger limits for jobs, output elements, trackgroup space, job number range, and improved trackgroup usage tracking. You should not move to the Z2 level until all systems in the MAS are stable on z/OS 1.2 or later. Level Z2 has implications for exits and vendor products The command affects the entire MAS.
$D PERFDATA	Displays JES2 performance statistics	Described in detail in the Washington Systems Center Flash W9744B. See 7.8.1, “PERFDATA” on page 84 for more information.
$P SPOOL	Drain a spool volume	Use this command to drain and delete an entire spool volume by processing all work on the volume and preventing any available space on the volume from being allocated. This command has an effect across the entire MAS.
$S SPOOL	Start a spool volume	Use to add or reactivate a spool volume to the spool configuration. This command has an effect across the entire MAS.
$T PERFDATA	Resets JES2 Performance statistics	See 7.8.1, “PERFDATA” on page 84 for more detail.
$T SPOOL	Modify a spool volume	Modify a spool volume MAS-wide. This command has an effect across the entire MAS.
$Z SPOOL	Halt a spool volume	Use to deallocate a spool volume after active work completes its current phase of processing (executing, printing, punching.) This command has an effect across the entire MAS.

Both the Health Monitor ($J) commands and the PERFDATA commands can give important information about JES2 performance to allow analysis of potential problems occurring in JES2 that might, at first glance, be cause for an unplanned IPL.

7.8.1 PERFDATA

JES2 provides internal performance statistics which can be displayed with the **`$D PERFDATA`** operator command. You can find more information about this “undocumented” performance tool in IBM Redpaper *JES2 Performance,* REDP3940.

Important: This is intended to be used under the direction of personnel knowledgeable in interpreting JES2 PERFDATA displays. Development may add, change or delete any part thereof as required to improve its usefulness as necessary, so there is no intent to document the externals in the JES2 product publications at this time.

There are a number of different types of information you can get using the **$D PERFDATA** command, depending on which additional keyword you provide:

- Checkpoint Statistics - **$D PERFDATA(CKPTSTAT)**

  This displays a summary of the checkpoint statistics that are available through the $TRACEID=17 data.

- CPU Statistics - **$D PERFDATA(CPUSTAT)**

  This displays a summarized view of the PCESTAT statistics, including:

  - CPU time and wall clock run time for each processor control element (PCE) and PCE type
  - I/O counts for each PCE and based on PCE type
  - $QSUSE time and checkpoint counts

- Event Statistics - **$D PERFDATA(EVENT)**

  This displays significant events such as $DISTERR errors and long-running PCEs.

- Initialization Statistics - **$D PERFDATA(INITSTAT)**

  This shows the CPU and wall clock time for each Initialization Routine module, warm start processing, and MVS initialization (from JES2 start until HASPNUC is first entered)

- PCE Statistics - **$D PERFDATA(PCESTAT)**

  This displays, for each generic PCE type, the activity, execution time, and delays encountered.

  - Elapsed and CPU time for each PCE type
  - I/O counts for each PCE and based on PCE type
  - $WAIT time and count based on $WAIT macro and PCE type.
  - $POST reasons, counts and wait times for each $WAIT macro.
  - $QSUSE time and checkpoint counts

- QSUSE utilization and delay - **$D PERFDATA(QSUSE)**

  This displays the usage and response time for $QSUSE service which is used by many JES2 processes to acquire the checkpoint.

  - Module and sequence number of every $QSUSE macro that $WAITed
  - Count of times each $QSUSE was entered and accumulated wait time
  - Average $WAIT time per $QSUSE macro

- WLM Sampling data - **$D PERFDATA(SAMPDATA)**

  This displays a summary of the Workload Manager (WLM) sampling data used to determine the status of service class queues used for WLM-managed initiators.

  - Service classes known to JES2 - registered, by system.
  - Sampling data for each service class, including the number of jobs in the MAS with queue delay, number of jobs ineligible for any system, or jobs limited by the job class maximums.
  - The count of jobs with queue delay and ineligible on the local system is also provided.
  - Report class sampling data is also given for the sysplex (JESplex).

- Subtask Statistics - **$D PERFDATA(SUBTSTAT)**

  This displays, for each JES2 subtask, the count, the average queue time, the run time, and the CPU time (added with APAR OW55693).

8

JES3

This chapter describes changes to JES3 to reduce the need for planned outages. After the introduction of the hot start refresh (HR) function in OS/390 2.4, most changes to the initialization stream can be made dynamically. Prior to this, a warm start was required, which implied a sysplex IPL.

Here we list the initialization statements and selected parameters, along with the release in which each became dynamically changeable.

8.1 JES3 initialization statements

Beginning with OS/390 V2R4, a new type of JES3 start known as *hot start with refresh* is allowed. This allows JES3, on a hot start, to reread the initialization stream to allow many (but not all) of the parameters to be changed without an IPL.

Previously, a JES3 warm start was required to perform this function. A JES3 warm start requires all the MVS images in the JES3 complex to be IPLed. A JES3 hot start, on the other hand, only requires JES3 to restart on the global. Jobs currently in execution either on the global or on any local processors in the JES3 complex continue to execute.

Keep in mind that not all initialization statements and parameters on initialization statements are processed during a hot start with refresh. Those statements and parameters which are not processed will be syntax-checked only.

Table 8-1 lists and describes all JES3 initialization statements, as well as whether they are dynamic and in which release they became dynamic.

Table 8-1 JES3 Initialization statements

Initialization statement	Available in z/OS	Parms dynamic	Description
ACCOUNT	1.4	Yes	Default Job accounting information. No parameters. Changed with hot refresh only.
BADTRACK	N/A	No	Identifies a defective track on spool and prevents its use for subsequent allocations. Following is the command to add a Bad track statement temporarily: `*MODIFY Q DD=ddname,CYL=ccc,TRACK=ttt` Making this permanent requires a warm or cold start.
BUFFER	1.4	Some	Establishes JES3 buffer pool and size of JES3 DASD records.
CIPARM	1.4	Yes	Specifies variable converter/interpreter (C/I) parameter default values.
CLASS	1.4	Yes	Specifies characteristics of the JES3 job class.
COMMDEFN	1.4	Yes	Identifies the VTAM/JES3 interface.
COMPACT	1.4	Yes	Identifies a compaction table.
CONSOLE	1.4	Yes	Specifies the characteristics of a RJP workstation console.
CONSTD	1.4	No	Defines standards for console configuration All parameters require a warm start.
DEADLINE	1.4	Yes	Establishes deadline scheduling algorithms as specified on the //*MAIN card. All type parameters can be changed with hot start refresh and commands to change it start with: `*MODIFY L,T=type`
DESTDEF	1.4	Yes	Specifies how inbound SYSOUT data sets from other NJE nodes are to be processed. Both DEVICE= and USERID= parameters can be changed with Hot start Refresh or, as of z/OS 1.5, using the command: `*MODIFY CONFIG,ADD=member`
DEVICE	1.4	Yes	Specifies the characteristics of each device defined to JES3.

Initialization statement	Available in z/OS	Parms dynamic	Description
DYNALDSN	1.4	Yes	Identifies protection requirements for dynamically allocated data sets. Both PROTECT= and BYPASS= parameters can be changed with hot start refresh.
DYNALLOC	1.4	Yes	Dynamically allocates a data set or device without changing the JES3 startup procedure.
FORMAT	1.4	No	Requests formatting for the JES3 spool data set. Changes to STT= and STTL= require cold start, and changes to DDNAME= and SPART= require warm start.
FSSDEF	1.4	Yes	Defines the characteristics of the functional subsystem.
GROUP	1.4	Yes	Specifies characteristics of the JES3 job class groups.
HWSNAME	1.4	Yes	Identifies groups of I/O units for high watermark setup. The type of device can be changed with a hot start refresh.
INCLUDE	1.4	Yes	Selects and includes segments of initialization stream into the primary initialization stream.
INTDEBUG	1.4	Yes	Provides the facility to generate a dump associated with an initialization stream error message. The index and message parameters can be changed with hot start refresh.
MAINPROC	1.4	Most	Identifies the characteristics of each main in the JES3 complex.
MSGROUTE	1.4	Yes	Assigns MVS message routing for each main. The MAIN= and J= parameters can be changed with a hot start refresh.
NJECONS	1.4	Yes	Defines the message class to which JES3 networking is to send networking messages. The CLASS= parameter can be changed with a hot start refresh.
NJERMT	1.4	Most	Defines a node in a network of nodes.
OPTIONS	1.4	Most	Specifies certain options for the JES3 system.
OUTSERV	1.4	Yes	Specifies output service defaults and standards.
RESCTLBK	1.4	Yes	Specifies that main storage be preallocated for the high-usage function control table (FCT). The FCT= parm can be changed with a hot start refresh.
RESDSN	1.4	Yes	Specifies the names of frequently used permanently resident data sets. The DSN= parameter can be changed with a hot start refresh.
RJPLINE	1.4	Yes	Specifies the characteristics of a single BSC line that the JES3 global will use for RJP.
RJPTERM	1.4	Yes	Specifies the characteristics of a remote terminal for BSC remote job processing.
RJPWS	1.4	Yes	Specifies the characteristics of a remote work station for SNA RJP.
SELECT	1.4	Yes	Specifies the scheduling parameters for job selection mode.

Initialization statement	Available in z/OS	Parms dynamic	Description
SETACC	1.4	Yes	Defines nonshared permanently resident direct access volumes prior to main initialization. The VOL= parameter can be changed with a hot start refresh.
SETNAME	1.4	Yes	Specifies groups of devices within device types. Made dynamic as of OS/390 2.9
SETPARAM	1.4	Most	Specifies Main Device Scheduler (MDS) for allocating I/O devices.
SETRES	1.4	Yes	Specifies which direct access volumes are to be made JES3 mounted at main initialization. The VOL= parameter can be changed with a hot start refresh.
SPART	N/A	No	Defines and names a spool partition.
STANDARDS	1.4	Most	Specifies standard default values and job options for the JES3 complex.
SYSID	1.4	Yes	Defines the default MVS/Bulk Data Transfer (BDT) node for the JES3 complex. The NAME= and CLASS= parameters can be changed with a hot start refresh.
SYSOUT	1.4	Yes	Specifies the characteristics of a SYSOUT class.
TRACK	N/A	No	Identifies a formatted volume that is allocated for the spool. STT= and STTL= require a cold start. DDNAME= and SPART= require a warm or cold start.

Note: Related commands to dynamically make the changes are included where applicable. If a hot start with refresh fails, any changes that were made to the initialization stream will not be in effect. That is, a subsequent hot start obtains the information from the last successful cold, warm, or hot start with refresh.

Also: During a JES3 hot start with refresh, only the parameters that are changeable are processed. Errors may result if non-changeable parameters are referenced or modified.

8.2 JES3 miscellaneous features

The following are miscellaneous JES3 features that assist in avoiding IPLs.

8.2.1 Dynamic config command support

JES3 allows you to add SNA/RJP workstations, non-channel-attached FSS printers, and so on dynamically (in other words, without having to restart JES3).

To add new definitions, you code initialization statements, just as you would if you were restarting JES3, and specify the member containing the initialization statements on the `*MODIFY,CONFIG` command. The member must be in the data set specified on the JES3IN DD statement that was used to start JES3.

The following initialization statements are supported during **`*MODIFY,CONFIG`** processing:

- RJPWS - Define SNARJP workstation characteristics
  - WS (RJPWS statement)
  - CONSOLE statement
  - DEVICE statement
- CONSOLE - Define SNA/RJP console
- DEVICE - Define SNA/RJP devices and non-channel-attached FSS-managed printers
- FSSDEF - Define writer FSS
  - FSSDEF statement
  - DEVICE statement
- INTDEBUG - Initialization debugging facility
- INCLUDE - Include another initialization stream member

  Note: If the JES3IN DD data set is concatenated, only the members in the first data set of the concatenation are used in processing the INCLUDE statement.

Command for dynamic config

Use the following command for dynamic config:

```
*MODIFY CONFIG,ADD=xxxx
```

Where xxxx is a member referenced in the data set referenced by the JES3IN DD procedure, or the first data set if it is a concatenation.

Note: Most changes made this way will not be kept across a JES3 restart other than a hot start.

See *z/OS JES3 Initialization and Tuning Reference,* SA22-7550 for further information.

8.2.2 Dynamic LPA facility

As of OS/390 V2R6, JES3 supports the use of the dynamic LPA facility to activate service to JES3 LPA modules. Prior to this release, an IPL with CLPA was required to activate the service.

To change JES3 LPA modules dynamically, you first use the **`SETPROG LPA`** command. For example, to change module IATSICA, you would issue the following command on your global and/or local processors:

```
SETPROG LPA,ADD,MODNAME=IATSICA,DSN=LNKLST
```

When the command completes, you then perform a hot or local start of JES3 without an IPL. JES3 picks up the new versions of the LPA modules during initialization.

8.2.3 SYSIN DD statement limit

As of z/OS 1.4, this feature prevents jobs from filling up JSAM buffers by cancelling them.

To use this feature, update the MAXINDD PARM on the STANDARDS initialization statement. The default value is the same as defined in the MAXJOBST parameter.

To determine the correct value for your installation, see *z/OS JES3 Initialization and Tuning Guide,* SA22-7549

8.2.4 Changing devices using HCD

You can use the hardware configuration dialog (HCD) to change devices in your JES3 installation. The implications of using this in OS/390 V2R9 JES3 are as follows:

- You can change characteristics of devices without performing an IPL or JES3 restart even if the device is defined to JES3 as a JUNIT or XUNIT. Prior to OS/390 V2R9, JES3 would not allow this change.
- You can delete a device that is defined to JES3 without performing an IPL or JES3 restart. If you do restart JES3 without deleting the device from your initialization stream, JES3 will tolerate the undefined device. However, you should schedule a hot start with refresh at your earliest opportunity to delete the device from JES3 cleanly.
- You can add devices to HCD. However, if you want to define these devices to JES3, you must also add them to your initialization stream and perform a hot start with refresh. You do not need to IPL.

9

UNIX System Services

This chapter describes features of UNIX System Services (USS) that reduce or remove the requirement for planned outages.

9.1 UNIX System Services (USS) enhancements checklist

Table 9-1 contains a checklist of enhancements to the UNIX System Services component of the operating system.

Table 9-1 Checklist of outage avoidance items for USS component of z/OS and OS/390

Features/Enhancements	Introduced in z/OS release	Description	Completed
Dynamically changing BPXPRMxx parameters	1.4	The SETOMVS and SET OMVS commands can be used to dynamically change the options that z/OS USS is currently using. See 9.1.1, "Dynamically changing BPXPRMxx parameters" on page 95 for more detail.	
OMVS Restart	1.4	Use the `MODIFY OMVS` command to recycle z/OS UNIX System Services. See 9.1.2, "OMVS Restart" on page 95 for more detail.	
Dynamically activating USS service	1.7	Allows dynamic activation and deactivation of service items (PTFs, ++APARS, ++Usermods) that affect the UNIX System Service component modules. See 9.1.3, "Dynamically activating USS component service items" on page 96 for more detail.	
File system diagnostic and recovery operations	1.4	It is possible to perform file system diagnostic or recovery operations via a z/OS `MODIFY` command. See 9.1.8, "File system diagnostic and recovery operations" on page 100 for more detail.	
Latch contention	1.6	The `MODIFY BPXOINIT` command can be used to abend user tasks that are holding latches for an excessive amount of time. See 9.1.9, "Latch contention" on page 101 for more detail.	
Process termination	1.6	z/OS 1.6 introduced enhancements to the process termination function. See 9.1.10, "Process termination" on page 102 for more detail.	
Remountable filesystems	1.5	A new function in z/OS 1.5 allows the remount of an HFS or zFS file system within a directory tree so as to change the access mode. See 9.1.11, "Changing the mount attribute of a mounted filesystem" on page 102 for more detail.	
Colony address space Master Scheduler support	1.4	Colony address spaces can be run as SUB=MSTR. See 9.1.12, "Colony address space Master Scheduler support" on page 103 for more detail.	
SWA(ABOVE)	1.5	Allows all SWA control blocks are to be allocated above the 16 MB line. See 9.1.13, "SWA(ABOVE)" on page 104 for more detail.	
zFS enhancements	1.7	z/OS 1.7 delivers some changes in the way zFS works that provide greater availability. See 9.1.14, "zFS enhancements" on page 105 for more detail.	

Features/Enhancements	Introduced in z/OS release	Description	Completed
IBM Health Checker for z/OS	1.7	IBM Health Checker for z/OS is a tool that checks the current active z/OS and sysplex settings and definitions for an image and compares their values to either those suggested by IBM or defined by the customer. See 9.1.15, "IBM Health Checker for z/OS" on page 106 for more detail.	

9.1.1 Dynamically changing BPXPRMxx parameters

The parameters that the **SETOMVS** command can modify are originally set in the BPXPRMxx Parmlib member during IPL. Changes to all the *system-wide* limits take effect immediately. When a *process* limit is updated, all processes that are using the system-wide process limit have their limits updated. All process limit changes take effect immediately except those processes with a user-defined process limit (defined in the OMVS segment or set with a **SETOMVS PID=** command). Exceptions are MAXASSIZE and MAXCPUTIME, which are not changed for active processes.

The command provides the ability to specify the changed values on the command line, or to reload a BPXPRMxx member in Parmlib (see the **RESET=** keyword which is similar to the SET OMVS command). The command also provides a **SYNTAXCHECK=** keyword that returns a message indicating either that the syntax is correct, or that syntax errors were found and written into the hard copy log. This command parses the Parmlib member in the same manner, and with the same messages as during IPL. This feature therefore allows dynamic changes to be made via a BPXPRMxx Parmlib member, while at the same time ensuring that the syntax is correct for the next IPL.

You can refer to the "SETOMVS" section of *z/OS MVS System Commands,* SA22-7627, for a table which documents which parameters can be changed using the **SETOMVS** and **SET OMVS** commands, as well as the difference between those commands.

As of z/OS 1.7, both **SETOMVS** and **SET OMVS** have the ability to remount an unmounted root filesystem.

For more information see the following:

- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS System Commands,* SA22-7627

Note: If a process-level limit is lowered with the SETOMVS command, some processes may immediately hit 100% usage. Depending on the process limit specified, and what the process is doing, this could cause some processes to fail.

9.1.2 OMVS Restart

Prior to z/OS 1.3, the only way to shut down the UNIX System Services environment was to IPL the system. As of z/OS 1.3, the OMVS shutdown and restart commands allow for amendment and re-initialization of the z/OS UNIX environment without IPLing.

MODIFY OMVS,SHUTDOWN shuts down the UNIX services environment, which includes quiescing all running UNIX services work on a given system.

MODIFY OMVS,RESTART restarts a previously shut down UNIX services environment.

The ability exists to perform a partial shutdown of z/OS UNIX to allow JES2 to be shut down for maintenance purposes. The procedure relies on all forked processes being successfully terminated.

For more information see the following:

- “Partial shutdowns (for JES2 maintenance)” in the chapter “Managing Operations” in *z/OS UNIX System Services Planning,* GA22-7800
- “Planned shutdowns using MODIFY OMVS,SHUTDOWN” in the chapter “Managing Operations” in *z/OS UNIX System Services Planning,* GA22-7800
- “Recycling z/OS UNIX System Services (z/OS UNIX)” in *z/OS MVS System Commands,* SA22-7627

Notes:

- There are instances where `MODIFY OMVS,SHUTDOWN` cannot shut down z/OS UNIX completely and an IPL is still required in order to correct the condition that instigated the shutdown.
- Use the `MODIFY OMVS,SHUTDOWN` command carefully, because this will take down other system address spaces. As a result, some system-wide resources may not be completely cleaned up during a shutdown and restart.

  Do not use this command to shut down and restart the z/OS UNIX environment on a frequent basis (doing so will eventually lead to the need for an IPL.) An example of a system-wide resource that can be consumed due to the shutdown are non-reusable ASIDs. If colony address spaces are being used, a non-reusable ASID will be consumed for each colony address space that is shut down.
- To shut down the z/OS UNIX System Services environment, any application or subsystem workloads using z/OS UNIX services must be quiesced. This will include TCP/IP and may include CICS, DB2, and IMS, depending on your configuration.

9.1.3 Dynamically activating USS component service items

z/OS 1.7 added the ability to dynamically activate some USS service. This capability is primarily intended to allow an installation to activate urgent corrective service that cannot wait for the next planned IPL of the systems. Although this capability can be used to activate preventive service on an ongoing basis, you need to take care with regard to the amount of available ECSA. For this reason it is not intended as a replacement for the regular application of service that does require an IPL.

Dynamic service activation allows you to dynamically activate and deactivate service items (PTFs, ++APARS, ++Usermods) that affect the UNIX System Service component modules without having to re-IPL. This will minimize the number of both planned and unplanned outages and will lead to a higher level of system availability.

In order to be prepared to exploit dynamic service activation, you have to stay current on z/OS UNIX System Services maintenance. Doing so will make it more likely that any given service item can be activated dynamically because the running system will be at a high enough level to accept the service item.

Note: Although this feature can be used to activate most z/OS UNIX System Services component PTFs, it is not intended to be used as a way to activate a large set of maintenance for preventive purposes. On a periodic or need basis, you have to determine the PTFs that you would be interested in activating dynamically for corrective purposes. These PTFS would likely be of the highest severity and impact to your system.

The PTFs that are capable of being activated dynamically are identified by ++HOLD REASON(DYNACT) data inside them. You must follow the instructions included with this hold data in order to properly activate the PTF.

9.1.4 Exploiting dynamic service activation

This capability is primarily intended to allow an installation to activate corrective service to avoid unplanned re-IPLs of your systems. Additionally, this capability can be used to activate a temporary patch that can be used in gathering additional documentation for a recurring system problem. Although this capability can be used to activate preventive service on an ongoing basis, it is not intended for this purpose as a replacement for the regular application of service that does require a re-IPL.

To exploit this feature do the following:

- Define SERV_LPALIB and SERV_LINKLIB parameters.

  On the SERV_LPALIB and SERV_LINKLIB parameters in the BPXPRMxx Parmlib member, specify the LPALIB and LINKLIB target libraries for z/OS UNIX modules where you install service with SMP/E and you intend to activate it dynamically. They are normally the same system clone data sets where you would SMP/E install your regular service to be used on the next IPL; for example:

  ```
  SERV_LPALIB('dsname','volser')
  SERV_LINKLIB('dsname','volser')
  ```

  These parameters can be set up at IPL time; at restart time of OMVS; or dynamically via the use of the **`SET OMVS=`** or SETOMVS command.

- The following are the console commands to do the activate and deactivate:
  - **`F OMVS,ACTIVATE=SERVICE`** activates maintenance at any time after z/OS UNIX System Services Initialization during the operation of a running system.
  - **`F OMVS,DEACTIVATE=SERVICE`** deactivates service that is no longer required or desired.
  - **`D OMVS,ACTIVATE=SERVICE`** invokes the BPXEKDA interface to retrieve dynamic service activation information.
  - **`D OMVS,O`** invokes C/C++ API get_system_settings() to retrieve new Parmlib settings.

9.1.5 Activate service items

To activate the service items dynamically after applying maintenance, do the following:

1. In the BPXPRMxx Parmlib member, define the target service activation libraries on `SERV_LPALIB` and `SERV_LINKLIB` parameters.

   In our example, these two parameters point to the SMPE target libraries. We put them in a separate BPXPRMSS member and activate the member by using the SETOMVS RESET=SS command.

2. Issue: **F OMVS,ACTIVATE=SERVICE**

   This will dynamically activate the service from the target libraries specified in SERV_LPALIB and SERV_LINKLIB parameters.

   The new service will only be activated if the system is at a high enough level to accept the service, and if all of the service items are found to be complete. Error messages will be displayed indicating which service items are incomplete or cannot be activated due to the level of the current system.

   Even when an **F OMVS,SHUTDOWN** has been performed to shut down z/OS UNIX System Services, this command can be issued to activate service against the component. Additionally, any service activated prior to an **F OMVS,SHUTDOWN** command stays in effect when z/OS UNIX System Services is restarted.

   You are prompted as follows to validate that the listed service items are the ones that are intended to be activated. This message insures that these are the service items that you intended to activate. Additionally, the amount of storage that will be consumed in both ECSA (Nnnnnn bytes) and in the OMVS address space private area (Mmmmm bytes) to perform the activation is identified in this message.

   ```
   BPXM061I THE FOLLOWING SERVICE ITEMS WILL BE ACTIVATED:
   OA09999  OA08888
   ECSA STORAGE BYTES:  24576 AND OMVS PRIVATE STORAGE BYTES: 12468
   WILL BE CONSUMED FOR THIS ACTIVATION.
   *06 BPXM061D REPLY "Y" TO CONTINUE. ANY OTHER REPLY ENDS THE COMMAND.
   ```

9.1.6 Deactivate service items

Issue command **F OMVS,DEACTIVATE=SERVICE** to back off the last set of service items that were activated dynamically.

Only the service items that were previously activated with the **F OMVS,ACTIVATE=SERVICE** are backed off. This function is intended to be used to remove maintenance that was put on for temporary purposes, or to remove maintenance that was previously activated and is causing problems on the system. When the command completes the service items are deactivated, but the modules that were part of the service items remain in storage because of potential latent usage of the modules.

You will be prompted to verify the deactivation.

```
BPXM063I THE FOLLOWING SERVICE ITEMS WILL BE DEACTIVATED:
         OA09999  OA08888
*07 BPXM063D REPLY "Y" TO CONTINUE. ANY OTHER REPLY ENDS THE COMMAND.
```

9.1.7 Display activated service items

To display all dynamically activated service items, issue command **D OMVS,ACTIVATE=SERVICE**. The service items are displayed in sets based on when they were activated, from the most recently activated set of service items to the oldest set of service items. The most recently activated set of service items is shown at the top of the display to indicate it is the highest level of service active on the system, as shown in Figure 9-1 on page 99. The display also indicates the amount of ECSA and OMVS address space private storage that is consumed for all of the activated service items.

```
BPXO059I 08.51.42 DISPLAY OMVS 284

OMVS     000E ACTIVE          OMVS=(6D)
DYNAMIC SERVICE ACTIVATION REPORT
 SET  #2:
  LINKLIB=SYS1.DYNLIB.PVT                  VOL=BPXLK1
  LPALIB=SYS1.DYNLIB.LPA                   VOL=BPXLK1
  OA02001  OA02002  OA02003  OA02004
SET  #1:
 LINKLIB=SYS2.DYNLIB.PVT                   VOL=BPXLK1
 LPALIB=SYS1.DYNLIB.LPA                    VOL=BPXLK1
 OA01001  OA01002  OA01003
ECSA STORAGE:  68496       OMVS  STORAGE:  268248
```

Figure 9-1 Display activated service

Display the Parmlib settings

Issue command **D OMVS,O** to display all z/OS UNIX System Services Parmlib settings, which includes the settings for the new SERV_LPALIB and SERV_LINKLIB parameters, as shown in Figure 9-2.

```
BPXO043I 12.34.23 DISPLAY OMVS 209
OMVS     000E ACTIVE          OMVS=(7F)
CURRENT UNIX CONFIGURATION SETTINGS:

MAXPROCSYS      =       256    MAXPROCUSER     =        16
.
.
.
SWA                      =  BELOW
SERV_LINKLIB     =  SYS1.DYNSERV.LINKLIB   BPXLK1
SERV_LPALIB      =  SYS1.DYNSERV.LPALIB    BPXLK1
```

Figure 9-2 Display parmlib settings

Documentation

For more information refer to:

- “Dynamically activating the UNIX System Services component service items” in chapter “Managing Operations” in *z/OS UNIX System Services Planning,* GA22-7800
- “Dynamically activating maintenance for z/OS UNIX System Services” in chapter “MVS System Commands Reference” in *z/OS MVS System Commands,* SA22-7627
- “Dynamic service activation for z/OS UNIX” in IBM Redbook *z/OS Version 1 Release 7 Implementation,* SG24-6755

Notes:

- Only those service items that are identified internally by UNIX System Services as capable of being dynamically activated are activated. Service items that are not explicitly identified as such cannot be activated.
- The **MODIFY OMVS,ACTIVATE=SERVICE** command only applies to the kernel component of USS (that is, LINKLIB and LPA modules). It does *not* encompass changes to code contained within HFS.
- If a fix capable of dynamic activation is found that cannot be activated due to back-level service on the active system or missing parts in the target activation libraries, the activation will fail.
- Some UNIX System Services modules and CSECTs will not be capable of dynamic activation due to their location or when they are run. As a result, they still require ++HOLD for IPL documentation in their PTFs (as almost all UNIX System Services PTFs do prior to z/OS 1.7). It is expected that a very small percentage of PTFs will be affected by this limitation.
- The activation of a set of service items potentially will consume storage in ECSA and the OMVS address space storage. This additional storage is permanent; that is, it is not released even if the service items are deactivated. A dynamic activation that involves fixes to modules in LPA resident load modules will cause additional consumption of ECSA.
- The amount of ECSA and OMVS address space storage that would be consumed by the service items associated with activation is indicated in messages displayed as a result of the **MODIFY OMVS,ACTIVATE=SERVICE** command. The issuer of the command is then prompted on whether to proceed with the activation based on this information. Give careful consideration to the amount of storage available before continuing with this command.

9.1.8 File system diagnostic and recovery operations

Note: This function is applicable only to a sysplex environment where shared file system has been enabled by specifying SYSPLEX(YES) in the BPXPRMxx Parmlib member named during system initialization. The command is intended to help diagnose and correct certain shared file system problems or errors that impact one or more systems in a sysplex environment.

A series of **MODIFY BPXOINIT** commands exist to allow file system diagnostic or recovery operations to be performed. These begin with **MODIFY BPXOINIT,FILESYS=** followed by a series of keywords that allow the user to perform a series of tasks including:

- Display information about all or specific filesysystems.
- Initiate an SVC dump to capture all of the file system sub-records in the active BPXMCDS couple data set.
- Perform automatic file system and couple data set diagnosis and repair.
- Re-initialize the file system hierarchy based on the ROOT and MOUNT statements in the BPXPRMxx Parmlib member used by each system during its initialization.

To obtain the best results, issue this command at the system with the highest shared file system software service level. To determine which system is executing with the highest shared file system software service level, issue the command **MODIFY BPXOINIT,FILESYS=DISPLAY,GLOBAL** and select the system with the highest “LFS Version” value.

Support for this command was introduced with z/OS 1.1 (and retrofitted back to OS/390 with APAR OW44461). Prior to this support becoming available, an IPL was required to remount the root file system, and may have been required to fix any filesystem errors.

For more information see the following:

- APAR OW54824 - you can obtain more information about this APAR at:

  `http://www.ibm.com/servers/eserver/zseries/zos/unix/pdf/ow54824.pdf`

- “Controlling z/OS UNIX System Services” in the chapter “MVS System Commands Reference” in *z/OS MVS System Commands,* SA22-7627

Notes:

- Use this command with caution, and only under the direction of an IBM service representative.
- Prior to OS/390 2.10 with APAR OW54824, a system could be targeted as a file system owner even though the system was in the process of (or had performed) a soft shutdown (**`MODIFY BPXOINIT,SHUTDOWN=FILESYS)`**.

  This can now be avoided by using the command **`MODIFY BPXOINIT,SHUTDOWN=FILEOWNER`** which provides the function of SHUTDOWN=FILESYS and also prevents the system from becoming a filesystem owner via a move or recovery operation until USS has been recycled.

9.1.9 Latch contention

Some installations have experienced problems with tasks holding onto latches for extended periods. In some cases, they resorted to IPLing the system to address the situation. z/OS 1.6 introduced the **`MODIFY BPXOINIT,RECOVER=LATCHES`** command which should be issued in response to message BPXM056E (which is issued 3 to 4 minutes after the contention is first detected) to aid in resolution of excessive latch contention.

The command is primarily intended as an aid in resolving latch hangs that are caused by user task usage of USS. However, it may be unable to resolve latch hangs caused by internal system tasks in the OMVS kernel address space, if the owning system task is processing critical system code that cannot be interrupted.

`DISPLAY OMVS,WAITERS` was introduced in z/OS 1.7 to display information about delays caused by the following conditions:

- Mount latch contention
- Outstanding unprocessed sysplex messages

See the section “DISPLAY OMVS Command” of *z/OS MVS System Commands,* SA22-7627 for an example of the output from this command.

For more information, refer to the following:

- “Ending processes” in chapter “Managing Operations” of *z/OS UNIX System Services Planning,* GA22-7800
- “Detecting latch contention” in chapter “Managing Operations” of *z/OS UNIX System Services Planning,* GA22-7800
- *z/OS MVS System Commands,* SA22-7627
- *z/OS MVS System Messages, Vol 3 (ASB-BPX),* SA22-7633

Notes:

- Only use this command if message BPXM056E is outstanding. The command causes one of the following:
  - If the contention can be resolved, the system DOMs the BPXM057E message.
  - If the contention cannot be resolved, the system issues message BPXM057E to indicate that condition.
- MVS isolates the abnormal termination to individual tasks, but this command can result in the termination of an entire process. Note that the abnormal termination will be caused by a non-retryable 422-1A5 abend that will cause the generation of a system dump due to the likelihood of an internal system problem. Additionally, if more than one latch is in contention, multiple tasks might be abended and result in requests for multiple dumps.

9.1.10 Process termination

To address situations where the **`KILL`** command will not terminate a process, z/OS 1.6 added the **`MODIFY BPXOINIT,SUPERKILL=pid`** command. This sends a termination signal to the target process. Prior to this enhancement, an IPL may have been required to force hung processes out of the system.

For more information see the following:

- "Controlling z/OS UNIX System Services" in chapter "MVS System Commands Reference" in *z/OS MVS System Commands,* SA22-7627

Note: This command terminates the entire process and any sub-processes within the address space. Only use this command as a last resort and only following unsuccessful attempts to terminate the process using **`MODIFY BPXOINIT TERM=`** and **`FORCE=`** commands.

9.1.11 Changing the mount attribute of a mounted filesystem

To change the mount mode of a file system (from read-only to read/write or vice versa), use the TSO/E UNMOUNT command or the ISPF shell. The REMOUNT operand on the UNMOUNT command specifies that the specified file system should be remounted, changing its mount mode in the process. Conditions under which a mounted filesystem would be remounted are:

- Maintenance cannot be performed on a read-only file system. The file system must be unmounted and then mounted again as read/write. If there are cascaded mounts, all of the file systems mounted on top of that file system must also be unmounted. A root filesystem can be unmounted and remounted. However, if the file system is a shared read-only root file system in a sysplex, it needs to be unmounted on all other systems in the sysplex.
- When not using shared filesystems, the remount facility can be used to mount file systems as read-only under normal operating situations and as read/write to perform maintenance.
- File systems that are to be remounted should not be exported by smb or dfs at the time of the remount.

Prior to this function becoming available, an IPL would have been required in order to make changes to a read-only root file system. The only alternative is that the whole structure below and including the file system itself (if being the sysplex root, this would hit the whole USS file structure) could be taken down and remounted again (if you can take down all applications that cannot live without specific USS file structures). This is possible, especially because IP can live without access to USS files for a while.

REMOUNT can be used in a shared file system if all systems in the sysplex have REMOUNT support (that is, they are at least at z/OS 1.5 or have the 1.4 PTF installed). The APAR adding the support for z/OS V1R4 is OA02584.

However, this APAR does not include the remount functionality for the USS chmount mount located in directory /usr/sbin. This means /usr/sbin/chmount only supports remounting (with options -r or -w for read-only or read-write) if run from a z/OS 1.5 or higher system, and if all systems in the USS sysplex sharing environment have the REMOUNT function code added. If there is at least one backlevel system, a remount request from any system will fail and the old behavior with errno 79x (=EINVAL) and errnojr 058804A5x (=JrNotSupInSysplex) will be seen, which means that this function is not supported in this environment.

Another way to remount a file system in an environment where a shared file system is being used is to unmount the file system and then mount it again in the desired mode.

For more information see the following:

- *z/OS Using REXX and z/OS UNIX System Services,* SA22-7806
- *zSeries Platform Test Report,* SA22-7997

Note:

- If a file is opened for a write, this is not checked if a remount operation changes the file system from read/write to read-only. Subsequent writes to the file will fail.
- zFS aggregates can be remounted if the primary file system is mounted. It will succeed when remounting from R/O to R/W or from R/W to R/O because the aggregate is detached during the unmount and then is attached in the new mode during the mount. However, remounts for HFS-compatible file systems in a sysplex will not work if both the clone and primary are mounted.

9.1.12 Colony address space Master Scheduler support

The default way to run a colony address spaces is to have it executing under JES, meaning that the address space cannot start if JES is unable to start for some reason. However, as of z/OS 1.3, this can be changed by including the SUB=MSTR parameter with the ASNAME keyword in the BPXPRMxx member. As a result, the colony address space runs as a master subsystem, outside JES control.

To set up a physical file system in a colony address space, create a cataloged procedure in SYS1.PROCLIB to start the colony address space. The name of the procedure must match the name specified on an ASNAME operand on the FILESYSTYPE statement in BPXPRMxx. The ASNAME keyword is specified as:

```
ASNAME(procname,'start_parms')
```

where:

procname	This is required and is a 1-to-8-character name in SYS1.PROCLIB.
'start_parms'	This is optional, and is a quoted string that is appended to the procname when the address space is started. The string can be up to 100 characters long.

The start_parms are not validated; they are just passed to the started task when the address space is started.

For example, an NFS Client with the cataloged procedure NFSCLNT is associated with the following FILESYSTYPE statement:

```
FILESYSTYPE TYPE (NFS)
ENTRYPOINT(GFSCINIT)
ASNAME(NFSCLNT,'SUB=MSTR')
```

Although this removes the necessity to have JES up before the colony address space, it does mean that the colony address space will no longer be able to use JES facilities. It also means that SDSF cannot be used to look at the address space joblog while the address space is running.

For more information, see "How to start colonies outside of JES" in *z/OS UNIX System Services Planning,* GA22-7800.

Notes:

- z/OS UNIX colony address spaces are started procedures. If running as a master subsystem, any DD SYSOUT= data sets specified in these procedures must be altered to point to a data set. This is because SYSOUT spool files are only supported under JES.
- If the NFS colony address space is started at IPL time, PATH= statements cannot be used in the JCL because the MOUNT statements have not yet been processed at this point in the IPL.
- Under certain conditions, Language Environment (LE) requires particular DD names to be opened. If these data sets have not been allocated in the procedure, LE dynamically allocates them with SYSOUT=*.

  The DD names are:

  - **SYSIN** - for standard input
  - **SYSPRINT** - for standard output
  - **SYSOUT** - for standard error
  - **CEEDUMP** - for capturing dumps formatted by Language Environment

  If any of these names are not currently used in the colony procedure, they must be added as DD DUMMY.

  If any of the existing DD SYSOUT= statements are not changed, or any of those dynamically allocated by Language Environment are not added, and an attempt is made to open that DD name, the result will be an ABEND S013.

9.1.13 SWA(ABOVE)

z/OS UNIX has the ability to exploit SWA(ABOVE). This was introduced to address the problem of a system wait state where the number of open filesystems was approaching 7500. SWA(ABOVE) is recommended to avoid these potential wait state conditions.

For more information see the following:

- The description of the BPXPRMxx member in *z/OS MVS Initialization and Tuning Reference,* SA22-7592

Note: Changing the SWA() parameter in BPXPRMxx requires an IPL to implement. However, restarting a colony address space will pick up the new SWA parameter immediately for that address space only.

9.1.14 zFS enhancements

Prior to z/OS 1.2, the HFS file system was the primary hierarchical file system. As of z/OS 1.2, installations can use any combination of HFS and zFS file systems. Because zFS has higher performance characteristics than HFS and is the strategic file system, support for HFS is planned to be discontinued in a future release and installations will need to migrate their remaining HFS file systems to zFS.

In z/OS V1R3, installations were allowed to run the z/OS UNIX environment with the root file system (or the version file system, in case of sysplex sharing) as a zFS root instead of an HFS root. However, zFS was still not supported during a Server Pac installation. The installation dialog panels allow specification of either HFS or zFS for a file system data set type since z/OS V1R5. As of z/OS 1.7, zFS is not only supported for the root, but also is the recommended file system.

z/OS 1.7 provides an ISPF-based tool (BPXWH2Z) to perform the migration of HFS file systems to zFS file systems. It has a panel interface that enables users to alter the space allocation, placement, SMS classes, and data set names. A HELP panel is provided.

This tool can:

- Migrate HFS file systems (both mounted and unmounted) to zFS file systems. If the HFS being migrated is mounted, the tool automatically unmounts it (after copying the data into the zFS file system) and then mounts the new zFS file system on the mount point where the HFS was mounted before.
- Define zFS aggregates by default to be approximately the same size as the HFS. The new allocation size can also be increased or decreased.
- Have the migration run in TSO foreground or UNIX background.

BPXWH2Z is located in partitioned data set SBPXEXEC (usually called SYS1.SBPXEXEC). When running the BPXWH2Z tool on a z/OS 1.7 system, it uses the z/OS 1.7 level of the `pax` command. This level has been enhanced for sparse file support and other characteristics that are of concern when migrating from an HFS to zFS file system.

Several enhancements have been added to BPXWH2Z with APAR OA13154. A completion message is now written into the summary file and a notification is sent when the conversion ends for both a successful and an unsuccessful execution.

Prior to z/OS 1.6, errors within zFS could result in filesystem termination. If zFS is stopped for some reason in z/OS V1R6, support has been added to the USS LFS code allowing continued access to zFS file systems by exploiting the USS sysplex sharing function. This applies to both file systems mounted read-write and those mounted read-only.

The zFS file systems mounted read-only will be mounted again locally when zFS is restarted and active again. File systems mounted read-write and successfully moved to a new owning system when zFS was going down locally, need be moved back to the local system if this is necessary because of performance reasons.

As of z/OS 1.7, zFS can now handle end-of-memory terminations and cleanup, rather than the filesystem terminating.

The HFS and zFS file system types in mount statements and command operands are now generic file system types that can mean either HFS or zFS. Based on the data set type, the system will determine which is appropriate. OMVS detects the file type from the data set directly and will honor the mount request.

Prior to z/OS 1.4, changes made to Parmlib member IOEFSPRM required zFS to be stopped and restarted. z/OS 1.4 added dynamic configuration support via the **`zfsadm`** shell command.

For more information see the following:

- *z/OS Distributed File Service zFS Administration,* SC24-5989
- *z/OS Distributed File Service zSeries File System Implementation,* SG24-6580
- "BPXPRMxx" in *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS UNIX System Services Planning,* GA22-7800
- *z/OS V1R7.0 Migration,* GA22-7499

9.1.15 IBM Health Checker for z/OS

The objective of the IBM Health Checker for z/OS is to identify potential problems before they affect system availability or cause outages. It is recommended that the IBM Health Checker for z/OS be customized and be run on a regular basis to provide early detection of potential problems. The Health Checker provides checks to help installations ensure their USS environment is set up for optimal availability.

For more information see the following:

- "IBM Health Checker for z/OS and z/OS UNIX" appendix in *z/OS UNIX System Services Planning,* GA22-7800
- *IBM Health Checker for z/OS: User's Guide,* SA22-7994

10

Hardware

This chapter describes features that have been introduced to reduce the need for planned outages for hardware changes. It includes discussion of both hardware and software features and capabilities. We assume that your current hardware configuration consists of multiple paths to all devices, and that single points of failure do not exist for any component.

10.1 Hardware facilities checklist

Table 10-1 contains a checklist of some hardware features with a brief description and which release of z/OS that it became available.

Table 10-1 Hardware facilities checklist

Features/Enhancements	Introduced in z/OS release	Description	Completed
Dynamic I/O configuration changes	All	Involves using the MVS `ACTIVATE` command to dynamically change the I/O Definition (IODF). See 10.1.1, "Dynamic I/O configuration changes" on page 109.	
Planning for HSA growth	All	In order to support dynamic changes to the device configuration, there must be unused space available in the HSA. See 10.1.2, "Planning for growth in HSA" on page 110.	
Dynamic logical partitions	1.5	Provides the ability to add or remove logical partitions on an existing z990, z890 or later processor. See 10.1.3, "Dynamic logical partitions" on page 110.	
LPAR Dynamic Storage Reconfiguration (DSR)	All	With some planning, storage for LPARs can be reconfigured without requiring a POR or IPL. See 10.1.4, "LPAR Dynamic Storage Reconfiguration (DSR)" on page 111.	
Use of Reserved CP capability	All	It is possible to bring additional, even newly installed, CPs online to an LPAR without even an IPL. See 10.1.5, "Defining Reserved CPs" on page 112 for more information.	
Multiple logical channel subsystem support (LCSS)	1.4	Configuration changes to multiple logical channel subsystems can be implemented dynamically as long as the required z/OS service is applied. See 10.1.6, "Multiple logical channel subsystem support (LCSS)" on page 113.	
Multiple Subchannel sets	1.7	This is a plan-ahead feature. See 10.1.7, "Multiple subchannel sets" on page 113 for further details.	
Over-defining channel paths	All	Defining more channel paths than are currently physically installed on an existing z990, z890 or later processor. See 10.1.8, "Overdefining channel paths" on page 113.	
Capacity Backup Upgrade (CBU), Capacity Upgrade on Demand (CUoD), and On/Off Capacity on Demand support	All	Features of System z® servers to enable dynamic capacity growth. See 10.1.9, "CBU, CUoD, and OOCoD support" on page 114 for further details.	
Exploit "Planahead" capability	All	"Planahead" ensures that future server upgrades can be implemented dynamically. See 10.1.10, "Planahead" on page 114 for more information.	
Software support for new devices	All	Ensure that the software support for new devices (which usually requires an IPL to implement) is installed in a timely manner. See 10.1.12, "Software support for new devices" on page 115 for further details.	

10.1.1 Dynamic I/O configuration changes

HCD is used to define an I/O configuration to the operating system, or software, and the channel subsystem, or hardware. Assuming the server's RESET profile has the "dynamic I/O" flag enabled, most changes to the I/O Configuration can be activated dynamically. This is done either using the ACTIVATE option from the HCD ISPF panels, or by using the MVS `ACTIVATE` command, provided the devices that are being changed support dynamic reconfiguration.

To determine if a device can be defined as dynamic, refer to the HCD device definition panel. If the panel shows that the device can be dynamic, it supports dynamic reconfiguration. Nearly all modern devices support dynamic reconfiguration; it really depends on what HCD device type is used to define the device. For example, if a new piece of hardware is defined to the operating system as a 3286 (a dated piece of hardware which does not support dynamic reconfiguration), then it will not support dynamic reconfiguration.

Dynamic device exceptions: Any display devices that are to be used as MCS consoles are required to exist in CONSOLxx and be defined as a UCB at IPL time.

Channel-to-Channel Connectors (CTCs) can be added and deleted, but cannot be removed from GRS rings, unless the CTC is managed by Cross-System Coupling Facility (XCF). If XCF is managing the CTCs, you must stop XCF from using them before dynamically deleting the CTCs.

Also, it is required that the IODF and IOCDS tokens match, from a hardware and software point of view. The following message at IPL time indicates an unmatched condition:

```
IOS505A DYNAMIC I/O CONFIGURATION CHANGES ARE NOT ALLOWED, THE HARDWARE AND
SOFTWARE CONFIGURATION DEFINITIONS DO NOT MATCH
```

If they do not match, you may be able to resynchronize by doing a SOFT ACTIVATE to move z/OS to an IODF whose token matches the hardware token. If a soft activate is not possible (because non-dynamic devices have been removed, for example), then an IPL is required, bringing the system up using the IODF that matches the hardware token.

In LPAR mode, any partition can activate changes that can affect the hardware configuration for all the other logical partitions on that server. To avoid unwanted disruption across LPARs, such as removing a device in use by another partition, follow this procedure:

1. In all but one of the logical partitions, activate the new software definitions using the MVS `ACTIVATE` command with the `SOFT=VALIDATE` option. This confirms that the components being deleted are not in use on any LPAR.
2. In the remaining partition, make both the hardware and software changes. This can be done with the `ACTIVATE` command *without* the SOFT option.

Note: If you issue the `ACTIVATE` command with the `TEST` option and the system detects no errors, there is still no guarantee that `ACTIVATE` will work without the `TEST` option.

Also, any dynamic change that causes a device's UCB to be deleted and added again causes the device's MIH time interval to be reset to the default MIH setting for its device class. To re-establish the previous MIH interval, issue the MVS `SETIOS MIH` command when the dynamic change has completed.

Be aware of the following prior to using the MVS **`ACTIVATE`** command:

- Make sure you activate the correct IODF. If you activate an incorrect or invalid IODF, you will receive error messages, and a procedure to inactivate the IODF and re-activate the previous IODF is required.
- Dynamic deletion of hardware components requires the use of the FORCE option and the device to be defined as dynamic. HCD only allows you to delete I/O components when you specify `YES` in the field `Allow hardware deletes(FORCE)` on the Activate New Configuration panel, or if you specify the FORCE option on the **`ACTIVATE`** command.

For more information on the use of the dynamic reconfiguration capability, refer to:

- *z/OS V1R6.0 HCD Planning,* GA22-7525
- IBM Redbook *MVS/ESA HCD and Dynamic I/O Reconfiguration Primer,* SG24-4037

10.1.2 Planning for growth in HSA

Prior to the introduction of the z990 server, you could specify a percentage expansion factor to allow for growth in HSA. This enabled device configurations to be changed dynamically. The size of the expansion factor should be determined by the number of configuration changes you expect to make between power-on resets (PORs). A larger expansion factor will allow for more changes.

On the z990 and later servers, rather than specifying an HSA expansion factor, you specify in HCD the maximum number of devices you plan to have in the configuration for each Logical Channel SubSystem (LCSS). Changing this value will require a POR of the server, so you should ensure that the value is large enough to accommodate the configuration you plan to implement before the next planned POR. The HCD dialog shows, for each channel subsystem, the maximum number of allowed devices and the actual number of defined devices. The same is true for the Channel Subsystem Summary Report of HCD.

The IOCP program can be used in batch to see how many subchannels an IOCDS will require, or you can use the MVS **`D IOS,CONFIG(HSA)`** command to determine how much HSA is available to grow based on the current IOCDS requirements. The information from these commands can be used with the HSA Estimator tool, available on ResourceLink.

Tip: If the server has sufficient available storage, specify the maximum number of devices for each LCSS right from the beginning, because this will avoid having to ever do a POR to provide more HSA for new devices.

10.1.3 Dynamic logical partitions

z/OS 1.5 (with APARs OA03689 and IR52441) on z990 or later servers provides the ability to pre-define spare LPARs that can subsequently be renamed and brought into service without a POR. Prior to this, a POR, affecting every LPAR on that server, would have been required to add, change, or remove LPAR definitions.

To prepare for future additions and deletions of LPARs, the partition is defined in HCD using a name of asterisk (*). This is replaced later with the correct logical partition name and the configuration dynamically activated.

Until the required hardware and software levels are installed, a “dummy” LPAR can be defined. You can dynamically add devices to that LPAR; however, the LPAR name cannot be changed without a POR.

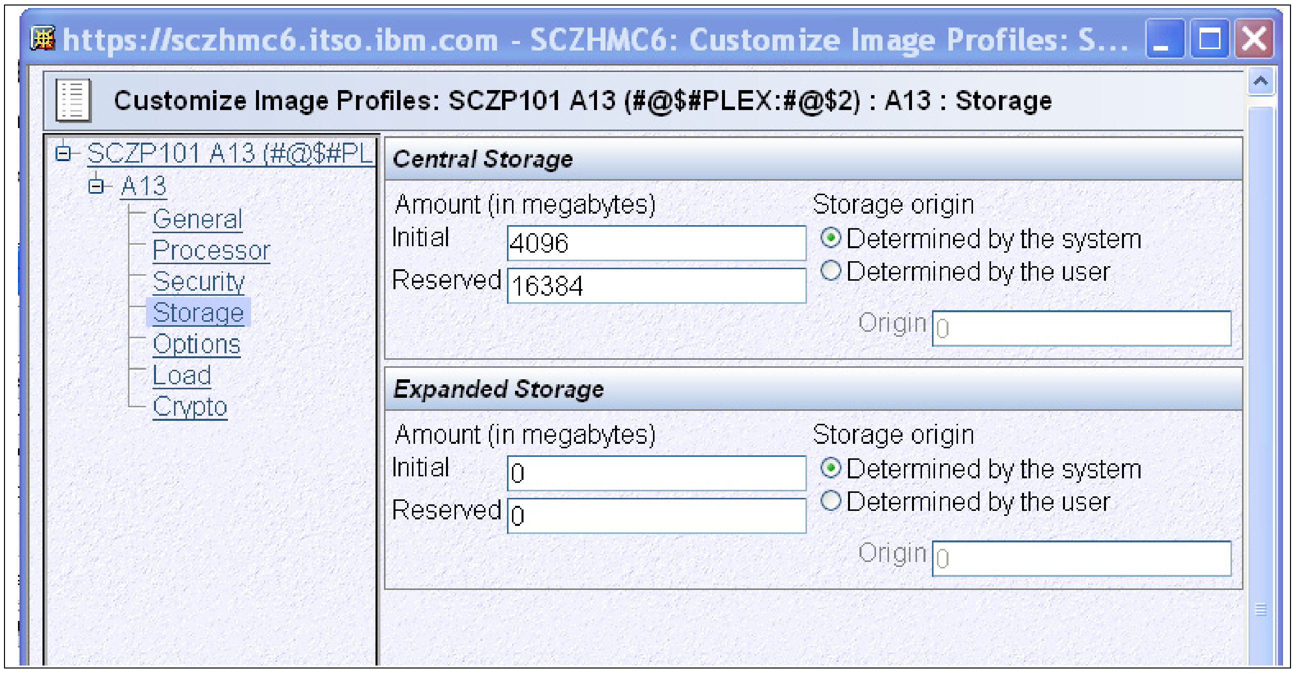
Tip: We recommend specifying the maximum number of LPARs right from the beginning. The only cost is some additional space in the IODF. Then, if you need to add an LPAR in the future, no POR is required. And if you *don't* ever need the additional LPAR, there has been no cost anyway.

For more information, refer to:

- *z/OS HCD User's Guide,* SC33-7988
- The chapter entitled "Reserved logical partitions" in *IBM System z9 109 Configuration Setup,* SG24-7203

10.1.4 LPAR Dynamic Storage Reconfiguration (DSR)

On the Storage tab in the LPAR Image profile on the HMC, it is possible to specify a "Reserved" amount of central and expanded storage that can subsequently dynamically be brought online to the LPAR. This is shown in Figure 10-1.


https://sczhmc6.itso.ibm.com - SCZHMC6: Customize Image Profiles: S...
SCZP101 A13 (#@$#PL
A13
General
Processor
Security
Storage
Options
Load
Crypto

Customize Image Profiles: SCZP101 A13 (#@$#PLEX:#@$2) : A13 : Storage

Central Storage

Amount (in megabytes)
Initial: 4096
Reserved: 16384
Storage origin: [✓ Determined by the system] Determined by the user
Origin: 0

Expanded Storage

Amount (in megabytes)
Initial: 0
Reserved: 0
Storage origin: [✓ Determined by the system] Determined by the user
Origin: 0

Figure 10-1 Specifying Reserved storage

The storage does not necessarily have to be available at IPL time, only when the storage is brought online via a command similar to the following:

```
CF STOR(E=nn),ONLINE
CF ESTOR(E=nn),ONLINE
```

Note: Expanded storage is only usable by z/OS when operating in 31-bit mode. As of z/OS 1.6, z/OS only operates in zArchitecture mode, meaning that it supports more than 2 GB of Central storage (the exact amount supported depends on the z/OS release), but it no longer supports expanded storage.

If you have a pool of unused storage on the server, it can be moved between LPARs in this way. If all your storage is already allocated, but you wish to be able to move storage between

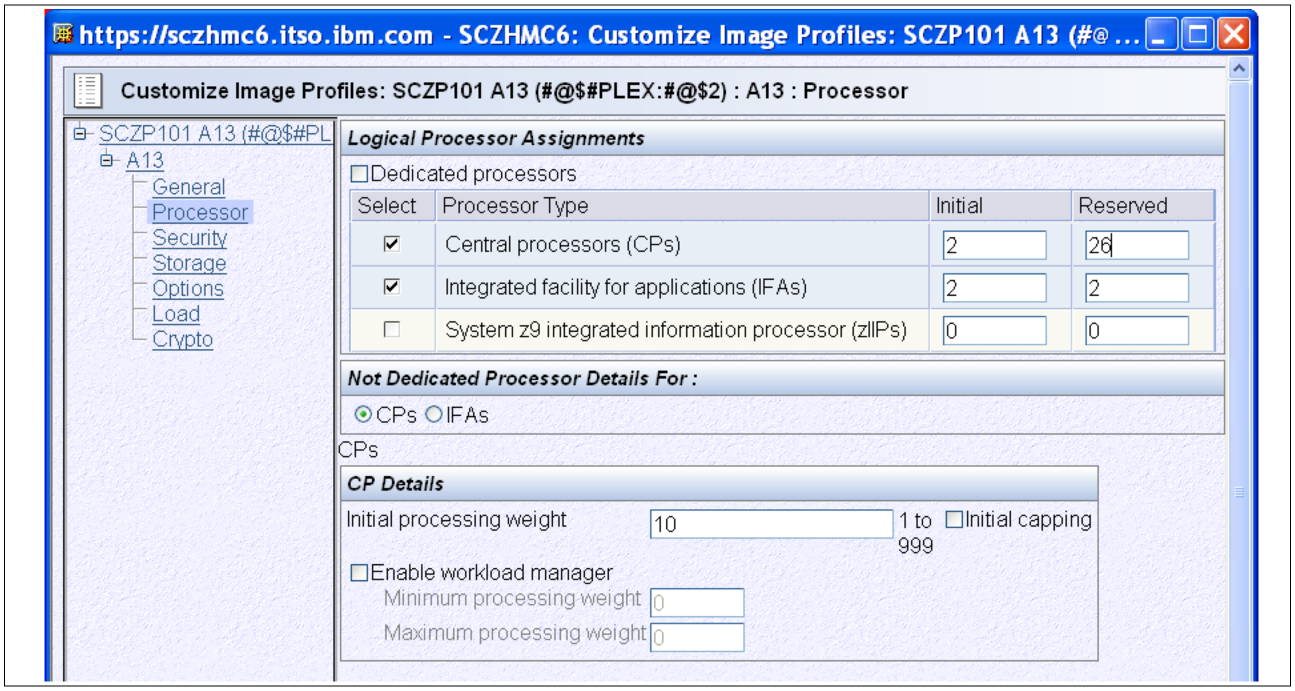
LPARs, you must specify a value on the RSU parameter in the IEASYSxx member of Parmlib. The RSU parameter determines how much storage you can configure offline to that system.

For more information on moving storage between LPARs, refer to:

- "Dynamic Storage Reconfiguration" in *System z9 Processor Resource / Systems Manager Planning Guide,* SB10-7041

10.1.5 Defining Reserved CPs

In addition to being able to pre-define storage that can subsequently be brought online when needed, you can also pre-define CPs that can be brought online at a future time. When you define an LPAR in the HMC Image profile (as shown in Figure 10-2), you can specify both Initial and Reserved CPs (and zAAP/IFAs and zIIPs, depending on the server capability). The number you specify in the `Initial` field is the number of logical CPs that will be online to this LPAR when it is activated.


https://sczhmc6.itso.ibm.com - SCZHMC6: Customize Image Profiles: SCZP101 A13 (#@ ...
SCZP101 A13 (#@$#PL
A13
General
Processor
Security
Storage
Options
Load
Crypto

Customize Image Profiles: SCZP101 A13 (#@$#PLEX:#@$2) : A13 : Processor

Logical Processor Assignments

☐ Dedicated processors

Select	Processor Type	Initial	Reserved
✓	Central processors (CPs)	2	26
✓	Integrated facility for applications (IFAs)	2	2
☐	System z9 integrated information processor (zIIPs)	0	0

Not Dedicated Processor Details For :

[✓ CPs] IFAs

CPs

CP Details

Initial processing weight: 10 (1 to 999) ☐ Initial capping

☐ Enable workload manager

Minimum processing weight: 0

Maximum processing weight: 0

Figure 10-2 Specifying Reserved CPs

In addition, more CPs, up to the number specified in the `Reserved` field, can be brought online to the LPAR without an IPL. There is no cost (either financially or in terms of overhead) in defining Reserved CPs, so INITIAL + RESERVED should equal the *smaller* of:

- The maximum number of CPs supported by the operating system, or
- The maximum number of CPs supported by that model of server.

For example, if you had an LPAR running z/OS 1.6 on a 2084-710, you might specify values of 2 Initial CPs, 26 Reserved CPs, 2 Initial IFAs, and 2 Reserved IFAs (giving a total of 32 PUs), as both z/OS 1.6 and 2084 support a maximum of 32 engines.

For more information about the use of Reserved CPs, refer to:

- *System z9 Processor Resource / Systems Manager Planning Guide,* SB10-7041

10.1.6 Multiple logical channel subsystem support (LCSS)

The z890, z990, and later processors support multiple LCSSs. A z/OS system with the z990 exploitation PTFs can do a dynamic ACTIVATE of hardware for any LCSS. However, a z/OS system with just the compatibility PTFs installed can only use a dynamic ACTIVATE to make hardware changes to an LCSS of zero (with some restrictions).

If you must make a hardware change to anything in an LCSS other than 0, you must do a power-on reset (POR), or do the hardware ACTIVATE from a system that has the z990 exploitation PTFs installed.

For an introduction to logical channel subsystems, see “z9-109 and multiple Logical Channel Subsystems” in *IBM System z9 109 Configuration Setup,* SG24-7203.

Tip: Adding an LCSS requires a POR. Therefore, define all possible LCSSs when you create the initial IODF for that server. You can then start to use that LCSS in the future without a POR at that time.

If you define an empty LCSS (that is, an LCSS without any logical partitions), you get an error message (CBDA658I) when you create a production IODF. You have to define at least one partition in each LCSS. If you define a named partition without channel path access, you get a warning message (CBDA857I). If you define a reserved partition, you do not get a message during creation of the production IODF.

See the following for further information about this topic:

- *z/OS V1R6.0 HCD Planning,* GA22-7525
- IBM Redbook *IBM System z Connectivity Handbook*, SG24-5444
- IBM Redbook *IBM System z9 109 Configuration Setup,* SG24-7203

10.1.7 Multiple subchannel sets

Starting with the z9 servers and z/OS 1.7, you can define a second subchannel set with ID 1 (SS 1) in addition to the existing subchannel set (SS 0) in all channel subsystems. With this additional subchannel set, you can configure more than 2*63K devices for a channel subsystem (specifically, you can define 63.75K devices in SS 0 and 64K-1 devices in SS 1). SS 1 can only contain Parallel Access Volume (PAV) alias devices (device types 3380A, 3390A) to SS 1. Device numbers may be duplicated across channel subsystems and subchannel sets. You can dynamically move alias devices from SS 0 to SS 1.

For more information, refer to:

- “Multiple subchannel sets (MSS)” in IBM Redbook *IBM System z9 109 Configuration Setup,* SG24-7203
- *z/OS HCD User’s Guide,* SC33-7988

10.1.8 Overdefining channel paths

As of z/OS 1.5, you can define more channel paths than are physically installed on the processor, to prepare for possible future channel card upgrades.

On z990, z890, and subsequent servers, define the PCHID value with an asterisk (*). When the channels are subsequently installed, replace the asterisk (*) with a valid PCHID value.

10.1.9 CBU, CUoD, and OOCoD support

Capacity Backup Upgrade (CBU) provides you with the ability to very quickly add additional PUs to be used in case of a disaster. Depending on the server type, CBU supports some or all PU types. The CBU contract provides for a fixed number of tests during the period of the contract. GDPS includes support to automatically enable the CBU PUs and to configure them online following a disaster.

Capacity Upgrade on Demand (CUoD) provides the ability to add PUs to the configuration non-disruptively, if sufficient planning has been done on both the hardware and software side. To enable the additional PUs, you would order and pay for an MES upgrade, and your PSR would subsequently enable the additional engines.

On/Off Capacity Upgrade on Demand (OOCoD) takes CUoD one step further. OOCoD is designed to provide cost-effective additional capacity to be enabled at peak processing times, (for month-end or year-end processing, for example). The difference is that the additional PUs are turned on and off under customer control, rather than involving an IBM representative.

For further information about these features, see the following:

- *IBM System z9 109 Technical Introduction,* SG24-6669
- *IBM System z Connectivity Handbook*, SG24-5444
- The August 2000 issue of the Hot Topics newsletter, available on the Web:

  `http://publibz.boulder.ibm.com/epubs/pdf/e0z1nl02.pdf`
- The Capacity Backup Upgrade Web page:

  `http://www.ibm.com/systems/z/cbu.html`
- The Capacity on Demand Web page:

  `http://www.ibm.com/servers/eserver/about/cod/`

10.1.10 Planahead

Hardware "planahead" is a capability to configure a new server in a manner that will allow subsequent upgrades, that would otherwise be disruptive, to be implemented dynamically. When you order a new server, you can work with your IBM representative to identify your planned configuration at some point in the future. The server can then be built with an infrastructure that will support dynamic addition of the components to get you to your planned configuration.

For example, you might be ordering a small number of FICON® channels now, but foresee that you will need many more in 18 months time. The installation of that number of channels might require an additional I/O cage. Installing that cage would normally be a disruptive process, so you can specify that you want the cage installed in the new server. The channels will not be installed until you purchase them in the future; however, at that time the existence of the additional cage ensures that the channels can be installed dynamically.

The price of the new server will reflect the presence of any additional hardware identified by the planahead process, so you will be paying for that hardware earlier than would otherwise be the case—however, you are avoiding a future outage. Therefore, you need to make a business decision as to whether the outage avoidance justifies the earlier capital investment.

10.1.11 Considerations for OSA code refresh

While patches for OSA adaptors can be dynamically loaded onto the server, in order to actually activate the patch, the OSA adaptor must be taken offline to *all* LPARs using it. To ensure that this does not result in a loss of access to the system (which could be viewed as an outage to end users), configure two OSA adaptors to every LPAR. Together with the use of Virtual IP Addresses, this will enable you to move all the load to one OSA while the other is recycled, thereby avoiding any lack of connectivity during this time.

For more information about configuring OSA adaptors for high availability, refer to:

- IBM Redbook *IBM System z Connectivity Handbook*, SG24-5444

10.1.12 Software support for new devices

For most customers, the software support for a new device type is available long before they install their first instance of that new device. Whereas most new devices can be installed and implemented dynamically, the support for those new devices (as delivered in PTFs) usually requires an IPL to implement.

Therefore, in order to get the maximum benefit from the dynamic activation capability of the hardware and software, you should make a special effort to ensure that the latest such PTFs get applied every time you have a maintenance cycle. The PTFs to deliver this new support will be documented in the PSP bucket for the new device. In addition, such PTFs are often marked as NEW FUNCTION.

Additionally, communication-related devices, such as Open Systems Adaptors (OSAs), usually require specific support in VTAM and/or TCP/IP. This support will usually require, at a minimum, a restart of VTAM or TCP, if not an IPL with CLPA.

Therefore, in addition to the HCD PTFs for the new device support, ensure that you pick up the VTAM and TCP service at the same time. Having all this support installed in advance will ensure that you can subsequently install and activate the new devices dynamically.

For more discussion on this topic, refer to 11.2.2, “Sample procedure for installing a new device” on page 126.

11

Miscellaneous and tools

This chapter describes miscellaneous changes made to recent releases of z/OS and OS/390 to assist in reducing the need for planned outages and extend the requirement between times of IPLs. It also highlights and products that we have found to be very useful, and incudes a brief discussion of systems management considerations.

11.1 Miscellaneous and tools checklist

Table 11-1 contains a checklist of IPL avoidance features and tools that could not be categorized in any other section of this paper. It includes a brief description and, if applicable, the z/OS release it became available.

Table 11-1 Miscellaneous and tools checklist

Features/Enhancements	Available in z/OS	Description	Completed
Dump considerations	All	It is important to achieve a balance between capturing enough information to allow diagnosis, and minimizing the performance impact of taking a dump. See 11.1.1, “Dump considerations” on page 118.	
IBM Health Checker for z/OS	1.7	A system health checker is provided as part of z/OS 1.7. It may also be installed on z/OS 1.4 and above. See 11.1.2, “IBM Health Checker for z/OS” on page 121 for more details.	
Symbolic Parmlib Parser	1.4	This is an IBM-supplied tool to perform a basic syntax check that resolves symbols. See 11.1.3, “Symbolic Parmlib Parser” on page 123 for more details.	
IMAGE Focus and Stand Alone Environment	All	These are third party products that provide IPL avoidance capabilities through syntax-checking and recovery facilities. See 11.1.4, “IMAGE Focus” on page 124 for more detail	
MVS eXtended Information (MXI)	All	This is an ISPF-based freeware application that enables the user to display configuration information about the active z/OS system. See 11.1.5, “MVS eXtended Information (MXI)” on page 125 for more detail.	
CHPID Mapping tool	All	This tool can help you map CHPIDs in preparation for avoiding single points of failure and IPLs in the future. See 11.1.6, “CHPID Mapping tool” on page 125 for more detail.	
Systems Management	All	There are various process improvements that can help reduce outages. See 11.2, “Systems Management considerations” on page 126.	

11.1.1 Dump considerations

The performance of dumps, and the parameters you use to control dumps, are important because of the impact on the system while the dump is being taken. On one hand, you want the dump to contain as much information as possible in order to minimize the risk that you will need to recreate the problem to get sufficient diagnostic information. But on the other hand, the duration of the dump can be impacted by the amount of information the dump will contain. This section contains suggestions to help you find an acceptable balance of dump contents and dump duration.

SVC dumps

One of the most important aspects of good dump performance is an ample and robust paging subsystem. Because of the low paging rates typical on many systems today, many customers have reduced the size and number of page data sets.

When the system takes an SVC dump, the dump contents are initially saved in storage before being moved to the dump data sets. For a large dump, this can result in a sudden and dramatic increase in the amount of storage required. If there has not been much paging taking place, it is likely that many of the pages in storage do not have a backing copy in the page data sets.

So, in order to use those pages to contain the dump information, the pages must be written to the page data sets. It is not uncommon to see dumps of 1 GB or more. In order for such dumps to be completed in a timely manner, you must have a paging subsystem that will have enough space for all this data *and* deliver acceptable performance. This means having enough page data sets, spread across multiple control units, preferably on FICON channels.

You can find information that will help you size local page data sets in "Example 4: Sizing Local Page Data Sets" in *z/OS MVS Initialization and Tuning Guide,* SA22-7591.

Parallel Access Volumes

The IBM recommendation was always that there should only be one page data set per DASD volume. This was in order to maximize the value of the suspend/resume logic used with paging I/Os. However, the use of DASD volumes that support PAV *may* enable you to have more than one page data set per volume and still get the benefit of suspend/resume.

This is an important consideration if your strategy is to move away from smaller volume sizes (3390-3 or smaller), but yet do not want to have many volumes that are only partly used. At the time of writing, the largest possible page data set is 4 GB, so if you want to fully utilize larger volumes, you will need to have multiple data sets on the volume.

An important point that you may not be aware of is that the volume *must* be defined with *dynamic* aliases. The auxiliary storage manager will not use static PAVs and therefore will use a single UCB for all page data sets on that volume—thereby losing the benefit of suspend/resume.

Dump data sets

One of the most common problems we encounter when looking at customer problems is that the related dump was not captured at all, only a partial dump was captured, or the dump was captured but overwritten before it could be processed.

To address this, we strongly recommend that you use dynamically-allocated dump data sets. These are very easy to set up and manage and should decrease the likelihood of dumps not being captured or being overwritten too quickly.

We also recommend using extended format data sets for the dump data sets. These will allow you to have larger data sets, they can be striped for better performance, and they support compression, so more dump data can be held in the same amount of disk space.

For more information, refer to "Using Extended Format Sequential Data Sets" in *z/OS MVS Diagnosis: Tools and Service Aids,* GA22-7589.

GRSQ information

In a GRS Star environment, each system only has information about the resources *it* has serialized. However, in a dump, it may be useful to get a *sysplex-wide* view of contention and resource serialization. The GRSQ option on the SDATA keyword indicates that all GRS control blocks and other GRS storage should be included in the dump. Because of the distributed nature of GRS Star, it is necessary for the dump process to issue GQSCANs to gather information from the other sysplex members. This can generate a large amount of XCF traffic and can potentially run for a long time, thus elongating the dump process, especially if the target systems are slow to respond.

To improve the performance of this process and reduce its impact on system service levels, APARs OA06591 and OA07975 have been released. These APARs do the following:

- They make all tasks within the last dumping address space dispatchable before the GRS sysplex data collection starts.
- They add a new keyword to the GRSCNFxx member that lets you control what level of information is gathered: all information; just global contention information; or only local information.

We recommend, at a minimum, applying the PTF for APAR OA06591 to make the tasks dispatchable as early as possible. The recommended settings for the GRSQ parameter in the GRSCNFxx member are:

ALL	If you are running GRS NONE or ring mode
CONTENTION	If you are running GRS Star mode

Dump suppression

There should be no need to capture multiple dumps for the same problem. Eliminating unnecessary dumps avoids wasting disk space for the dump data sets and, more importantly, avoids the impact on system service while the dump is being taken.

z/OS includes a function called Dump Analysis and Elimination. This function saves information about all dumps in a data set called SYS1.DAE. Every time a dump is taken, and assuming that DAE is enabled and the data set exists, DAE will analyze the dump to see if it is a duplicate of a previous dump. Because the data set can be (and should be) shared by all systems in the sysplex, DAE can determine if a duplicate dump has been taken anywhere in the sysplex. If it has, the dump will be suppressed.

More information on DAE can be found in “Using DAE to Suppress Dumps” in *z/OS MVS Diagnosis: Tools and Service Aids,* GA22-7589.

Standalone dumps

While not strictly related to IPL avoidance, we provide some information here on standalone dumps because there are things you can do to speed up standalone dump processing, and the duration of the dump determines how long your system will be unavailable.

Our first recommendation is to ensure that the standalone dump data sets are placed on the fastest available DASD volumes. Testing within IBM has shown that the speed of the DASD has the largest single impact on the dump elapsed time.

The next recommendation is to use the support delivered in APAR OA04140 to stripe the dump data sets across several volumes. This APAR supports up to 32 stripes. To use this support, the data sets must be defined as sequential-extended data sets. This means:

- The data sets must be SMS-managed.
- They must have a DATACLAS that specifies no compression.
- They should have a STORCLAS that specifies a sustained data rate of zero (to suppress SMS striping).

You must use the AMDSADDD exec to allocate and format the data sets. You can explicitly request a STORCLAS and/or DATACLAS when AMDSADDD is invoked.

We also recommend using FICON channels for the dump data set devices. Tests in IBM showed about a 20% reduction in dump elapsed time using FICON channels compared to an equivalent configuration using ESCON® channels.

Finally, make a conscious decision about how much data you will save in the dump. The standalone dump process has two phases. The first phase dumps the contents of the LPARs processor storage. At the end of this phase, a message is issued to inform you. The next phase reads the paged-out information from the page data sets and copies that to the stand alone dump data sets. This second phase will normally take far longer than the first phase.

Therefore, you need to decide whether to stop the dump after processor storage has been dumped, or to let the dump run to completion. Stopping the dump after phase 1 will obviously reduce the dump elapsed time. However, you run the risk that information required to debug the problem has been paged out and therefore will not be in the dump.

On the other hand, letting the dump run to completion will ensure that you have all required information. However, there is a good chance that all the information required to debug the problem was actually resident in processor storage, meaning that the dump really took longer than necessary. So you have to decide whether to go for the reduced elapsed time with the risk that you may have to recreate the problem, or that you can live with the increased downtime in return for the security of gathering all required information.

11.1.2 IBM Health Checker for z/OS

Elongated periods between IPLs bring with them their own unique set of problems. The IBM Health Checker for z/OS has been designed to identify potential problems before they impact system availability or, in worst cases, cause outages. It checks the current active z/OS and sysplex settings and definitions for a system and compares the values to those suggested by IBM or defined by the installation.

IBM Health Checker for z/OS is not meant to be a diagnostic or monitoring tool, but rather a continuously running tool that finds potential problems. It produces output in the form of detailed messages to highlight both potential problems and suggested resolutions, so that action can be taken in a timely manner.

The IBM Health Checker for z/OS consists of two components: the framework and the checks.

Framework

The IBM Health Checker for z/OS framework provides a structure for checks to gather system information and mechanisms to report their findings. The framework is a common and open architecture, supporting check development by IBM, independent software vendors (ISVs), and users.

Checks

Checks are routines that look for component-, element-, or product-specific z/OS settings and definitions, checking for potential problems. A base set of checks is supplied with z/OS 1.7. Additional checks will be shipped in the service stream as they become available. Checks for z/OS 1.4 to z/OS 1.6 are only available via the service stream. The specific component or element owns, delivers, and supports the checks.

Checks are separate from the IBM Health Checker for z/OS framework. A check analyzes a configuration using the following:

- Changes in settings or configuration values that occur dynamically over the life of an IPL. Checks that look for changes in these values should run periodically to keep the installation aware of changes.
- Threshold levels approaching the upper limits, especially those that might occur gradually or insidiously. These should be monitored and historical trends analyzed.
- Single points of failure in a configuration.
- Unhealthy combinations of configurations or values that an installation might not think to check.

Check settings (interval, severity, parameters, and so on) can be modified or overridden by either using SDSF via a MODIFY command, or by statements in the HZSPRMxx Parmlib member. The HZSPRMxx Parmlib member should always be updated if either dynamic technique is used and the changes are required across restarts. These are called *installation updates*.

When a check flags an exception, a summary WTO is issued to alert the operator of the problem. A check will also write messages to its message buffer. Messages written to a check's message buffer will include details about an exception, configuration details, and “everything is OK” messages. If the logging function of IBM Health Checker for z/OS is enabled, check message buffers will be saved in a designated log stream. The message buffers for checks can be viewed though the CK option of SDSF, and can be copied/printed using the HZSPRINT utility.

The IBM Health Checker for z/OS works best when run continuously on a system so that it can report when the system has changed. When an exception occurs, it should be resolved so that the same exceptions do not appear continuously.

The IBM Health Checker for z/OS (FMID HZS7720) is shipped as part of z/OS 1.7. It is also available as a Web deliverable: IBM Health Checker for 1.4/5/6 of z/OS and z/OS.e. The Web deliverable is functionally identical to the integrated version and should not be confused with the prototype (IBM Health Checker for z/OS and Sysplex). The Web deliverable version is available on the following download site for z/OS 1.4, 1.5, and 1.6:

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

IBM Health Checker for z/OS can run on a Parallel Sysplex, monoplex, or XCF local mode environment running z/OS 1.4, 1.5, 1.6 and 1.7.

Checks are available both as an integrated part of a z/OS release or separately, as PTFs. Many new and updated checks will be distributed as PTFs, so that they are not dependent on z/OS release boundaries and can be added at any time. To identify checks that have been provided in PTFs, use the procedure described in the section entitled “Obtain checks for IBM Health Checker for z/OS” in *IBM Health Checker for z/OS: User’s Guide,* SA22-7994.

The IBM Health Checker for z/OS has a comprehensive manual documenting how to set up the product and customize the checks to an individual installations requirements.

For more information see the following:

- PSP Buckets, *z/OS and z/OS.e Planning for Installation,* GA22-7504
- *IBM Health Checker for z/OS: User’s Guide,* SA22-7994

Recommendation: Allow the IBM Health Checker for z/OS to run on a continuous basis to provide maximum lead time on warnings of problems. This alerts the user to take dynamic remedial action if available.

11.1.3 Symbolic Parmlib Parser

The Symbolic Parmlib Parser allows for the verification of symbolic substitutions without doing an IPL. The Parser is in SYS1.SAMPLIB and is activated as follows:

```
TSO EX 'SYS1.SAMPLIB(SPPINST)' '''SYS1.SAMPLIB(SPPPACK)'''
```

This will create and populate four data sets, where `myid` is the value specified on the TSO PROF PRE(myid) command:

```
myid.PARMLIB.EXEC
myid.PARMLIB.PANELS
myid.PARMLIB.MESSAGES
myid.PARMLIB.NOTES
```

After the data sets are created, issue the following command:

```
TSO EX 'myid.PARMLIB.EXEC(SYSPARM)'
```

The myid.PARMLIB.NOTES file, member WORKFLOWS, gives an overview of the tool's capabilities. These include:

- LOADxx processing
- Parmlib symbolic preprocessing
- Parmlib member selection list
- Obtaining a Parmlib member selection list from a concatenation of Parmlibs

The tool has several limitations:

- The tool does not support editing of SYSn.IPLPARM data sets. The tools can be used to generate the member in SYS1.PARMLIB or high_level_qualifer.PARMLIB and then a copy of the member can be placed into the appropriate SYSn.IPLPARM data set.
- The tool assumes that any SYSn.IPLPARM data set is on the same volume as the specified Parmlib data set.

  It does not support the concept of a separate IODF and SYSRES volume, nor does it use Master Catalog to locate a SYSn.IPLPARM data set.
- Although the code can detect the presence of a VSAM data set, it does not have the ability to read one. Therefore, the IODF data set is only checked for existence. Its contents are not verified.
- The Master Catalog information is not verified.
- When in EDIT mode, the data in a member is not altered until it is saved. If running in split screen mode, the effects of the change will not be seen until the EDIT mode changes are saved.

For more information see the following:

- Symbolic Parmlib Parser Appendix in *z/OS MVS Initialization and Tuning Reference,* SA22-7592

Note: The Symbolic Parmlib Parser has limited functionality. Other products exist in the marketplace (for example, IMAGE Focus from New Era Software) that expand on the Symbolic Parmlib Parser syntax-checking capabilities and provide enriched functionality.

11.1.4 IMAGE Focus

IMAGE Focus is a software product from NewEra Software, Inc. It uses proprietary technology to track changes to operating system parameters, and provides the ability to do a “Virtual IPL” of a z/OS image to ensure that a real IPL will not encounter invalid parameters.

Image Focus runs as a started task under z/OS. It works by providing a series of “inspectors”. These inspectors (OS-Inspector, Sysplex-Inspector, JES-Inspector and VTAM-Inspector) perform the actual checking of each component.

IMAGE Focus is composed of three components, each of which may be optionally installed and operated independently of the others. These components are:

- Recovery

  The IMAGE Focus recovery started task is designed to maintain its own independent communications subsystem and provides ISPF application support to a single locally-attached non-SNA 3270 console. This provides the ability to log on, check, and fix problems when VTAM, JES, or TSO are unavailable at IPL time. This potentially avoids the need to perform unnecessary IPLs following the discovery of a problem; that is, an installation may “forward” fix problems found early in the IPL process.

- Multi-user

  When installed as a VTAM application to support multiple simultaneous users, IMAGE Focus maintains a multi-user started task. This provides the ability for users to log on using their external security product-controlled user IDs and check, browse, and edit system parameters.

- Background

  The IMAGE Focus background started task reports IPL changes that would result in future IPL failures to a designated user or group through the TSO Broadcast Facility or via e-mail. These notices are sent at intervals controlled by IMAGE Focus, or optionally by the installation’s job scheduling package.

The checking process starts with the LOADxx member whose value is entered by the user, or as a variable if run in batch. It checks whether IPL text exists on the IPL volume and whether the SYS1.NUCLEUS data set can be opened. It processes each PARMLIB and PROCLIB member for syntactical correctness and related data sets for referential integrity and attribute characteristics. This process may be run in the foreground by a user, in batch, or on a continuous basis.

During this continuous inspection, the inspector will determine if there are any problems with system definitions, warn of potential failures, and compile reports on suspect components. These reports identify errors in critical data sets, document their locations, and catalog the data sets that may prevent a successful future IPL.

This checking extends to JES, VTAM, and TCP/IP, and the product provides the ability to create installation-specific checking routines over and above those which are supplied.

The product provides a “what if” option that includes support for new releases of the operating system. This allows the user to plug existing IPL parameters into the next version of z/OS and verify whether they are still viable.

For more information see the following Web site:

`http://www.newera.com`

11.1.5 MVS eXtended Information (MXI)

IBM FLASH10273 mentions a non-IBM tool called MXI. MXI is an ISPF-based application which also supports batch and remote (via TCP/IP) site execution. The application is designed to present information of various system attributes and control blocks in an easy-to-read format.

The FLASH makes use of the product to ascertain cross-memory connections. Filtering is available using ISPF-like masking characters, as is the ability to “point-and-shoot” on various fields to drill down to more detailed displays.

The following items are a subset of the displays MXI can produce:

- Active Address Spaces and ASVT Slot Usage
- Allocated data sets for any Address Space
- Common Storage Usage By Address Space or Subpool
- Orphaned Common Storage
- Cross Memory Connections
- CPU and LPAR Information
- Enqueue Requests and Contention
- LLA Module Statistics
- Memory Contents of ANY Address Space
- Memory Delete Queue
- Real and Auxiliary Storage Usage
- Subsystems
- SVCs and PC Routines
- Sysplex Information
- WLM Information

For more information see the following:

`http://www.mximvs.com`
`http://www.rocketsoftware.com`

11.1.6 CHPID Mapping tool

The CHPID Mapping tool is used to validate a working IODF for all Physical Channel Identifier (PCHID) values. One of the functions of the tool is to check for single points of failure in the channel subsystem. The tool can be downloaded from ResourceLink at:

`http://www.ibm.com/servers/resourcelink`

More information is available in *z/OS HCD User’s Guide,* SC33-7988.

11.2 Systems Management considerations

In addition to functions and facilities provided in the hardware and software, the way you manage your systems can have a fundamental impact on how you manage planned outages. This section discusses topics that you should give consideration to.

11.2.1 Planning ahead

Many events can lead to a planned outage: you may need to install new hardware or software; you may need to apply service to the operating system or its components; or you may need to address storage fragmentation or “storage creep”.

However, careful planning can help you postpone or even eliminate a planned outage. In 10.1.10, “Planahead” on page 114 we discuss the hardware planahead function that can help you avoid PORs. But there is also a system setup consideration.

Let’s say that you have a product that has a storage leak. Over time, this will result in more and more ECSA being used. The only immediate way to address this is to IPL the system. Of course, you should get the owner of the code to fix the storage leak. But until it is fixed, you can extend the time between IPLs by adjusting the size of ECSA.

To understand how much you need to adjust it by, you need to monitor the growth so you can accurately predict how much additional common storage you need to allocate in order to increase the interval between IPLs. This methodology also applies to other system resources such as storage fragmentation, non-reusable address spaces, and so on: measure, extrapolate, adjust, and monitor.

Tip: It is not uncommon to find installations that regularly IPL to address storage creep (because they have always done that), yet when they actually investigate the problem they find that it no longer exists. In fact, one of the most common reasons given for frequent IPLs is “because we have always done it this way”.

Although there are certainly valid reasons for IPLing a system, it is good practice to revalidate the reasons for your IPL frequency on a regular basis—at a minimum after every time you update the service level of the system or major subsystems.

11.2.2 Sample procedure for installing a new device

Let’s say you want to install some new communications device - a brand new Open Systems Adaptor, for example. There are a number of things that must be considered before the device can be used by the system:

- The hardware must be installed, including any supporting hardware, like power supplies, I/O cages, mother cards, and so on.
- The server configuration (IOCDS) must be updated to indicate which LPARs can use the new device.
- The new device must be defined correctly to the operating system.
- The software that will use the device (TCP, for example) must be running at a level that understands how to interface with this new device.
- The device must be defined to that software so it can be brought into use.

All of these steps can be carried out dynamically *as long as* sufficient advance planning has been done.

A typical sequence for achieving this might be:

1. When the server is ordered from IBM, you know that you will need to increase its communications capabilities in the future, so you go through the planahead process with your IBM representative, ensuring that all the required supporting hardware is built into the box when it is new.
2. One year later, as part of your preparation for your quarterly service upgrade, you review the hardware PSP buckets for any devices you may have in plan, to ensure all relevant PTFs are included in the service you are applying.
3. Two months after this, you order the new OSA and prepare for its installation. You go into HCD and define the adaptor and add it to the access or candidate list for all LPARs that will be able to use it. You also add it to the software configuration for all z/OS systems that will be using the device. You then use the new configuration to create a new IOCDS and do a software and hardware dynamic activate of that IOCDS.
4. Your IBM engineer then installs the OSA in the server.
5. You update the TCP configuration definitions and use the TCP OBEYFILE command to dynamically bring that new device into use by TCP.

11.2.3 IPL frequency

As discussed in 1.2, “Role of Parallel Sysplex exploitation” on page 2, the predominant reason most sites do not want to IPL too frequently is because of the impact the IPL has on application availability. *If* you were able to implement a configuration where all critical applications fully supported data sharing and dynamic workload balancing, you could take a system down without impacting the availability of those applications. This gives you the flexibility to choose your IPL frequency on issues other than application availability.

You might still decide that you only want to IPL once every six months, or you might prefer monthly IPLs—but that decision can be based on sound technical reasons, and not driven by availability requirements. This section provides reasons why you might want frequent IPLs, and reasons for infrequent ones. Keep these points in mind when making your decision.

Risks of IPLing too infrequently

One of the greatest risks of very infrequent IPLs is that the number of changes per IPL is increased. For example, if you make 60 changes a year that require an IPL, and you IPL every two months, that is 10 changes per IPL. But if you only IPL twice a year, that is 30 changes per IPL. And the more changes there are per IPL, the higher is the likelihood that one of them will encounter an error.

Another problem with having many changes going into effect in a limited window is human nature. People are often more liable to make mistakes when operating under pressure. At best, this will elongate the outage. At worst, it may result in all changes being backed out—and a rare outage window being wasted.

You can decrease the number of changes being implemented at the IPL by fully exploiting the dynamic change capabilities discussed in this paper. However, if you make many dynamic changes between IPLs, you need to have “bullet-proof” Systems Management processes; this ensures that every one of those dynamic changes gets reflected back into the relevant Parmlib member so that it is not regressed at the next IPL. Also, if something does accidentally get regressed, and it has been six months since the last IPL, it can be difficult (and time-consuming) to try to identify exactly what change was lost.

You also need to consider the quality impact of having very long intervals between IPL opportunities. For example, suppose you have a new product that requires an IPL, and there are only two IPLs per year. This means that the product will either have to go in at an upcoming IPL, or else have to wait for another six months. However, suppose that there are problems in testing which mean that the product will not be fully tested before the IPL. In such a case, do you go ahead and install a product that is not properly tested, or do you wait for another six months? If there were more IPL windows you could hold off, complete your testing, and take the product live in two months time at the next IPL.

There is also a business impact by having very long intervals between IPLs because it can greatly delay the uptake of new technology. We worked with one customer that only had one IPL a year (because of very high availability requirements), resulting in it taking years rather than months to implement data sharing (which, ironically, would have addressed their availability requirements).

Another risk of running for too long between IPLs is that it increases the time between applying preventive service to your system. To reduce the chances of encountering a problem that has already been fixed, IBM recommends that you review HIPER PTFs every week, and apply any HIPER service at your next maintenance slot. The frequency of that slot depends on your environment and your experiences. If you are experiencing a lot of change or find that you are encountering problems that already have a fix available, you may decide to apply HIPER service more frequently (maybe every month or every quarter). On the other hand, if your environment is stable and you are not encountering bugs, you may decide to apply maintenance less frequently, such as half-yearly.

One of the more common reasons for IPLing more frequently is to address storage leaks or storage fragmentation. While you can address these to an extent by adjusting the amount of common storage you define, there is a greater risk that, as storage becomes more and more fragmented over time, you will be impacted if you run for extremely long times between IPLs.

An additional consideration is that if the IPLs are very far apart, the operators may not have much experience with them, and this could result in errors.

Disadvantages of IPLing too frequently

The other side of the discussion about operator skills is that if the operators are unskilled, having them perform frequent IPLs increases the number of opportunities for human error interfering with the IPL. One could argue that IPLing more frequently should improve operator skills, so this should actually decrease the likelihood of mistakes. However, while we are completely in favor of operator education, IPLing a production system should not be a way of getting that education; that is what test systems are for.

There are some system functions that only get carried out at IPL time (for example, reading LPALST data sets). The more frequently you IPL, the more likely you are to encounter a problem that might exist in one of these functions.

Another challenge is that, in the window while a system is being shut down and restarted, the capacity normally associated with that system may be unavailable to the sysplex. This is most likely to be the case if you only have one system per sysplex on each server.

If you have more than one system from a sysplex on a server, set up the LPARs so that the resources normally associated with each LPAR can be made available to the other LPAR during the outage. One way to do this is to define the LPARs with shared CPs and also with Reserved CPs. While one LPAR is being stopped, you can configure on some of the Reserved CPs to the remaining LPAR, and then configure them off again after the system is fully reinitialized.

A

Items still requiring an IPL

This appendix contains a list of items that cannot be modified dynamically and require an IPL to implement.

Items requiring an IPL to implement

Despite all the improvements that have been made to the operating system and its components over the last few years, there are still some changes that must be implemented via an IPL. Table A-1 contains a list of the most common such changes.

Table A-1 List of items still requiring an IPL

Item	Description
JES3 - add spool volume	Requires a warm start of JES3 and a sysplex IPL to add a new spool volume.
RACF - new ICHRDSNT and ICHRRNG	Changes to the RACF Data set name table (ICHRDSNT) and RACF range table require a sysplex IPL.
CONSOLxx in SYS1.PARMLIB	You cannot add a new hardware console without an IPL.
Backout of GRS Star	Although you can move from GRS Ring to GRS Star dynamically, a sysplex IPL is needed to go back to Ring mode, should it be necessary.
MAXUSER, RSVSTRT and RSVNONR parameters	These IEASYSxx parameters can only be modified via an IPL.
IBM service	To apply maintenance to modules that reside in LPALIB usually requires an IPL. When it is necessary, the HOLDDATA instructions (++HOLDDATA) that accompany PTFs document this. Mention is made in this document to maintenance of subsystems such as the USS read-only root file system, JES2 and TCP/IP. Although it is technically possible to apply maintenance to these dynamically, a lot of sites find it “easier” to close the system down and restart. This is due to the complications involved with closing all applications using any of these subsystems. While many sites have automation in place to close a system down cleanly and then restart it, far fewer have similar automation to stop and start individual subsystems.
USS SWA() parameter	Changing the SWA() parameter in BPXPRMxx requires an IPL to implement.
USS SYSPLEX() parameter	Changing the SYSPLEX() parameter in BPXPRMxx requires an IPL to implement.
MODIFY OMVS,SHUTDOWN	There are instances where `F OMVS,SHUTDOWN` cannot shut down z/OS UNIX completely and an IPL is still required in order to correct the condition that instigated the shutdown.
LPA modules	It is usually necessary to re-IPL to replace LPA modules. For example, many service updates of LPA modules will require a re-IPL
BRLM implementation	Moving to the Distributed Byte Range Lock Manager (BRLM) can be done dynamically, but a sysplex-wide IPL is required to back out this change.
PDSE Sharing	Reverting to PDSESHARING(NORMAL) requires a sysplex IPL.

Item	Description
New master catalog	Changes to either the LOADxx member of Parmlib or the SYSCATLG member in SYS1.NUCLEUS require an IPL to pick up the change.
System Symbols	An IPL is required to implement or modify system symbols. IEASYMUP can be used for this purpose, but it is not fully supported and should be used with caution.
Parmlib members	The following Parmlib members can only be updated with an IPL to pick up changes: ► ALLOCxx ► BLSCECT ► BLSCUSER ► CNIDTRxx ► CONFIGxx ► DEVSUP ► EPHWP00 ► IEAAPP00 ► IEAFIX00 ► IEAPAKxx ► IOEPRMxx ► IEASYSxx ► LOADxx (For all parameters other than Parmlib concatenation) ► MSTJCLxx ► NUCLSTxx ► SCHEDxx (For all parameters other than PPT entries)
PLPA and COMMON PAGE data sets	Amendments to these page data sets require an IPL.
SYSPLEX enablement	Moving from XCFLOCAL mode to MONOPLEX or MULTISYS mode requires an IPL.

.

B

IEASYMUP

This appendix describes the IEASYMUP sample program provided in z/OS and subsequent releases of z/OS. This program can be used, within certain restrictions, to dynamically add system symbols and to change the values of existing symbols.

Creating the IEASYMUP program

IEASYMUP is delivered as an object deck in SYS1.SAMPLIB. No sample JCL is provided to install or link the deck into a load module. However, the JCL shown in Example B-1 can be used to define an SMP/E USERMOD for this purpose.

Example: B-1 Sample SMP/E USERMOD for IEASYMUP program

```
//* This UMOD installs a ++MOD with the LMOD IEASYMUP in SYS1.LINKLIB *//
//* It also uclin's the ++SAMP to replace rmid to highlight any future   *//
//* IBM maintenance                                                    *//
//*
//*........
//Users SMP/E JCL procedure goes here
//*........
//SMPPTFIN DD DATA,DLM=$$
++USERMOD(umod_name) REWORK(2005001).
++VER(Z038) FMID(HBB7709) .
++JCLIN .
//LKED EXEC PGM=IEWL,PARM='LIST,MAP,XREF,NCAL,LET'
//SYSLMOD DD DSN=LINKLIB,DISP=SHR
//SAMPLIB DD DSN=SAMPLIB
//SYSLIN DD *
INCLUDE SAMPLIB(IEASYMUP)
SETCODE AC(1)
ENTRY IEASYMUP
NAME IEASYMUP(R)
++MOD(IEASYMUP) DISTLIB(SAMPLIB) LKLIB(SAMPLIB).
$$
//SMPCNTL DD *
SET BDY(GLOBAL) .
RECEIVE S(umod_name) .
SET BDY(MVST100) .
APPLY CHECK S(umod_name) REDO RC(RECEIVE=0) .
APPLY S(umod_name) REDO RC(APPLY=0) .
UCLIN RC(APPLY=4) .
REP SAMP(IEASYMUP) RMID(umod_name) .
ENDUCL .
/*
```

Note that the library you link IEASYMUP into must be APF-authorized.

IEASYMUP restrictions

Before you start changing all your system symbols using IEASYMUP, it is vital to understand some restrictions related to changing system symbols dynamically.

- This program (IEASYMUP) is provided on an as-is basis, as are all samples in SYS1.SAMPLIB. There is no official support for the program, although we will address errors on a best-efforts basis.
- If you are not careful in how you use the program, you could end up with different tasks using different values for a symbol you have changed. For example, if you have a long-running REXX exec, it could obtain the value of a symbol when it starts, and then continue to use that value thereafter. So even though you have updated the value of the symbol, that REXX exec will continue to use the old value.

- When JES3 starts, each JES3 passes the symbols on that system, and the value of those symbols, to the other JES3s. This is because in JES3, the JCL for a started task could get interpreted on one system, but be executed on another. So JES3 needs to know the symbols on all systems so that the correct values can be substituted.

  If you update a symbol with this program, the updated value does not get propagated to the other systems. So, in a JES3 complex, JCL that gets interpreted on other systems would continue to contain the previous value of the symbol, rather than the current, correct value as updated by IEASYMUP.

Using IEASYMUP

In order to prevent unauthorized personnel from updating system symbols using IEASYMUP, the program uses a SAF call to ensure the caller is authorized. Therefore, before you can use the program you must set up a security profile in the FACILITY class called IEASYMUP.*. If you wish to allow access to a subset of system symbols, you can further set up profiles called IEASYMUP.symbolname. An example is as follows:

```
RDEFINE FACILITY IEASYMUP.* UACC(NONE)
RDEFINE FACILITY IEASYMUP.symbolname UACC(NONE)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH
```

These profiles will stop anyone from using IEASYMUP to update any system symbols. In order to be able to update `symbolname`, you need UPDATE access to the associated profile, as shown in this example:

```
PERMIT IEASYMUP.symbolname CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
```

If you attempt to update a symbol that is protected by a profile and you do not have update access to that profile, the job will end with return code 288 and the job log will contain:

```
ICH408I USER(KYNEF   ) GROUP(SYS1    ) NAME(FRANK KYNE
  IEASYMUP.TESTFK1 CL(FACILITY)
  INSUFFICIENT ACCESS AUTHORITY
  FROM IEASYMUP.TESTF* (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
ISYM003I Symbol table update not done. RC=00000120
```

If you forgot to define any IEASYMUP profile, the job will fail with the same return code.

The JCL to run IEASYMUP is extremely simple; you only need an EXEC card, and a parameter specifying the change you wish to make. A sample is shown in Example B-2.

Example: B-2 Sample JCL to run IEASYMUP

```
//KYNEFR  JOB (0,0),'UPDATE SYMBOL',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//SYMUPDTE EXEC PGM=IEASYMUP,PARM='TESTFK=TEST1'
//STEPLIB  DD  DSN=KYNEF.IEASYMUP.LOADLIB,DISP=SHR
```

The input parameter is specified using the PARM keyword on the EXEC card, in the format:

```
PARM='KEY1=VALUE1 KEY2=VALUE2 ...KEYN=VALUEN'
```

where:

- Symbols can be used in defining values if the program is invoked as a started task or started job.
- The name of each KeyN should not contain the leading ampersand (&) or the trailing dot (.).

- Each key=value must be separated from the preceding one by exactly one blank.
- There must be no trailing blanks at the end of the parameter string.
- Upper case translation is not performed by the processing, so be careful to put everything in upper case as needed.
- Must be in an APF-authorized library.
- Symbol values, including the length, can be changed.
- New symbols can be added.
- An ampersand (&) cannot be part of the name or substitution text.
- The specified symbol name must be no longer than 8 characters (as it does not contain the leading ampersand (&) or trailing dot (.).
- The specified substitution text must be no longer than the specified symbol name plus one for the ampersand (&).
- z/OS 1.4 raised the limit from 98 to 800 static symbol definitions. IBM guarantees the ability to define 800 installation-specified symbols. The limit is as many symbols as can fit into the symbol table. The symbol table can be up to 32512 bytes long, including a 4–byte header. Each entry in the table consists of:
  - 16–bytes, plus
  - The symbol name, including the preceding ampersand (&) and trailing dot (.) For example:

    ```
    &MYSYM.
    ```
  - The substitution text value. For example:

    ```
    MYVALUE
    ```

IEASYMUP error codes

Error return codes from IEASYMUP are of the form xxxxxxyy, where xxxxxx is the symbol number, and yy is the reason code.

The reason codes are as follows:

8	Missing parameter. No symbol name was specified.
C	Symbol name contains an ampersand (&). The symbol name must follow the rules described in *z/OS MVS Initialization and Tuning Reference,* SA22-7592.
10	“bad symbol name length”. Either IEASYMUP did not even find the equal sign (=) indicating the delimiter between name and value, or it found it after more than eight characters. The symbol name must be eight characters or less.
14	The symbol length was greater than 8 characters.
18	Too many symbols were specified in the PARM statement. Reduce the number of symbols and resubmit.
1C	A Reserved symbol name was specified. See *z/OS MVS Initialization and Tuning Reference,* SA22-7592 for a list of Reserved symbol names.
20	The user associated with the IEASYMUP job did not have access to the RACF profile protecting the symbol.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 138. Note that some of the documents referenced here may be available in softcopy only.

- *DFSMS Release 10 Technical Update,* SG24-6120
- *IBM System z9 109 Configuration Setup,* SG24-7203
- *IBM System z9 109 Technical Introduction,* SG24-6669
- *IBM System z Connectivity Handbook*, SG24-5444
- *MVS/ESA HCD and Dynamic I/O Reconfiguration Primer,* SG24-4037
- *Parallel Sysplex - Managing Software for Availability,* SG24-5451
- *Partitioned Data Set Extended Usage Guide,* SG24-6106-01
- *S/390 Parallel Sysplex: Resource Sharing,* SG24-5666
- *S/390 Time Management and IBM 9037 Sysplex Timer,* SG24-2070
- *Systems Programmers Guide to: z/OS System Logger,* SG24-6898
- *z/OS Distributed File Service zSeries File System Implementation,* SG24-6580
- *z/OS V1R3 and V1R5 DFSMS Technical Guide,* SG24-6979
- *z/OS Version 1 Release 5 Implementation,* SG24-6326
- *z/OS Version 1 Release 7 Implementation,* SG24-6755

Other publications

These publications are also relevant as further information sources:

- *IBM Health Checker for z/OS: User's Guide,* SA22-7994
- *z/OS DFSMS Managing Catalogs,* SC26-7409
- *z/OS MVS Data Areas, Vol 1 (ABEP-DALT),* GA22-7581
- *z/OS MVS Diagnosis: Tools and Service Aids,* GA22-7589
- *z/OS MVS Initialization and Tuning Guide,* SA22-7591
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS Planning: Operations,* SA22-7601
- *z/OS MVS System Commands,* SA22-7627
- *z/OS V1R6.0 MVS Extended Addressability Guide,* SA22-7614
- *z/OS V1R7.0 Migration,* GA22-7499
- *z/OS Communications Server: IP Configuration Reference,* SC31-8776

Online resources

These Web sites and URLs are also relevant as further information sources:

- White paper about consoles and availability:

  http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130166.pdf

- Sample CNTRIDxx member:

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

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM Redbooks

z/OS Planned Outage Avoidance Checklist

(0.2”spine)
0.17”<->0.473”
90<->249 pages

z/OS Planned Outage Avoidance Checklist

Making dynamic changes

Planning ahead to avoid planned outages

Recovery enhancements to avoid IPLs

One of the offshoots of the increasingly competitive global business environment is growing pressure to reduce the number of outages, both of a planned and unplanned nature. In the area of planned outages, IBM has been steadily reducing the number of changes that require a system restart to implement. However, because of the detailed nature of these changes, many of them are not widely publicized and therefore not widely known.
This IBM Redbook attempts to redress this situation by listing the changes that have been made to the operating system since OS/390 1.3.

SG24-7328-00 ISBN 0738494925