z/OS

MVS Initialization and Tuning Reference

Version 2 Release 1

SA23-1380-04

z/OS

MVS Initialization and Tuning Reference

Version 2 Release 1

SA23-1380-04

Note
Before using this information and the product it supports, read the information in "Notices" on page 803.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

This publication describes the members of SYS1.PARMLIB for z/OS® (5650-ZOS), and the processes related to initializing the system. For each member, the document describes the meaning and use of each parameter, syntax rules, syntax examples, value ranges that are syntactically acceptable, default values, and performance notes where applicable.

This publication is a companion to the *z/OS MVS Initialization and Tuning Guide*.

For information about how to install the software products that are necessary to run z/OS, see *z/OS Planning for Installation*.

Who should use this information

This publication is for anyone whose job includes installing hardware and software, and customizing the software. This document is intended to be a reference on how to code system parameters and make other changes required. Usually, these tasks are performed by a systems programmer.

The publication assumes the user can:

- Code JCL statements to execute programs or cataloged procedures.
- Code in assembler language and read assembler, loader, linkage editor, and binder output.

This publication is also for anyone who tunes the system. System tuning requires you to determine where the system needs adjustment, to understand the effects of changing the system parameters, and to determine the changes to the system parameters that will bring about the desired effect.

MVS workload management

Many of the performance functions described in this document can be defined using MVS™ workload management (WLM). For more information see: *z/OS MVS Planning: Workload Management* and *z/OS MVS Programming: Workload Management Services*.

Where to find more information

Where necessary, this document references information in other documents, using the shortened version of the document title. For complete titles and order numbers of the documents for all products that are part of OS/390, see *z/OS Information Roadmap*.

How to read syntax conventions

This topic describes how to read syntax conventions. It defines syntax notations and provides syntax examples that contain these items.

Table 1. Syntax conventions

Notation	Meaning	Syntax example	Sample entry
Apostrophes	Apostrophes indicate a parameter string and must be entered as shown.	`SEND 'message',NOW`	`SEND 'listings ready',NOW`
Comma	Commas must be entered as shown.	`DISPLAY C,K`	`DISPLAY C,K`
Ellipsis ...	Ellipsis indicates that the preceding item or group of items can be repeated one or more times. Do not enter the ellipsis.	`VARY (devspec[,devspec]...),ONLINE`	`VARY (282,283,287),ONLINE`
Parentheses and special characters	Parentheses and special characters must be entered as shown.	`DUMP COMM=(text)`	`DUMP COMM=(PAYROLL)`
Underline	Underline indicates a default option. If you select an underlined alternative, you do not have to specify it when you enter the command.	`K M[,AMRF={Y | N}] |,REF`	`K M`

Lowercase parameter	Lowercase indicates a variable term. Substitute your own value for the item.	`MOUNT devnum`	`MOUNT A30`
			or
			`mount a30`

Uppercase parameter	Uppercase indicates the item must be entered using the characters shown. Enter the item in either upper or lowercase.	`DISPLAY SMF`	`DISPLAY SMF`
			or
			`display smf`

Single brackets	Single brackets represent single or group-related items that are optional. Enter one or none of these items.	`DISPLAY SLIP[=xxxx]`	`DISPLAY SLIP=W292`
Stacked brackets	Stacked brackets represent group-related items that are optional. Enter one or none of these items.	`[TERMINAL]` `[NOTERMINAL]`	`NOTERMINAL`
Single braces	Single braces represent group-related items that are alternatives. You must enter one of the items. You cannot enter more than one.	`{COMCHECK | COMK}`	`COMK`

Table 1. Syntax conventions (continued)

Notation	Meaning	Syntax example	Sample entry

Stacked braces	Stacked braces represent group related items that are alternatives. You must enter one of the items. You cannot enter more than one.	`MN {DSNAME}`
		`{SPACE }`
		`{STATUS}`
			`MN SPACE`

Or-bar (|)	An or-bar indicates a mutually exclusive choice. When used with brackets, enter one or none of the items. When used with braces, you must enter one of the items.	`ACTIVATE RECOVER=SOURCE | TARGET`	`ACTIVATE RECOVER=SOURCE`

Stacked items with or-bars (|) and brackets	Stacked items with or-bars indicates a mutually-exclusive choice. Enter one or none of these items.	`CD RESET [ ,SDUMP ]`
		`|,SYSABEND`
		`|,SYSUDUMP`
		`|,SYSMDUMP`
		`|,ALL`
			`CD RESET,SYSUDUMP`

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 MVS Initialization and Tuning Reference
  SA23-1380-04
- The topic and page number that is related to your comment.
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS Version 2 Release 1 (V2R1) as updated February, 2015

The following changes are made for z/OS Version 2 Release 1 (V2R1) as updated February, 2015.

New

- The CORE statement was added to the CONFIG*xx* parmlib member. It specifies the configuration for a core. See "Statements and parameters for CONFIGxx" on page 225.
- The MULTINCRFLC parameter is added to the DEVSUP*xx* parmlib member. You can use it to disable Multiple Incremental FlashCopy (Incremental FlashCopy Version 2). Refer to Chapter 28, "DEVSUPxx (device support options)," on page 305.
- APAR OA43661, which exploits multi-target PPRC support, added a new keyword, PPRCMT, for the DEVSUP*xx* parmlib member. You can use it to enable or disable multi-target PPRC support. Refer to "IBM-supplied default for DEVSUPxx" on page 309 and "Statements/Parameters for DEVSUPxx" on page 310.
- With APAR OA46482, EASYTIERHINTS was added to the DIAGxx parmlib member. It enables the Easy-Tier Copy Temperature function for software-defined storage. See "Statements/parameters for DIAGxx" on page 326.
- With APAR OA45297, a new trap, IeaSlipConfirm, was added to the TRAPS NAME parameter of the DIAGxx parmlib member. See "Statements/parameters for DIAGxx" on page 326.
- An example to illustrate the new high CSA options was added to "Statements/parameters for IEADMCxx" on page 380.
- The MT_CP_MODE statement was added to the IEAOPT*xx* parmlib member. It specifies the multithreading (MT) mode. See Chapter 49, "IEAOPTxx (OPT parameters)," on page 397 and "Statements/parameters for IEAOPTxx" on page 398.
- The HYPERWRITE statement was added in IECIOS*xx*. See "HYPERWRITE" on page 510.
- The CORE statement was added to the LOAD*xx* parmlib member. It specifies that z/OS should configure a processor view of core. See "Statements/ parameters for LOADxx" on page 625.

Changed

- The text length of system symbol &SYSOSLVL has been corrected. The length is 8. Table 4 on page 35.
- In IEAOPT*xx*, the **MCCFXTPR** keyword has a new default value for systems with a large amount of real storage, and the **RCCFXET** and **RCCFXTT** keywords now accept a value of `AUTO`. See "Statements/parameters for IEAOPTxx" on page 398.

- The ZAAPZIIP parameter in the IEASYSxx parmlib member was updated. See "ZAAPZIIP" on page 493.
- The MACHMIG statement of the LOADxx parmlib member was extended to allow the specification of VEF, which indicates that the vector extension facility is not to be exploited, even if it is available. See "Statements/parameters for LOADxx" on page 625.

Deleted

No content was removed from this information.

Summary of changes for z/OS Version 2 Release 1 (V2R1) as updated September, 2014

The following changes were made for z/OS Version 2 Release 1 (V2R1) as updated September, 2014.

New

- For the DEVSUP*xx* parmlib member, a restriction was added to the TAPEPBLKSZLIM= parameter to indicate that DFSMSdss only supports BLKSZLIM of 65,520 and higher. See "Statements/Parameters for DEVSUPxx" on page 310.
- With APAR OA44049, diagnostic CTRACE data was added to VSAM RLS BMF GET/FREE requests. With the new parameter CSRPOOLDIAG in the DIAG*xx* parmlib member, the get and free services of the CSR cellpool return diagnostic data. See "Syntax format of DIAGxx" on page 325 and "Statements/parameters for DIAGxx" on page 326.
- With APAR OA41968, a new keyword, INCLUDE1MAFC, was added to the LFAREA parameter in the IEASYSxx parmlib member. It specifies that the 1 MB pages are to be included in the available frame count (RCEAFC), and improves paging. See "LFAREA" on page 455.

Changed

- The BPXPRM*xx* parmlib member was updated to clarify that the value of the SHRLIBRGNSIZE parameter must be evenly divisible by 1048576. See "Statements and parameters for BPXPRMxx" on page 143.
- Clarification was added to the TIMEZONE parameter of the CLOCK parmlib member about applications that are required to run with local time equal to UTC. See "Statements/parameters for CLOCKxx" on page 196.
- An explanation was added to the CONFIGxx parmlib section that the depreciated ESTOR parameter is tolerated. See "Statements and parameters for CONFIGxx" on page 225.
- With APAR OA44526, the lower limit for the IEAOPTxx keyword BLWLINTHD was changed. See "Statements/parameters for IEAOPTxx" on page 398.
- The documentation for the BreakPointValue parameter for the IGDSMS*xx* parmlib member was clarified and made more complete. See "Optional keywords for IGDSMSxx" on page 542.
- The IBM-supplied program properties table (PPT) had several updates.
  - With APAR OA42165, the PPT table was updated to include CRITICALPAGING for the BCPii address space program, HWIAMIN2.

– With APAR OA43256, additional support of zEDC Express update and fixes for z/OS PCIE support were included. IQPINIT for PCIE services was added to the PPT table.
– With APAR OA44695, support was added for IOSVROUT and programs attached to it to obtain 64-bit virtual storage. The "No honor IEFUSI region settings" value was added to IOSVROUT in the PPT table.

See Table 34 on page 737 in the SCHED*xx* parmlib section.

Deleted

No content was removed from this information.

Summary of changes for z/OS Version 2 Release 1 (V2R1) as updated March, 2014

The following changes were made for z/OS Version 2 Release 1 (V2R1) as updated March, 2014.

New

- The ZEDC_R ("zEDC Required") and ZEDC_P ("zEDC Preferred") options were added to the COMPRESS parameter in the IGDSMSxx member of SYS1.PARMLIB. For more information on the zEDC compression enhancements, see *z/OS DFSMS Using the New Functions*.

Changed

- The default substitution text for &SYSOSLVL was clarified. See Table 4 on page 35.
- The BPXPRMxx member had various changes. See "Statements and parameters for BPXPRMxx" on page 143. Some of the updates included the following ones:
  – The BPXPRMxx parameters were alphabetized.
  – The FSFULL option is supported by TFS, in addition to HFS and TFS.
  – A recommendation for LOSTMSG was added.
  – For the FILESYSTYPE statement, clarifications to the TFS configuration parameters were added to the PARM parameter.
  – For the MOUNT statement, a restriction was added to the DDNAME parameter. Clarifications to the TFS configuration parameters were also added to the PARM parameter.
- For ALLOCxx, two new keywords were added to the SYSTEM parameter:
  – BATCH_RCLMIGDS(SERIAL | PARALLEL)
  – OPTCDB_SPLIT(EXPLICIT | CATALOG)
- A minor clarification was made to the IHALCCA and IHAPCCA parameters in DIAGxx. See "Statements/parameters for DIAGxx" on page 326.
- A note was added clarifying that WARNUND only applies to primary system parameters that are listed in IEASYSxx, and not their associated sub-parameters. See Chapter 54, "IEASYSxx (system parameter list)," on page 431.

Summary of changes for z/OS Version 2 Release 1 (V2R1) as updated December, 2013

The following changes were made for z/OS Version 2 Release 1 (V2R1) as updated December, 2014.

New

- The MAXGENS_LIMIT parameter of PARMLIB member IGDSMSxx specifies an upper limit for the MAXGENS parameter on the JCL DD statement. MAXGENS specifies the number of generations for members in version 2 PDSEs. For more information, refer to Chapter 59, "IGDSMSxx (storage management subsystem definition)," on page 537.
- New information about comments was added to IGDSMSxx. See "Syntax rules for IGDSMSxx" on page 539.

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Part 1. Overview

Chapter 1. System tailoring

System tailoring is the overall process by which an installation selects its operating system. The process consists of the specification of system options through these mechanisms:

- MVS hardware configuration definition (HCD), which is described in "MVS hardware configuration definition."
- Initialization-time selections which appear in the topic "System tailoring at initialization time" on page 4.
- Implicit system parameters which appear in the topic "Implicit system parameters" on page 27.
- After IPL, system tailoring through operator commands. One example of a command that will perform a system tailoring function is the SETPROG command (for example, SETPROG LPA,ADD). You get the same result whether you activate a function by issuing a SETPROG command or by activating the PROGxx parmlib member.

  For more information about how to use commands to start, load, initialize, and control your system, see *z/OS MVS System Commands*.

Note: Many system options have defaults. The IBM® defaults for some of these options might change in a future release. If you want to continue using the current default, set the default value that you want.

An installation can identify one or more active instances of the operating system. For example, the installation might choose to identify a specific instance to be used only during off-shift hours.

To identify an instance of the operating system, the installation assigns a unique 16-character EBCDIC identifier by using the AMASPZAP program immediately after system initialization, or by creating an alternate nucleus. For more information about AMASPZAP, see *z/OS MVS Diagnosis: Tools and Service Aids*. For more information about creating an alternate nucleus, see "Specifying an alternate nucleus" on page 9.

MVS hardware configuration definition

The hardware configuration definition (HCD) allows you to use the HCD dialog to perform the following tasks for hardware configuration:

- Define operating system configuration
- Define the channel subsystem (CSS) configuration
- Define ESCON director (ESCD) and switch configurations
- Activate configuration data
- Maintain I/O definition files
- Query and print configuration data
- Migrate existing configuration data

For information about using the HCD dialogue, see *z/OS HCD User's Guide*.

System tailoring at initialization time

Initialization-time choices that help to tailor the system can come from several sources:

- Various types of IPLs.
- Operator action, described in "Operator entry of parameters" on page 5.
- SYS1.PARMLIB and additional parmlib data sets are described in "Description and use of the parmlib concatenation" on page 6. This data set is one of the main sources of IPL-time parameters.
- Specifying an alternate nucleus or master catalog, described in "Specifying an alternate nucleus" on page 9 and "Specifying an alternate master (system) catalog" on page 10.
- Specifying different master scheduler JCL, described in "Understanding the master scheduler job control language" on page 10, which includes information about when you might need different master scheduler JCL and how to establish the JCL you need.
- The JES2 or JES3 initialization data set. (See either *z/OS JES2 Initialization and Tuning Guide* or *z/OS JES3 Initialization and Tuning Guide*.)

Types of IPL

There are several types of IPL:

Cold Start
: Any IPL that loads (or reloads) the PLPA, but does not preserve VIO data set pages. The first IPL after system installation is always a cold start because the PLPA is initially loaded. Subsequent IPLs are cold starts when the PLPA is reloaded either to alter its contents or to restore its contents if they were destroyed.

Quick Start
: Any IPL that does not reload the PLPA and does not preserve VIO data set pages. (The system resets the page and segment tables to match the last-created PLPA.)

Warm Start
: Any IPL that does not reload the PLPA, but does preserve journaled VIO data set pages.

The first IPL after system installation

At the first IPL after system installation, the system automatically loads the PLPA from the LPALST concatenation. The page data sets for this IPL are those specified in the IEASYS00 parmlib member, plus any specified by the operator.

After the first IPL, you must run IFCDIP00 to initialize the LOGREC data set. This routine must also be run whenever the LOGREC data set is reallocated.

An IPL at which the PLPA is reloaded

The PLPA must be reloaded: (1) at the first IPL after system initialization, when the system loads it automatically, (2) at an IPL after the installation has added or modified one or more modules in the LPALST concatenation, has tested the alteration, and now wants to put the replacement module(s) in the PLPA, and (3) at an IPL after the PLPA page data set has been damaged (and is therefore unusable) and its contents must be restored. The PLPA can also be reloaded for other reasons (such as when the addition of more storage causes the nucleus and the PLPA to overlap). Reloading the PLPA should be discretionary; that is, it

should not be a common occurrence. It should be done only when necessary because the associated I/O slows down the IPL and because previously existing VIO data set pages are not preserved.

To reload the PLPA from the LPALST concatenation, the operator enters CLPA (create link pack area) as one of the responses to the SPECIFY SYSTEM PARAMETERS prompt. For more information about loading the PLPA, see the CLPA parameter in Chapter 54, “IEASYSxx (system parameter list),” on page 431.

An IPL after power-up

The IPL performed after power-up is called a “quick start”, because the PLPA from the previous IPL can be used without reloading from the LPALST concatenation. For a “quick start”, the CVIO system parameter is used; VIO data set pages are purged, page data sets added (optionally reserved for non-VIO paging). The operator or the IEASYSxx parmlib member can add additional page data sets by specifying the PAGE parameter (with or without the NONVIO system parameter). For information about the CVIO, PAGE, and NONVIO parameters, see Chapter 54, “IEASYSxx (system parameter list),” on page 431.

An IPL after a system crash

If the operator does not enter the CLPA or CVIO system parameters, the operator can "warm start" the system after a system crash. Existing journaled VIO data set pages and PLPA pages are retained. The specified parmlib parameter list (IEASYSxx) cannot include the CVIO or CLPA system parameters. (The specification of one or more IEASYSxx members by the operator at IPL time is described in the next topic, “Operator entry of parameters.”)

Any definitions of existing page data sets as non-VIO local page data sets are preserved. Also the operator can define a local page data set that previously was used for VIO paging as a non-VIO local page data set. During system operation, the VIO pages on the newly designated non-VIO local page data set will migrate to any local page data set used for VIO paging. An installation can remove a local page data set that was designated on the previous IPL as a non-VIO local paging data set. Removing a local page data set before a "warm start" requires that the local page data set contain no VIO pages. If the local page data set contains VIO pages, the system changes the "warm start" into a "quick start". A local page data set specified as NONVIO can contain VIO pages if one of the following conditions exists:

- Directed VIO was turned off while the previous system was active.
- A warning message to the operator indicated that VIO pages spilled to a non-VIO data set because no more space was available on those page data sets that were being used for VIO.

For more information about designating non-VIO page data sets, see the NONVIO system parameter in Chapter 54, “IEASYSxx (system parameter list),” on page 431.

Operator entry of parameters

If an IEASYSxx identifier is not specified in the LOADxx parmlib member, the operator is prompted to respond to the SPECIFY SYSTEM PARAMETERS message to direct the system components to the desired parmlib members.

Note: Use the LOAD parameter on the system console to allow the operator to force the prompt for system parameters. For more information, see *z/OS MVS System Commands*.

The operator can select the default general parameter list IEASYS00, or enter SYSP=(aa,bb...) to select one or more alternate general parameter lists, such as IEASYS01, IEASYS02, and so forth. If the operator responds to the SPECIFY SYSTEM PARAMETERS message without specifying the SYSP parameter, the system will use IEASYS00.

The system always processes the IEASYS00 member first, regardless of where you specify IEASYSxx suffixes. If the same parameter appears in both IEASYS00 and a specified alternate IEASYSxx list, the value in the alternate list overrides the value in IEASYS00. Also, a parameter value in a later specified IEASYSxx list overrides the same parameter in an earlier specified list. Table 8 on page 43 shows how the system overrides suffixes. See the description of the IEASYSxx member for more information.

The operator need not enter parameter values directly, except for those cases in which parameters are missing, are syntactically invalid, cannot be read, or must be supplemented to satisfy a special case. (An example of a special case is the operator entry of the PAGE parameter to increase the amount of paging space.)

If an error occurs with certain parmlib members, the operator is prompted to manually enter one or more of the member's parameters. If the parameter cannot be corrected, the operator can accept the system defaults. Most parameters have defaults, either as default parmlib members, or as coded values in system components. If a default does not exist (and if a parameter is not required), the operator can cancel the parameter. (The defaults are listed in the individual descriptions of parmlib members.)

If the parameter list supplied by the operator is longer than one line (there are 80 characters per line), the operator can follow the last parameter with a comma or a blank and CONT. For details about how to continue system parameters, see the description of the REPLY command in *z/OS MVS System Commands*.

An operator-entered parameter overrides the same parameter specified in parmlib member IEASYS00 or IEASYSxx, except for:

- A parameter for which operator intervention is prohibited (OPI=NO). In this case, the system ignores the parameter that the operator entered(unless the parmlib parameter was syntactically invalid and is being corrected from the console).
- The PAGE parameter. The page data set names that the operator enters are added for the duration of the IPL to those specified in either IEASYS00 or IEASYSxx. (For information about the PAGE parameter, see Chapter 54, "IEASYSxx (system parameter list)," on page 431.)

Note: To determine the LOAD parameter that was used for the current IPL, check ECVTMLPR field in the ECVT data area. The ECVT is mapped by the IHAECVT mapping macro. For a description of the ECVT, you can see *z/OS MVS Data Areas* in the z/OS Internet library (http://www.ibm.com/systems/z/os/zos/bkserv/).

Description and use of the parmlib concatenation

This topic discusses the parmlib concatenation, its purpose, ways to control the parmlib data set(s), and general syntax rules for creating most members of the data set(s). Table 2 on page 16 contains an overview of all the parmlib members.

The parmlib concatenation is a set of up to 16 partitioned data sets defined through PARMLIB statements in the LOADxx member of either SYSn.IPLPARM or

SYS1.PARMLIB which contains many initialization parameters in a pre-specified form in a single logical data set, thus minimizing the need for the operator to enter parameters. SYS1.PARMLIB makes the 17th or last data set in the concatenation and is the default parmlib concatenation if no PARMLIB statements exist in LOADxx. For specific information about how to define a logical parmlib concatenation, see Chapter 68, "LOADxx (system configuration data sets)," on page 619. The SYS1.PARMLIB data set itself can be blocked and can have multiple extents, but it must reside on a single volume. The parmlib concatenation used at IPL must be a PDS. However, after IPL you may issue a SETLOAD command to switch to a different parmlib concatenation which contains PDSEs. For information about processing of concatenated data sets see *z/OS DFSMS Using Data Sets*.

Parmlib contains both a basic or default general parameter list IEASYS00 and possible alternate general parameter lists, called IEASYSaa, IEASYSbb, and so forth. Parmlib also contains specialized members, such as COMMNDxx, and IEALPAxx. Any general parameter list can contain both parameter values and "directors". The directors (such as MLPA=01) point or direct the system to one or more specialized members, such as IEALPA01.

The parmlib concatenation is read by the system at IPL, and later by other components such as the system resource manager (SRM), the TIOC, and SMF, which are invoked by operator commands. The TIOC is the terminal I/O coordinator, whose parameters are described under member IKJPRM00. SMF is the System Management Facility whose parameters are described under member SMFPRMxx.

The system always reads member IEASYS00, the default parameter list. Your installation can override or augment the contents of IEASYS00 with one or more alternate general parameter lists. You can further supplement or partially override IEASYS00 with parameters in other IEASYSxx members or operator-entered parameters. You can specify the IEASYSxx members that the system is to use in:

- The IEASYMxx parmlib member
- The LOADxx parmlib member.

The operator selects the IEASYSxx member using the SYSP parameter at IPL. The parameter values in IEASYS00 remain in effect for the life of an IPL unless they are overridden by parameters specified in alternate IEASYSxx members or by the operator. See "Step 2. Determine where to specify system parameters" on page 42 for details about how the system processes system parameter specifications.

How to control the parmlib

With a parmlib concatenation, you have more flexibility in managing parmlib members and changes to parmlib members. To control parmlib and ensure that it is manageable, you need to consider the following:

- The RACF® (or other security product) read access is a must for applications that need to read SYS1.PARMLIB.
- Use the ability to have up to 16 installation-defined parmlib data sets separate your parmlib members along organization or function lines and use appropriate RACF security for each data set.
- Include members with installation changes in one of the 16 installation-defined parmlib data sets to avoid having the member overlaid by IBM maintenance on SYS1.PARMLIB.

  Note: If a member exists more than once within the parmlib concatenation, the first occurrence is used.

- Use an installation-defined parmlib data set to contain any parmlib members to be used on test systems. They can be included in front of your "standard" parmlib concatenation without forcing changes to the "standard" parmlib concatenation.
- Delete unsupported parameters and members. Because most components treat unsupported parameters from previous releases as syntax errors, you probably need to remove the old parameters or build parmlib from scratch. This action minimizes the need for operator responses during an IPL. Then, you can save space by removing unsupported members.
- Use the parmlib members for the appropriate functions. For example, use COMMNDxx to contain commands useful at system initialization. Use IEACMDxx for IBM*-supplied commands. Use IEASLPxx for SLIP commands. See each member for further information.
- Update the parmlib with new and replacement members, as you gain familiarity with the new release.
- Keep track of which parameters are included in particular parmlib members. This bookkeeping is necessary for two reasons: 1) The system does not keep track of parmlib members and their parameters and 2) The default general parameter list IEASYS00 is always read by the system and master scheduler initialization. The parameters in IEASYS00 can be overridden by the same parameters when they are specified in alternate general lists, such as IEASYS01, or IEASYS02. Then, certain parameters, such as FIX, APF, and MLPA, direct the system to particular specialized members (in this example, IEAFIXxx, and IEALPAxx). The installation should keep records of which parameters and which values are in particular members, and which general members point to which particular specialized members (COMMNDxx, IEALPAxx, and so forth). A grid or matrix for such bookkeeping is very helpful.
- Allocate sufficient space for parmlib. One way to estimate space is to count the number of 80-character records in all members which are to be included in one parmlib data set and factor in the block size of the data set. Then add a suitable growth factor (e.g., 100-300%) to allow for future growth of alternate members. Consult Table 2 on page 16 to determine which members can have multiple alternates. To recapture space occupied by deleted members, use the "compress" function of IEBCOPY. However, should the data set run out of space, you can copy the members to a larger data set, create a new LOADxx member in which you replace the PARMLIB statement for the full data set with a parmlib statement for the new larger data set, and then issue a SETLOAD command to switch to the concatenation with the new data set.
- Ensure EXITxx resides in SYS1.PARMLIB, because you can only access it from SYS1.PARMLIB.
- GTFPARM must also reside in SYS1.PARMLIB unless an alternate data set is used, as discussed in Chapter 36, “GTFPARM (generalized trace facility parameters),” on page 357.
- Decide which volumes and devices should hold the parmlib concatenation. You must catalog the data set, unless it resides in SYSRES or its volume serial number is included in the parmlib statement in LOADxx. You can place the data set on a slow or moderate speed device.
- Use a security product (like RACF*) to protect the data sets. The purpose is to preserve system security and integrity by protecting various members that define authorized programs and libraries, including but not limited to the appendage member (IEAAPPxx), the authorized program facility members (IEAAPFxx and PROGxx), the LPA libraries (LPALSTxx, IEALPAxx, PROGxx), and the link libraries (LNKLSTxx). In addition, ensure that you protect any

libraries that are named in those members to ensure that only appropriate users with system maintenance responsibility can update them.

General syntax rules for the creation of members

The following general syntax rules apply to the creation of most parmlib members. You can find the description of the exceptions to these rules under specific members. The general rules are:

- Logical record size is 80 bytes.
- Block size must be a multiple of 80.
- Any columns between 1 and 71 can contain data.
- Statements are entered in uppercase characters.
- Suffix member identifiers can be any combination of A-Z and 0-9, though some member identifiers may allow other characters.
- Columns 72 through 80 are ignored.
- For some parmlib members, continuation is indicated by a comma followed by one or more blanks after the last entry on a record.
- Leading blanks are suppressed. A record therefore need not start at a particular column.
- Suffix member identifiers (such as LNK=A2) can be any alphanumeric combination.
- Comments for some parmlib members are indicated by using /* and */ as the delimiters in columns 1-71, for example:

  ```
  /*comment*/
  ```

  /* and */ characters within a single-quoted string are usually not treated as comment delimiters.

  For nested comments, delimiters must be balanced. For example, you can nest comments as follows:

  ```
  /*comment1/*comment2*/*/
  ```

  Some parmlib members require other methods. Check specific parmlib members for information about specifying comments.

Sharing parmlib members

You can set up parmlib so two or more MVS systems can share parmlib members — even if those systems require unique values in those members. When coding parmlib members, you can use *system symbols* to temporarily replace unique values. Each system defines its own unique values for the system symbols. As with variables in program, each system replaces the system symbols with the defined values when the system symbols are processed. With ability to share parmlib members that require unique values allows, you can view the sysplex as a *single system image* with as little as one SYS1.PARMLIB data set.

For complete details about how to set up parmlib so two or more systems can share it, see Chapter 2, “Sharing parmlib definitions,” on page 33.

Specifying an alternate nucleus

Another less common way to change the system at an IPL is to cause the IPL program to read a member of a nucleus data set that is different from IEANUC01, the default nucleus member. One reason for such a nucleus switch may be the need to apply a PTF to the nucleus. You can IPL a secondary (alternate) nucleus by either of the following methods:

- Editing the alternate nucleus character of the LOAD parameter string before selecting the "initialize SCP" function. Use the system control (SYSCTL) frame of the system console. Modify the last character of the eight-character parameter string to specify the suffix for IEANUC0x. The IPL program retrieves this character from the system console frame and concatenates it as a suffix to IEANUC0 to form the alternate SYS1.NUCLEUS member name.
- Using the NUCLEUS statement of the LOADxx member to specify the desired alternate IEANUC0x member. See Chapter 68, "LOADxx (system configuration data sets)," on page 619 for more information.

Note: When you specify an alternate nucleus, the proper architectural extension of the nucleus must also exist. See ARCHLVL for more information about architectural extensions to the nucleus.

Specifying an alternate master (system) catalog

Another way to change the system at an IPL is to tell the system to select an alternate master catalog. Use one of the following methods:

- Identify the data set that contains the alternate master catalog on the SYSCAT statement in the LOADxx parmlib member.
- Tell the system to read a member of SYS1.NUCLEUS that is different from SYSCATLG, the default member. Respond to system message IEA347A with a two-character suffix. The two characters are appended to SYSCAT to form the member name that the system is to read.

The system prompts the operator for a SYSCAT suffix only if you:

- Do not specify the SYSCAT statement in LOADxx
- Specify, through the SYSCTL frame, an initialization message suppression indicator (IMSI) on the LOAD parameter that tells the system to prompt for the master catalog response.

Understanding the master scheduler job control language

The master scheduler JCL data set (commonly called master JCL) controls system initialization and processing. It contains data definition (DD) statements for all system input and output data sets that are needed to communicate between the operating system and the primary job entry subsystem, which can be JES2 or JES3. An IPL is required to add, remove or change any DD statements in the master JCL.

To change system initialization or processing, you can change the information in the member or use an alternate data set. You might need to change the master JCL at an IPL. For example, if you plan to use jobs as the source JCL for started tasks, IBM suggests that you modify the master JCL. See "Setting up started tasks with the Master JCL" on page 12 for more information.

You can also modify the master scheduler JCL data set to include START commands for other subsystems, along with DD statements necessary to communicate with them. You can also delete DD statements that do not apply to your installation's interactive configuration.

Where does the master JCL reside?

This topic describes:

- Where the master JCL resides
- How to change the master JCL

- How to set up started tasks with the master JCL
- How to write your own master JCL.

IBM supplies default master JCL in the MSTJCL00 load module in SYS1.LINKLIB. Figure 1 shows MSTJCL00 as it exists before it is assembled and link-edited into SYS1.LINKLIB:

```
MSTJCL00 CSECT
         DC    CL80'//MSTJCL00 JOB MSGLEVEL=(1,1),TIME=1440'
         DC    CL80'//         EXEC PGM=IEEMB860'
         DC    CL80'//STCINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//TSOINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR'
         DC    CL80'//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR'
         DC    CL80'/*'
         END
```

Figure 1. MSTJCL00 Load Module in SYS1.LINKLIB

MSTJCL00 contains:

- DD statements needed to define the internal reader data sets for started task control and TSO/E logons
- SYS1.UADS, a system data set used in TSO/E logons and terminal communications.

MSTJCL00 does not contain the START command that starts the primary job entry subsystem during master scheduler initialization. You can define and start the primary job entry subsystem using the PRIMARY parameter in the IEFSSNxx parmlib member.

Your installation can either use the default master JCL (shown in Figure 1) or specify an alternate version of the master JCL, as described in "Changing the master scheduler JCL."

Changing the master scheduler JCL

When making changes to the master JCL, keep the following in mind:

- If you add DD statements to the master JCL, create the associated data sets before you load the initial program that is to use them. If the system cannot allocate a data set that is defined in the master JCL, system initialization fails.
- If you specify the START command in the master JCL for the primary job entry subsystem, specify the NOSTART parameter for the subsystem in the IEFSSNxx parmlib member. For the syntax of the NOSTART parameter, see Chapter 56, "IEFSSNxx (subsystem definitions) - keyword parameter form," on page 519 in this information.
- No work can be done that requires JES input or output services until the primary job entry subsystem is started.

To make changes to the master JCL, you can specify an alternate version of the master JCL in one of the following:

- A MSTJCLxx parmlib member (recommended)
- A MSTJCLxx load module in SYS1.LINKLIB.

If you plan to specify an alternate version of the master JCL in the MSTJCLxx load module in SYS1.LINKLIB, remember that you need to assemble the alternate

version and link-edit it into SYS1.LINKLIB each time you make a change. It is easier to change the master JCL when it is specified in the MSTJCLxx parmlib member, because the assemble and link-edit are not required.

Note: IBM supplies the IEESMJCL member of SYS1.SAMPLIB as an example of alternate master JCL. The CSECT name in IEESMJCL is MSTJCL05. When specifying an alternate version of the master JCL, IBM suggests that you modify IEESMJCL according to your needs. The statements in IEESMJCL are only examples. They are not necessarily the same values that IBM supplies in the default load module MSTJCL00.

Coding the parameter in IEASYSxx

The MSTRJCL parameter in the IEASYSxx parmlib member specifies the data set that is to contain the master JCL:

- If the system finds a MSTJCLxx parmlib member with a suffix that matches the value specified or defaulted on the MSTRJCL parameter, it uses the master JCL in the MSTJCLxx parmlib member and does not process the MSTJCLxx load module in SYS1.LINKLIB.
- If the system *does not* find a MSTJCLxx parmlib member that matches the value specified or defaulted on MSTRJCL, it uses the master JCL in the corresponding MSTJCLxx load module in SYS1.LINKLIB.

Specifying the master JCL in parmlib

Take the following steps to specify the master JCL in a MSTJCLxx parmlib member:

1. Code the master JCL in the MSTJCLxx member, as described in Chapter 73, "MSTJCLxx (master scheduler JCL)," on page 687.
2. Code the MSTRJCL=xx system parameter in the IEASYSxx parmlib member or in response to the SPECIFY SYSTEM PARAMETERS prompt. (See Chapter 54, "IEASYSxx (system parameter list)," on page 431 for the syntax of the MSTRJCL=xx system parameter.)

Specifying the master JCL in the MSTJCLxx load module

Take the following steps to specify an alternate version of the master JCL in a MSTJCLxx load module in SYS1.LINKLIB:

1. Code the master JCL in the alternate data set. Assemble the data set. Then link-edit the data set as a module in SYS1.LINKLIB (or a library concatenated to SYS1.LINKLIB, using a LNKLSTxx or PROGxx member of of the parmlib concatenation).
2. Code the MSTRJCL=xx system parameter in the IEASYSxx parmlib member or in response to the SPECIFY SYSTEM PARAMETERS prompt. (See Chapter 54, "IEASYSxx (system parameter list)," on page 431 for the syntax of the MSTRJCL=xx system parameter.)
3. Ensure that SYS1.PARMLIB does not contain a MSTJCLxx member that matches the suffix specified on the MSTRJCL=xx system parameter. (If such a member exists, the system uses the master JCL in that member, and does not process the master JCL in the MSTJCLxx load module.)

Setting up started tasks with the Master JCL

The source JCL for a started task can be a job (source JCL that begins with a JOB statement) or a cataloged procedure. See *z/OS MVS JCL Reference* for information that describes the advantages of using one form of source JCL over another (using a job rather than a procedure).

If the source JCL for a started task is a job, the member containing the JCL must be part of a data set in the IEFPDSI DD or the IEFJOBS DD concatenation of MSTJCLxx. (If the member is not part of a data set in the IEFPDSI or IEFJOBS concatenation of MSTJCLxx, the procedures that act as source JCL for other started tasks will not be found.) IBM suggests that you define a new data set in MSTJCLxx (pointed to by the IEFJOBS DD statement) that will contain the tailored JCL to support started tasks.

To create source JCL that is a job (with, for example, JOB, JES2 JECL, and OUTPUT statements) for a started task, you must first decide whether these jobs should be mixed with procedures (part of the IEFPDSI concatenation) or placed into a separate data set (part of the IEFJOBS concatenation) containing only jobs.

Using IEFJOBS to define started tasks

If MSTJCLxx contains a DD named IEFJOBS, the source JCL for a started task can be placed in one of the data sets within the IEFJOBS concatenation. Doing so allows jobs and the procedures they invoke to have the same name. These jobs can contain the minimum set of JCL needed to define the job level characteristics (for example, JOB statements, JCLLIB, and JECL), and then either invoke an existing procedure or use the INCLUDE keyword to invoke the desired set of JCL.

Use a name that allows you to quickly identify the data set. For example, the entry in MSTJCLxx could appear as:

```
//IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR
```

For this example, you can put the source JCL for the started tasks in the SYS1.STCJOBS data set.

Consider these reasons for using IEFJOBS to define started tasks:

- MVS and other products ship procedures placed in procedure libraries. Your modifications to the members that contain your source JCL for started tasks might be lost if a new version of the procedure is received and placed into the procedure library.
- Defining data sets for IEFJOBS allows you to have the member name containing the source JCL invoke a procedure of the same name. For example, member name DUMPCHK in the IEFJOBS data set SYS1.STCJOBS is a job that invokes procedure DUMPCHK in SYS1.PROCLIB. This minimizes the effect on commands that might be issued from a variety of sources, and allows job parameters to be added transparently.

  Note: Do not attempt to change the IEESYSAS procedure to a started job, or place a member named IEESYSAS in the IEFJOBS data set concatenation. IEESYSAS is reserved for use by the system for starting address spaces.
- Maintaining procedures in procedure libraries and jobs in an IEFJOBS data set can reduce potential confusion. For example, if you place a job in a procedure library, a user could mistakenly assume that the job is a procedure and invoke it as a procedure within a job; a job invoked as a procedure within a job fails.

Using IEFPDSI to define started tasks

As shipped, MSTJCL00 contains an IEFPDSI DD statement that points to only one procedure library (SYS1.PROCLIB) used by the master subsystem. You can place individual jobs in any other procedure libraries pointed to by the IEFPDSI DD statement.

As an alternative to placing individual jobs in any of the other procedure libraries pointed to by the IEFJOBS DD statement, you can instead put jobs in the members of SYS1.PROCLIB or other data sets in the IEFPDSI concatenation. In this case, you must make sure that there is not another member of the same name in one of the data sets in the concatenation ahead of the data set containing the member with the source JCL.

Started task processing

When you start a started task, the system determines whether the START command refers to a procedure or a job. (The system validates that the IEFJOBS DD exists within the MSTJCLxx member.) If the IEFJOBS DD exists, the system searches the IEFJOBS DD concatenation for the member requested in the START command. If there is no member by that name in the IEFJOBS concatenation, or if the IEFJOBS concatenation does not exist, the system searches the IEFPDSI DD for the member requested in the START command.

If a member is found, the system examines the first record for a valid JOB statement and, if one exists, uses the member as the JCL source for the started task.

If the member does not have a valid JOB statement in its first record, the system assumes that the source JCL is a procedure and creates JCL to invoke the procedure. After JCL source has been created (or found), the system processes the JCL.

Writing your own master scheduler JCL

To write your own master scheduler JCL, specify the following JCL statements for the master scheduler JCL data set:

//MSTJCLxx JOB
A JOB statement is required.

// EXEC
An EXEC statement is required with PGM=IEEMB860. Program IEEMB860 is the master scheduler that does the IPL processing.

The EXEC statement must contain a time limit of 1440 or the program must be defined as a system task in the program properties table (PPT) to ensure that the master scheduler does not time out. See the description of the SCHEDxx parmlib member in Chapter 77, "SCHEDxx (PPT, master trace table, and abend codes for automatic restart)," on page 729 for information about allowing your installation to specify programs in the PPT.

The master scheduler (IEEMB860) is defined in the PPT with several options; among them are SYST (system task) and NODSI (no data set integrity). SYST ensures that the master scheduler does not time out. NODSI means that the master scheduler JCL defined data sets are not enqueued or shared by MVS. This keeps the master scheduler from having to deal with enqueue contention for data sets, such as SYS1.PARMLIB. It also means that it is possible to move members within the data set while the system is trying to read them. This can result in I/O errors. Any manipulation of the data sets defined to the master scheduler JCL should be done with extreme caution.

//STCINRDR DD
This DD statement defines the internal reader where started tasks are to be sent.

//TSOINRDR DD
This DD statement defines the internal reader where TSO logon started tasks are to be sent.

//IEFJOBS DD
This DD statement defines the data set that contains job source JCL for started tasks. This data set can be a PDSE and can be SMS managed. The data set can also be a concatenation.

//IEFPDSI DD
This DD statement defines the data set that contains procedure source JCL for started tasks. Normally this data set is SYS1.PROCLIB; it can be a concatenation. For useful work to be performed, the data set must at least contain the procedure for the primary JES.

//IEFPARM DD
Optionally, this DD statement defines SYS1.PARMLIB or an equivalent for use by the master scheduler address space following master scheduler initialization; it can be a concatenation. Some subsystems and address spaces use other techniques to read parameter information from SYS1.PARMLIB or other sources.

If you specify this DD statement and there are parmlib statements in the LOADxx member of SYS1.PARMLIB, IEFPARM is ignored, a warning message is issued and the LOADxx parmlib data sets are used. If the specified parmlib statements in LOADxx cannot be found, IEFPARM is ignored, a warning message is issued and the system uses SYS1.PARMLIB as the default. If there are no parmlib data sets specified in LOADxx, the system uses the parmlibs on the IEFPARM DD statement.

If you do not specify this DD statement, the system uses the parmlib data sets specified in the LOADxx member of SYS1.PARMLIB. If no parmlib data sets were specified in LOADxx, or if the specified parmlib data sets cannot be found, the system uses SYS1.PARMLIB as the default. If no SYS1.PARMLIB is cataloged, the system uses the SYS1.PARMLIB on the SYSRES volume.

//SYSUADS DD
Optionally, this DD statement is defined for TSO/E.

//SYSRACF DD
Optionally, this DD statement is defined for RACF. IBM suggests the use of ICHRDSNT for the specification of the primary and backup RACF databases. See *z/OS Security Server RACF System Programmer's Guide* for more information.

Note: You can only specify a primary RACF database name with the SYSRACF DD.

/* Ends the JCL.

Also:

- The master scheduler JCL can contain only one step.
- MVS system commands can be placed in the master scheduler JCL. These commands are read in before the master scheduler is executed. They are normally processed before those commands that are specified in the COMMNDxx member of Parmlib.
- OUTPUT JCL and DD SYSOUT statements can be placed in the master scheduler JCL.

- The master scheduler JCL can contain a JOBLIB or STEPLIB DD statement (which are usually used for testing).
- Other DD statements that may be needed during master scheduler processing, such as RACF-related DD statements, can be placed in the master scheduler JCL.
- The master scheduler JCL no longer has to point to the broadcast data set. You now specify the broadcast data set in the IKJTSOxx member. The master JCL will no longer allocate the broadcast data set. TSO/E will use either the default (SYS1.BRODCAST) or the BROADCAST parameter in IKJTSOxx to allocate the broadcast data set. If you wish to go on using SYS1.BRODCAST, no action is required. If you want to use a broadcast data set other than SYS1.BRODCAST, specify it in IKJTSOxx.
- All data sets that are defined in the master scheduler JCL and do not specify VOL=SER=volser on the data set name must be cataloged in the master catalog.
- The master scheduler JCL cannot contain any EXEC PROC statements.
- The master scheduler JCL runs in the master scheduler's address space. The TIOT size is 4K, which means that approximately 250 unit allocations can be done in the master scheduler's address space. Therefore, be careful not to exceed this limit when allocating data sets in the master scheduler JCL.
- The master scheduler JCL does not support the following JCL statements:
  - ELSE
  - ENDIF
  - IF/THEN
  - INCLUDE
  - JCLLIB
- The master JCL can contain the JCL statements COMMAND and SET.
- The master JCL can contain system symbols. Remember, however, that the system does not process symbols in MSTJCLxx in the same way that it processes symbols in parmlib members. Because MSTJCLxx contains JCL, the system processes symbols in MSTJCLxx during JCL processing. The results of symbolic substitution reflect the substitution rules that are in effect during JCL processing.

  See Chapter 2, “Sharing parmlib definitions,” on page 33 for general information about defining and using system symbols in parmlib. See *z/OS MVS JCL Reference* for details about using system symbols in JCL.
- The master JCL cannot contain the &SYSUID symbol. &SYSUID, which is normally allowed in JCL, is not defined for the master JCL.

Overview of parmlib members

Table 2 contains an overview of parmlib members. Note that IBM-supplied parmlib members use '00' for 'xx'.

Table 2. Characteristics of Parmlib Members

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
ADYSETxx: Parameters that control dump analysis and elimination (DAE) processing.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
yes	optional	yes	At IPL and SET DAE command.	no	Error message requires the operator to correct the parmlib member SET DAE command.	Error message requires the operator to start DAE again (using the SET DAE command) after the problem is fixed.	N/A	yes	yes
ALLOCxx: Parameters that control installation defaults for allocation values.									
yes, in SYS1. SAMPLIB	optional	yes	IPL	no	Error message requires the operator to correct the parmlib member.	Error message is issued.	Error message is issued.	yes	yes
ANTMIN00: Parameters that control ANTMAIN.									
Yes	optional	no	IPL, ANTMAIN startup	no	Error message is issued.	Error message is issued.	Error message is issued.	no	yes
ANTXIN00: Parameters that control Extended Remote Copy (XRC).									
Yes	optional	yes	XSTART and XSET commands, ANTASnnn ANTCLnnn address space startup	yes	Error message is issued.	Error message is issued.	Error message is issued.	no	yes
APPCPMxx: Parameters that define or modify the APPC/MVS configuration.									
yes, in SYS1. SAMPLIB	optional	yes	START APPC or SET APPC operator command	yes	Error message requires the operator to correct the parmlib member.	Error message is issued.	Error message is issued.	yes	yes
ASCHPMxx: Parameters that define scheduling information for the ASCH transaction scheduler.									
yes, in SYS1. SAMPLIB	optional	yes	START ASCH or SET ASCH operator command	yes	Error message requires the operator to correct the parmlib member.	Error message is issued.	Error message is issued.	yes	yes
AUTORxx: Parameters that change the auto-reply processing settings on a system.									
yes	optional	no	IPL and SET AUTOR command	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
AXR00: Parameters that are needed when installation changes to default settings are required.									
yes, in SYS1. SAMPLIB	optional	no	IPL	no	Error messages require the operator to correct the parmlib member.	Error message is issued.	Error message is issued.	yes	no
BLSCECT: Parameters that control dump formatting performed by IPCS, and by the system (through the ABEND and SNAP macros).									
yes	optional	no	At IPL and at START BLSJPRMI	no	Error message is issued.	Error message is issued.	N/A	no	yes

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
BLSCUSER: Parameters that specify IPCS customization.									
no	optional	no	At IPL and at START BLSJPRMI	no	Error message is issued.	Error message is issued.	N/A	no	yes
BPXPRMxx: Parameters that control the OS/390 UNIX System Services environment and the hierarchical file system (HFS). The system uses these values when initializing the OS/390 UNIX System Services kernel.									
yes, in SYS1. SAMPLIB	optional	no	IPL	no	Error message requires the operator to correct the parmlib member.	Error message is issued.	Error message is issued.	yes	yes
CEAPRMxx: Parameters that customize common event adapter (CEA) values.									
yes, in SYS1. PARMLIB	required	no	At IPL and F CEA,CEA= command	yes, through `F CEA,DISPLAY, PARMS` command.	Error message is issued.	Error message is issued.	Error message is issued.	no	no
CEEPRMxx: Parameters that control default Language Environment® runtime options for a system.									
yes, in SYS1. SAMPLIB	optional	no	At IPL and SET CEE command	yes	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
CLOCKxx: Parameters that control operator prompting to set the TOD clock, specifying the difference between the local time and UTC, and ETR usage.									
yes	optional	no	IPL	yes	Prompts operator to respecify, or cancel parameter through ENTER key.	Prompts operator to respecify, or cancel parameter through ENTER key.	Prompts operator to respecify, or cancel parameter through ENTER key.	yes	yes
CNGRPxx: Parameters that define console groups.									
no	optional	no	At IPL and SET CNGRP command.	no	Error message is issued.	Operator is prompted for a new CNGRPxx member.	Error message.	yes	yes
CNLcccxx: Defines how translated messages are to be displayed at your installation.									
yes	optional	no	At IPL and SET MMS command.	no	Error message is issued.	Error message is issued.	Error message is issued.	no	yes
COFDLFxx: Allows a program to store DLF objects that can be shared by many jobs in virtual storage managed by Hiperbatch.									
yes	optional	yes	At IPL and START DLF command.	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
COFVLFxx: Allows an authorized program to store named objects in virtual storage managed by VLF.									
yes	optional	yes	START VLF command.	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
COMMNDxx: Commands to be issued by the control program immediately after initialization. JES commands may not be included.									
no	optional	no	IPL	no	Invalid commands are ignored and an error message is issued.	No commands are processed.	N/A	yes (see Note 4 on page 27)	yes
CONFIGxx: Allows the installation to define a standard configuration that is compared with current configuration and to reconfigure processors, storage, and channel paths.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
no	optional	no	DISPLAY M command with CONFIG option and CONFIG command with MEMBER option.	no	Invalid commands are ignored and an error message is issued.	Processing is terminated and an error message is issued.	N/A	yes	yes
CONSOLxx: Parameters to define an installation's console configuration, initialization values for console services, the default routing codes for all WTO/WTOR messages that have none assigned, and the characteristics of the hardcopy message set. CONSOLxx also contains parameters that define the hardcopy medium.									
yes, IEACONxx in SYS1. SAMPLIB	required (unless joining a sysplex)	no	IPL and SET CON= command.	Yes, if L option is specified with the CON parameter in IEASYSxx or by the operator. The L option is only valid at IPL	Error message is issued.	Operator is prompted for a new CONSOLxx member.	Error message.	yes	yes
COUPLExx: Describes the systems complex (sysplex) environment for the system.									
yes	required	yes	IPL Note: Options changed using SETXCF command	no	Error message is issued. The operator may also be prompted for a new COUPLExx member.	The operator is prompted for a new COUPLExx member.	Error message.	yes	yes
CSVLLAxx: Allows an installation to list the entry point name or LNKLST libraries that can be refreshed by the 'MODIFY LLA, UPDATE=xx' command.									
no	optional	no	MODIFY LLA and START LLA commands	yes	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
CTncccxx: Specifies component trace options.									
yes	depends on the component	no	At IPL or TRACE CT command	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
CUNUNIxx: Specifies parameters for Unicode conversion services.									
no	optional	no	Both	Yes, messages are issued when Unicode conversion services are active	Error message is issued.	Error message is issued.	Error message is issued.	No	No
DEVSUPxx: Allows an installation to specify whether data will be stored in a compacted format on a 3480 or 3490 tape subsystem with the Improved Data Recording Capability feature.									
no	optional	no	IPL	no	Error message is issued.	Error message is issued.	Error message is issued.	no	yes
DFHSSIxx: Specifies message-formatting initialization parameters for the CICS® subsystem of the SYS1.PARMLIB library.									
yes	optional	no	IPL	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
DIAGxx: Contains diagnostic commands that control the common storage tracking and GETMAIN/FREEMAIN/STORAGE (GFS) trace functions.									
yes	optional	no	At IPL or SET DIAG command	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
EPHWP00: Allows z/OS UNIX to use man pages.									
yes	Required for z/OS UNIX to use man pages.	no	Any BookRead invocation.	no	Error message is issued.	Error message is issued.	Error message is issued.	no	no
EXITxx: Allows an installation to specify installation exits for allocation decisions.									
no	optional	no	IPL	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	no
EXSPATxx: Allows an installation to specify actions taken to recover from excessive spin conditions without operator involvement.									
no	optional	no	SET EXS command	no	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
GRSCNFxx: Configuration parameters for systems that are to share resources in a global resource serialization complex.									
yes	Required only if GRS=START, GRS=JOIN or GRS= TRYJOIN is specified. Ignored if GRS=NONE is specified.	yes, by specification of the RESMIL parameter.	IPL	no	Error message requires the operator to either correct the parmlib member and restart the system or reply NONE.	Error message requires the operator to either restart the system after the problem is fixed or reply NONE.	N/A	yes	yes
GRSRNLxx: Resource name lists (RNLs) that the system uses when a global resource serialization complex is active.									
yes	Required only if GRS=START, GRS=JOIN or GRS= TRYJOIN is specified and GRSRNL= EXCLUDE is not specified. Ignored if GRS=NONE is specified.	yes	IPL or SET GRSRNL command	no	At IPL: Error message requires the operator to either correct the parmlib member and restart the system or reply NONE. On SET GRSRNL command: Error message and the command is not done.	At IPL: Error message requires the operator to either restart the system after the problem is fixed or reply NONE. On SET GRSRNL command: Error message and the command is not done.	N/A	yes	yes
GTFPARM: Parameters to control GTF.									
yes	Optionally used. Can be modified by installation.	Indirectly because SRM sysevent trace option is a tuning aid.	START GTF	Yes, automatically at START GTF.	On any error, prompts for all parameters.	Same as with syntax error.	N/A	no	no
HZSPRMxx: Contains statements that define overrides to IBM Health Checker for z/OS check defaults and specify the IBM Health Checker for z/OS policy or policies for managing checks.									
no	optional	no	S *hzsproc* or F *hzsproc* command.	Yes, through `F hzsproc ,DISPLAY` command	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
IDAVDTxx: Parameters for VSAM dynamic trace.									
IEAABD00: Default parameters for an ABEND dump when a SYSABEND DD statement has been specified.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
yes	Optional. However, if member is unavailable, ABEND dumps may not be possible without DUMPOPT lists.	no	IPL	no	Message lists valid parameters that were accepted. Invalid parameters are rejected.	Error message. Parameters are rejected.	Error message is issued.	no	yes
IEAAPFxx: Names of authorized program libraries.									
no	optional. SYS1. LINKLIB and SYS1.SVCLIB are always authorized (see Note 5 on page 27).	no	IPL	no	Prompts operator to respecify bad parameters or cancel parameters through ENTER key.	Same as with syntax error.	N/A	no	yes
IEAAPP00: Names of authorized installation-written I/O appendage routines.									
no	optional	no	IPL	no	Error message. Partial appendage name table is built, if possible.	Same as with syntax error.	N/A	yes	yes
IEACMD00: IBM-supplied commands that are processed during system initialization.									
yes	optional	yes	IPL	no	Ignores invalid command name.	No commands are processed.	N/A	yes	yes
IEADMCxx: Parameters for DUMP command.									
no	optional	no	DUMP command	no	Error message. Parameters are rejected.	Error message. Parameters are rejected.	N/A	yes	yes
IEADMP00: Default parameters for an ABEND dump when SYSUDUMP DD statement has been specified.									
yes	Optional. However, if member is unavailable, ABEND dumps may not be possible without DUMPOPT lists. Can be modified by installation.	no	IPL	no	Message lists valid parameters that were accepted. Invalid parameters are rejected.	Error message. Parameters are rejected.	N/A	no	yes
IEADMR00: Default parameters for an ABEND dump when SYSMDUMP DD statement has been specified.									
yes	Optional. However, if member is unavailable, ABEND dumps may not be possible without DUMPOPT lists. Can be modified by installation.	no	IPL	no	Message lists valid parameters that were accepted. Invalid parameters are rejected.	Error message. Parameters are rejected.	N/A	no	yes

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
IEAFIXxx: Names of modules to be fixed in storage for the duration of the IPL.									
no	optional	yes	IPL	yes, if L option is specified with FIX parameter in IEASYSxx, or by the operator.	Prompts operator to respecify bad parameters or cancel them through ENTER key.	Same as with syntax error.	Error message. Obsolete module names are ignored.	yes	yes
IEALPAxx: Names of reenterable modules that are to be loaded as a temporary extension to the PLPA.									
no	optional	yes	IPL	yes, if L option is specified with the MLPA parameter in IEASYSxx, or by the operator.	Prompts operator to respecify, or cancel parameter through ENTER key.	Same as with syntax error.	Error message. Ignores obsolete module names.	yes	yes
IEAOPTxx: Parameters that control resource and workload management algorithms in the system resources manager. Beginning with z/OS V1R3, WLM compatibility mode is no longer available. Accordingly, you can no longer use any of the IEAOPTxx options that were valid in compatibility mode only. The information has been left here for reference purposes, and for use on previous systems.									
IEAOPT00: yes IEAOPTxx: no	optional	yes	At IPL and SET OPT command.	yes, if L option is specified with the OPT parameter in IEASYSxx, or by the operator.	Prompts operator to respecify or cancel parameter through ENTER key. If parameter is canceled, various defaults are used.	Same as with syntax error.	N/A	yes	yes
IEAPAKxx: "Pack List" names of groups of modules in the LPALST concatenation that the system will load between page boundaries to minimize page faults.									
No. Optionally, installation can provide before IPL.	optional	yes	IPL	no	Bypasses the pack group that contains the error. Processes the next group pack.	The pack groups read before the errors are processed. Other pack groups are omitted.	Error message. Ignores obsolete modules names.	yes	yes
IEASLPxx: Contains valid SLIP commands.									
Yes, in SYS1. PARMLIB.	required if SET SLIP=00 command is specified in IEACMD00.	no	At IPL and SET SLIP command	no	Normal SLIP response to prompt the operator.	Processing stops.	Syntax error message is issued.	yes	yes
IEASVCxx: Allows the installation to define its own SVCs in the SVC table.									
no	optional	no	IPL	Yes, if L option is specified with the SVC parameter in IEASYSxx, or by the operator.	Message is issued and the statement in error is not processed. Processing continues with the next statement.	Same as with syntax error.	Same as with syntax error.	yes	yes
IEASYMxx: Specifies, for one or more systems in a multisystem environment, the static system symbols and suffixes of IEASYSxx members that the system is to use. One or more IEASYMxx members are selected using the IEASYM parameter in the LOADxx parmlib member.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
no	optional	no	IPL	L parameter must be specified with parameter or number.	Prompts operator to respecify or cancel parameter through ENTER key. Parameters processed before the error are discarded. Also, in some error situations, a wait state occurs.	Same as with syntax error.	Same as with syntax error.	yes	yes
IEASYSxx: System parameters that are valid responses to the SPECIFY SYSTEM PARAMETERS message. Multiple system parameter lists are valid. The list is chosen by the operator SYSP parameter or through the SYSPARM statement of the LOADxx parmlib member (see Chapter 68, "LOADxx (system configuration data sets)," on page 619 for more information).									
no	See Note 2 on page 27.	See Note 3 on page 27.	IPL	See Note 1 on page 27 (L parameter must be specified with parameter or number).	Prompts operator to respecify, or cancel parameter through ENTER key. Parameters processed before the error are retained.	Parameters processed before the error are retained. Operator is asked to specify an alternate member. If he does, new parameters override those retained.	Obsolete parameters are treated as syntax errors.	yes	yes
IECIOSxx: Parameters that control missing interrupt handler (MIH) intervals and update hot I/O detection table (HIDT) values.									
no	optional	no	IPL or SET IOS command (to change the MIH values only).	no	Message is issued and default value is substituted.	Same as with syntax error.	Obsolete parameters are treated as syntax errors.	yes	yes
IEFSSNxx: Parameters that identify what subsystems are to be initialized.									
yes	required	yes, depending on the function of each subsystem.	IPL	no	Message issued identifying erroneous record. Next record is read and processed.	Processing of the parmlib members is terminated and an error message is issued.	N/A	yes	yes
IFGPSEDI: Initializes the Enhanced Data Integrity(EDI) for shared physical sequential data sets function.									
no	optional	no	At IPL and START IFGEDI command.	no	Error message requires the operator to correct the parmlib member.	Same as with syntax error.	Same as with syntax error.	no	no
IFAPRDxx: Parameters that define a product enablement policy.									
yes	optional	no	At IPL and SET PROD command.	no, but information is available through DISPLAY.	Skips to next statement in member.	Processing is terminated and an error message is issued.	Skips to next statement in member.	yes	yes

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
IGDSMSxx: Initialize the Storage Management Subsystem (SMS) and specify the names of the active control data set (ACDS) and and the communications data set (COMMDS).									
no	required	no	At IPL and SET SMS command.	yes	Default value is substituted.	Same as with syntax error.	Obsolete parameters are ignored.	yes	yes
IKJTSOxx: For TSO/E specifies authorized commands and authorized program, programs that are authorized when called through the TSO service facility, commands that may not be issued in the background, and defaults for SEND and LISTBC processing.									
yes, in SYS1. SAMPLIB	optional	no	IPL or TSO/E PARMLIB command	no	Error message is issued. Processing continues.	Same as with syntax error.	N/A	yes	yes
IOEPRMxx: Parameters that control the processing options for the zFS PFS.									
yes	optional	yes	IPL and start or restart of zFS (using `SETOMVS =(xx)` or reply R to BPXF032D message)	yes, using `MODIFY ZFS,QUERY, SETTINGS` operator command	Ignored.	Error message is issued	Ignored.	yes	yes
IPCSPRxx: Parameters that are used during an IPCS session.									
yes	optional	no	IPCS command initialization	no	Error message is issued. IPCS session continues.	Same as with syntax error.	Same as with syntax error.	no	yes
IQPPRMxx: Defines parameters for PCIE related devices.									
no	optional	yes	Both IPL and SET IQP command	Yes, if L parameter specified with the IQP parameter in IEASYSxx, or by the operator	Error message is issued	Error message is issued	Error message is issued	yes	yes
IVTPRM00: Sets Communications Storage Manager (ICSM) parameters.									
yes, as a VTAM® component	optional	yes	Initialized by first `CREATE _POOL` request received.	yes	Error message is issued. Processing continues.	Same as with syntax error.	Same as with syntax error.	yes	yes
IXGCNFxx: Parameters that control processing options for system logger.									
yes, in SYS1. SAMPLIB	optional	yes	At IPL and restart of IXGLOGR (through Start IXGLOGRS proc) or SET IXGCNF command.	Yes, merged information is available through `DISPLAY LOGGER,IXGCNF` command.	At IPL: Error message is issued and default values are substituted for all parameters.	Same as with syntax error.	Same as with syntax error.	yes	yes
LNKLSTxx: List of data sets to be concatenated to form the LNKLST concatenation. You can also use PROGxx to define the concatenation.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
Yes, in SYS1. SAMPLIB	required if PROGxx is not used for LNKLST concatenation; otherwise, ignored if PROGxx is used.	no	IPL	Yes, if L option is specified with LNK parameter in IEASYSxx, or by the operator.	Prompts operator to respecify, or cancel parameter through ENTER key.	No data sets are concatenated to SYS1. LINKLIB. Operator can re-IPL to specify an alternate LNKLSTxx member.	Error message. Data sets not found will not be concatenated to SYS1. LINKLIB	yes	yes
LOADxx: Specifies data sets MVS uses to configure your system.									
No. (See the `IPXLOADX` member of SYS1. SAMPLIB for JCL to create a sample LOADxx member.)	required	no	IPL - from LOAD parameter on system console	no	Wait state is loaded. Error message is issued.	Wait state is loaded.	Error message is issued.	Supports `&SYSR1` as a volser. No other system symbol support.	no
LPALSTxx: List of data sets to be concatenated to SYS1.LPALIB from which the system builds the pageable LPA (PLPA).									
no	optional	no	IPL	Yes, if L option is specified with LPA parameter in IEASYSxx, or by the operator.	Any valid data set definitions preceding the syntax error are concatenated to SYS1. LPALIB.	No data sets are concatenated to SYS1. LPALIB. Operator can reIPL to specify an alternate LPALSTxx member.	N/A	yes	yes
MMSLSTxx: Specifies information that the MVS message service (MMS) uses to control the languages that are available in your installation.									
yes, in SYS1. SAMPLIB	optional	no	IPL using the CONSOLxx member option. Modify with SET MMS command	Yes, through the DISPLAY command	Message is issued for each invalid member.	Parameters are rejected.	Same as with syntax error.	yes	yes
MPFLSTxx: Parameters that the message processing facility uses to control message processing and display.									
no	optional	no	IPL using the CONSOLxx member option. Modify with SET MPF command	Yes, through the DISPLAY command.	Message is issued for each invalid member.	Message is issued; processing continues.	Same as with syntax error.	yes	yes
MSGFLDxx: Parameters that message flood automation uses to detect and handle message floods.									
yes, as `CNZZMFXX` in SYS1. SAMPLIB	optional	no	SET MSGFLD command	yes, through the `DISPLAY MSGFLD` command	Error message is issued. Processing continues.	Same as with syntax error.	Same as with syntax error.	no	yes
MSTJCLxx: Contains the master scheduler JCL that controls system initialization and processing.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
yes, in SYS1. LINKLIB	required	no	IPL	yes	Abend issued and the statement in error is not processed. The system abnormally ends master scheduler initialization.	Same as with syntax error.	Same as with syntax error.	yes (see Note 4 on page 27)	yes
NUCLSTxx: Specifies members of SYS1.NUCLEUS to be included in, or excluded from, the nucleus region at IPL-time.									
no	optional	no	IPL	no	Wait state is loaded; error message is issued.	Wait state is loaded.	N/A	no	no
PFKTABxx: Parameters contain the definitions for program function key tables (PFK tables).									
no	optional	no	IPL or SET PFK command.	no	Error message is issued.	Error message is issued.	N/A	yes	yes
PROGxx: Contains statements that define the format and contents of the APF-authorized program library list, control the use of installation exits and exit routines, define the LNKLST concatenation, and specify alternate data sets for SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE to appear at the beginning of the LNKLST concatenation and SYS1.LPALIB to appear at the beginning of the LPALST concatenation.									
no	optional; required if LNKLSTxx is not used for the LNKLST concatenation	no	IPL or SET PROG command.	Yes through DISPLAY PROG	Error message is issued.	Error message is issued.	Error message is issued.	yes	yes
SCHEDxx: Provides centralized control over the size of the master trace table, the completion codes to be eligible for automatic restart and programs to be included in the PPT.									
no	optional	no	At IPL and SET SCH command	Yes, (L) is specified in IEASYSxx or in response to the specify system parameters prompt.	Diagnostic error message is issued. Different processing is done for each statement type in IEASYSxx.	Diagnostic error message is issued and the next record is read.		yes	yes
SMFPRMxx: Parameters that define SMF options.									
yes, in SYS1. SAMPLIB	optional	yes	IPL or SET SMF command	Yes, if PROMPT (list) or PROMPT (ALL) parameter is specified.	Prompts operator to enter parameter or re-IPL.	Same as with syntax error.	Obsolete parameters are ignored.	yes	yes
TSOKEYxx: VTIOC parameters that are used by TSO/VTAM time sharing.									
no	optional	yes	START TSO command	Yes, automatically at START TSO.	Default value is substituted.	Default value is substituted.	N/A	yes	yes
VATLSTxx: Volume attribute list that defines the "mount" and "use" attributes of direct access volumes.									
no	optional	yes	IPL	No. Operator has option to get list only if error occurs.	Error message. Bypasses bad entry. Processes remaining entries.	Operator is given choice of processing remaining list(s) if multiple lists, or specifying new VATLST VATLSTxx member, or reIPLing.	N/A	yes	yes
XCFPOLxx: Specifies the actions that a system in a sysplex on PR/SM™ is to take when another system in the sysplex becomes inactive.									

Table 2. Characteristics of Parmlib Members (continued)

Supplied by IBM	Required or Optional	Affects performance directly?	Read at IPL or at command?	Allows listing of parameters at IPL or command?	Response to errors (N/A = not applicable)			System symbols support?	Concatenated parmlib support?
					Syntax error	Read error	Unsupported parameters		
yes	optional	no	SETXCF PRSMPOLICY command.	no	Error message requires the operator to correct the parmlib member	Error message requires the operator to correct the parmlib member	Error message requires the operator to correct the parmlib member	yes	no

Note:

1. These parameters can be listed at IPL: APF, DUMP, FIX, MLPA, SYSP, and OPT.
2. The only mandatory parameter is PAGE. Other parameters have coded defaults.
3. Performance-oriented parameters in IEASYSxx: CMD, FIX, MLPA, OPT, REAL, RSU
4. System symbols in COMMNDxx and MSTJCLxx are processed differently than system symbols in other parmlib members. See Chapter 21, "COMMNDxx (commands automatically issued at initialization)," on page 219 and Chapter 73, "MSTJCLxx (master scheduler JCL)," on page 687 for details.
5. The system also treats all modules that are part of the link pack area (pageable LPA, fixed LPA, modified LPA, and dynamic LPA) as having come from an APF-authorized library.

Implicit system parameters

Various system requirements, although not involving explicit parameters, affect the way the system performs. These system requirements may be considered as "implicit" parameters. They involve DD statements, data sets, hardware choices, and so forth. Some examples are:

- SYSABEND, SYSMDUMP, and SYSUDUMP DD statements. Without these statements, the parameters in parmlib members IEAABD00, IEADMR00, and IEADMP00 and the dump option lists in ABEND macro instructions are meaningless because an ABEND dump cannot be taken.
- The SMF data sets. If these data sets are not allocated on direct access volumes and cataloged, no SMF recording occurs.
- Addition of new modules to the LPALST concatenation through use of IEBCOPY, the Linkage Editor or the binder. Adding new modules affects the size and usefulness of the PLPA that is loaded by specifying the CLPA parameter at IPL.
- Choice of the device on which the PLPA paging data sets will reside. This choice affects the speed at which PLPA modules can be paged into storage and thus influences system performance.
- Definition of page data sets through the DEFINE PAGESPACE command. The PAGE parameter, issued at IPL through parmlib and/or the operator, is significant only if the specified data sets have been previously formatted by the DEFINE PAGESPACE command. (For information about this command, see *z/OS DFSMS Access Method Services Commands*.)

Managing system security — APF-authorized library list

The authorized program facility (APF) allows your installation to identify system or user programs that can use sensitive system functions. To be APF-authorized, programs must reside in APF-authorized libraries, and be link-edited with authorization code AC=1. The system maintains a list of APF-authorized libraries that contains the following information for each library:

- The library name
- An identifier for the volume that contains the library.

The system automatically places SYS1.LINKLIB and SYS1.SVCLIB in the first two APF list entries. Your installation can specify the remaining entries in the APF list. In addition, any module in the link pack area will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Note: When LNKAUTH=APFTAB is specified, the system considers SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE to be APF-authorized when they are accessed as part of the concatenation (even when they are not included in the APF list).

Choosing an APF list format

You can specify the list of APF-authorized libraries in a dynamic or static format. There is only one APF list, the format of which is either static or dynamic. With the dynamic format, you can take the following actions:

- Update the APF list without having to reIPL the system, and
- Specify as many APF-authorized libraries as you need; there is no system-imposed maximum number.

In contrast, you can update a static APF list only at IPL, and it can contain a maximum of 255 entries. IBM suggests that you use a dynamic APF list to take advantage of the dynamic update capabilities.

Before you change the format of the APF list to dynamic, ensure that programs and vendor products are converted to use dynamic APF services (see Chapter 41, “IEAAPFxx (authorized program facility list),” on page 371 for information about converting to a dynamic APF list) and that the proper program products are installed (see *z/OS Planning for Installation*).

Table 3 describes restrictions that are associated with changing the format of the APF list:

- The first column shows the format of the APF list set at IPL. The next three columns show changes to the APF list that might occur, in order, during normal processing.
- Under the "Format change allowed?" column, the table indicates whether a format change is allowed when the APF list is in the last format listed in the first four columns.
- Under the "Contents of APF list" column, the table shows the contents of the APF list when the format is the last format listed in the first four columns.

Table 3. Restrictions on changing the format of the APF list

Format set at IPL	First format change	Second format change	Third format change	Format change allowed?	Contents of APF list
Static	Not made	Not made	Not made	Yes	The libraries included at IPL
Dynamic	Not allowed	Not allowed	Not allowed	No	The libraries included at IPL, and any updates that occur during normal processing
Static	Dynamic	Not made	Not made	Yes	The libraries included at IPL, and any updates that occur during normal processing

Table 3. Restrictions on changing the format of the APF list (continued)

Format set at IPL	First format change	Second format change	Third format change	Format change allowed?	Contents of APF list
Static	Dynamic	Static	Not made	Yes	The libraries included at IPL, but not the updates that occurred during normal processing (when the format was dynamic)
Static	Dynamic	Static	Dynamic	Yes	The libraries included at IPL, and any updates that occurred during normal processing (during both instances when the format is dynamic)

IBM suggests that you take the following steps when changing the format of the APF list:

1. IPL with the APF list in a static format; then, when you are ready, change the format to dynamic during normal processing. If you IPL with a dynamic format and you find an error, you cannot convert the format back to static.
2. Convert from a dynamic format back to a static format *only* when you find an error when using the dynamic APF list. For example, a program may not be converted to use CSVAPF services to access the list (see the description of the CSVAPF macro in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for an example of how to perform this conversion). After you fix the error, you can change the format back to dynamic.
3. Specify a dynamic format for the APF list at the next IPL.

For information about restrictions that are associated with changing the format of the APF list, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Specifying the APF list

The PROGxx and IEAAPFxx parmlib members allow you to define the format and contents of the APF list. You can use either member, but IBM suggests using PROGxx, which offers the following advantages over IEAAPFxx:

- You can specify multiple PROGxx parmlib members using the PROG=xx system parameter
- You can specify a dynamic or static APF list format; the dynamic format allows you to update the APF list at any time during normal processing or at IPL.
- You can specify an unlimited number of libraries in the APF list.

Note: You can use both members, but the resulting list is dynamic only when you specify dynamic format in PROGxx.

If you specify a dynamic APF list format in PROGxx, you can specify the PROG=xx system parameter to set one or more current PROGxx parmlib members at IPL, or the operator can enter a SET PROG=xx command to set a current PROGxx parmlib member during normal processing. When the operator enters a SET PROG=xx command to update a dynamic APF list, the updated APF list remains active only for the duration of the current IPL. For information about how to use the SET PROG=xx command to specify the current PROGxx parmlib member, see *z/OS MVS System Commands*.

If you currently use the IEAAPFxx parmlib member to define the APF list, you can convert the format of IEAAPFxx to a PROGxx format using the IEAAPFPR REXX exec provided by IBM.

Assigning the RACF TRUSTED attribute

You can use RACF to assign the TRUSTED attribute to key started procedures and address spaces. Doing so generally allows the started procedure or address space to bypass RACF authorization checking and to successfully access or create any resource it needs.

A trusted started procedure or address space is treated as a z/OS UNIX superuser if a z/OS UNIX user identifier (UID) is assigned to it in the OMVS segment, even when the assigned UID is not 0.

Guidelines:

- Assign the TRUSTED attribute when one of the following conditions applies:
  - The started procedure or address space creates or accesses a wide variety of unpredictably named data sets within your installation.
  - Insufficient authority to an accessed resource might risk an unsuccessful IPL or other system problem.
- Avoid assigning TRUSTED to a z/OS started procedure or address space unless it is listed here or you are instructed to do so by the product documentation.

Assign the TRUSTED attribute to the following z/OS started tasks and address spaces:

- CATALOG
- CEA for z/OSMF ISPF applications
- DUMPSRV
- HIS
- IEEVMPCR
- IOSAS
- IXGLOGR
- JES2 or JES3
- JESXCF
- JES3AUX
- LLA
- NFS
- RACF
- RMF™
- RMFGAT
- SMF
- SMSPDSE1
- TCPIP
- VLF
- VTAM
- WLM
- XCFAS
- SMS

Optional candidates for the TRUSTED attribute include the following:

- APSWPROA, APSWPROB, APSWPROC, APSWPROM, or APSWPROT
- CEA (optional for everything except z/OSMF ISPF applications)
- DFHSM
- DFS
- GPMSERVE
- OMVS
- SMSVSAM
- zFS

For more information, see "Associating started procedures and jobs with user IDs" in *z/OS Security Server RACF System Programmer's Guide*, and "Using Started Procedures" in *z/OS Security Server RACF Security Administrator's Guide*.

Specifying installation exits

With the PROGxx and EXITxx parmlib members, you can specify installation exits. You can use either of these members, but IBM suggests using PROGxx. With PROGxx, you can take the following steps, at IPL or while the system is running:

- Add exit routines to an exit
- Change the state of an exit routine
- Delete exit routines from an exit
- Change the attributes of an exit
- Undefine implicitly defined exits
- Specify multiple PROGxx parmlib members using the PROG=xx system parameter.

You can use the PROG=xx system parameter at IPL to specify the particular PROGxx parmlib member the system is to use; or the operator can enter a SET PROG=xx command to set a current PROGxx parmlib member during normal processing. When the operator enters a SET PROG=xx command, the change remains in effect only for the duration of the current IPL. For information about how to use the SET PROG=xx command, see *z/OS MVS System Commands*.

If you currently use the EXITxx parmlib member to specify installation exits, you can convert the format of EXITxx to the PROGxx format by using the IEFEXPR REXX exec provided by IBM.

The system also allows you to specify SMF exits through the SMFPRMxx parmlib member. You can associate exit routines with these SMF exits by adding EXIT statements to PROGxx that are equivalent to those in SMFPRMxx. The EXIT statements in SMFPRMxx list the exits; the EXIT statements in PROGxx list the associated exit routines. The EXIT statements in SMFPRMxx must remain. See "Specifying SMF Exits to the Dynamic Exits Facility" on page 767 for an example of how to add EXIT statements to PROGxx.

Specifying LNKLST concatenations

The PROGxx and LNKLSTxx parmlib members allow you to specify the LNKLST concatenation. You can use either of these members, but IBM suggests using PROGxx. See PROG.

Chapter 2. Sharing parmlib definitions

This section describes how to set up parmlib so two or more systems can share it. Topics include:

- Objectives for sharing parmlib
- Filter parameters in LOADxx
- What system symbols are
- System-provided system symbols
- Planning tasks for sharing parmlib
- Defining system symbols and changing existing system symbol definitions
- Using system symbols in parmlib.
- A preprocessor to verify symbols without IPLing
- Displaying static system symbols
- Diagnosing problems with static system symbols
- Using indirect volume serial support

Objectives for sharing parmlib

In a multisystem environment, your objective should be to share one parmlib concatenation with as many systems as possible.

- Examine your installation configuration and determine the best way to consolidate all LOADxx information for all MVS images in a sysplex into one LOADxx member using the HWNAME, LPARNAME and VMUSERID parameters. The LOADxx parameters HWNAME, LPARNAME and VMUSERID provide segmentation of the processor hardware name, the logical partition name and the VM Userid. For more information about filter parameters, see Chapter 68, "LOADxx (system configuration data sets)," on page 619.
- Determine how many *members* of parmlib you can share with other systems. For example, if two systems use different CLOCKxx members, and only one value in each member is different, chances are that the two systems can share one CLOCKxx member.
- If you cannot share members, determine if you can share system parameter definitions in one IEASYSxx member. For example, if you must use separate CLOCKxx members for two different systems, you can code the CLOCK=xx parameter in a shared IEASYSxx member that specifies the unique CLOCKxx member for each system.

System symbols are the elements that represent unique values in shared members. When determining which members are to be shared, you must consider which system symbols are available to you, the values of those system symbols, and how you can use them to optimize the sharing of parmlib.

The following sections explain how to use system symbols and plan for parmlib sharing.

What are system symbols?

System symbols are elements that allow systems to share parmlib definitions while retaining unique values in those definitions. System symbols act like variables in a program; they can take on different values, based on the input to the program. When you specify a system symbol in a shared parmlib definition, the system

symbol acts as a "place holder". Each system that shares the definition replaces the system symbol with a unique value during initialization.

Restriction: A symbol definition will fail if it begins with a number. If any symbol in a symbol file is in error, the entire file of symbols is invalidated.

Before you share parmlib definitions, you must understand the types of system symbols and the elements that comprise them.

The following terms describe the elements of system symbols:

Symbol Name
The name that is assigned to a symbol. It begins with an ampersand (&) and optionally ends with a period (.). System symbols should usually be specified in uppercase. There are places, for example, operator commands, where system symbols can be specified in lowercase.

Substitution Text
The character string that the system substitutes for a symbol each time it appears. Substitution text can identify characteristics of resources, such as the system on which a resource is located, or the date and time of processing. When you define static system symbols in the IEASYMxx parmlib member (see "Step 4. Create an IEASYMxx parmlib member" on page 44), the substitution text can contain other static system symbols; the *resolved substitution text* refers to the character string that is produced after all symbols in the substitution text are resolved.

The following terms describe the types of symbols:

Dynamic System Symbol
A system symbol whose substitution text can change at any point in an IPL. Dynamic system symbols represent values that can change often, such as dates and times. A set of dynamic system symbols is defined to the system; your installation cannot provide additional dynamic system symbols.

Static System Symbol
A symbol whose substitution text is defined at system initialization and remains fixed for the life of an IPL unless the SETLOAD xx,IEASYM command is issued. The SETLOAD xx,IEASYM command can be used to make changes to system symbols in the local system symbol table without initiating an IPL. See *z/OS MVS System Commands*. . (One exception, &SYSPLEX, has a substitution text that can change at an additional point in an IPL; see "Step 5. Code support for system symbols in LOADxx" on page 49 for details.) Static system symbols are used to represent fixed values such as system names and sysplex names.

Static system symbols have two types:

- System-defined static system symbols already have their names defined to the system. Your installation defines substitution texts or accepts system default texts for the static system symbols, which are:
  - &SYSCLONE
  - &SYSNAME
  - &SYSOSLVL (Note: Your installation cannot define substitution text for &SYSOSLVL.)
  - &SYSPLEX
  - &SYSR1 (Note: Your installation cannot define substitution text for &SYSR1.)

  - &SYSALVL (Your installation cannot define substitution text for &SYSALVL.)
- Installation-defined static system symbols are defined by your installation. The system programmer specifies their names and substitution texts in the SYS1.PARMLIB data set.

In addition to the system symbols listed above, the system allows you to define and use the following types of symbols:

- JCL symbol is a symbol that represents variable information in JCL. You can define JCL symbols on EXEC, PROC, and SET statements in JCL, and use them only in:
  - JCL statements in the job stream
  - Statements in cataloged or in-stream procedures
  - DD statements that are added to a procedure.

  For more information about using JCL symbols, see *z/OS MVS JCL Reference*.
- IPCS symbol is a symbol that IPCS uses to represent data areas in dumps that are processed with IPCS subcommands.

  For more information about using IPCS symbols, see *z/OS MVS IPCS User's Guide*.

The following sections describe the static and dynamic system symbols that you can define and use in parmlib.

Note: Although IBM suggests the use of ending periods on system symbols, the text in the documentation does not specify them, except in examples, out of consideration for readability.

Static system symbols

The system substitutes text for static system symbols when it processes parmlib members. For static system symbols that the system provides, you can define substitution texts in parmlib or accept the default substitution texts. You can also define at least 800 additional static system symbols. See "Setting up a shared parmlib" on page 39 for more information about how to define static system symbols. Table 4 describes the static system symbols.

Table 4. Static system symbols

System symbol	Description	Text length	Where defined	Valid releases	Default substitution text
&SYSALVL	The architecture level of the system.	1 char	LOADxx parmlib member	OS/390® Release 10 and later	The value specified for ARCHLVL in the LOADxx parmlib member
&SYSCLONE	Shorthand notation for the name of the system; often used in fields that are limited to two characters.	1-2 chars	IEASYMxx parmlib member	MVS/ESA 5.2 and later	Last two characters of substitution text defined to &SYSNAME system symbol.
&SYSNAME	The name of the system.	1-8 chars	IEASYMxx or IEASYSxx parmlib member	MVS/ESA 5.1 and later	The processor identifier. See "Step 3. Determine where to specify the system name" on page 44 for details.

Table 4. Static system symbols (continued)

System symbol	Description	Text length	Where defined	Valid releases	Default substitution text
&SYSOSLVL	The version, release, and modification level of the operating system software product.	8 chars	Automatically set by the system from the ECVT data area	z/OS Version 2 Release 1 and later	Zxvvrrmm where Zx is the operating system (for example, Z1 is z/OS), vv is the version number from ECVTPVER (for example, 02), rr is the release number from ECVTPREL (for example, 01) and mm is the modification number from ECVTPMOD (for example, 00). **Example:** For z/OS 2.1, the value is Z1020100.
&SYSPLEX	The name of the sysplex.	1-8 chars	COUPLExx or LOADxx parmlib member	MVS/ESA 5.1 and later	If LOADxx does not specify the sysplex name, &SYSPLEX will default to LOCAL until the COUPLExx member is processed.
&SYSR1	The IPL Volume Serial name.	1-6 chars	Set by system to IPL volume serial	OS/390 Release 4 and later	None
Installation defined system symbols	Additional system symbols may be defined by your installation. The maximum number of symbols you can have depends on their size. IBM guarantees you the ability to define 800 symbols of your own, but you can probably define more. You may have as many symbols as you 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 & and trailing . (for example, &MYSYM.) • The substitution text value (for example, "MYVALUE")	0-8 chars	IEASYMxx parmlib member	MVS/ESA 5.2 and later	None

Dynamic system symbols

Table 5 describes the dynamic system symbols and the releases for which they are valid. The names of the following system symbols are changed from previous releases of MVS:

New symbol	Old symbol
&YYMMDD	&DATE
&LYYMMDD	&LDATE
&HHMMSS	&TIME
&LHHMMSS	&LTIME

To maintain compatibility, the original system symbols are still supported by the DUMPDS command. However, IBM suggests that you change to use the new system symbols because the substitution texts for the old system symbols might cause buffer overflows.

Table 5. Dynamic system symbols

System symbol	Description
&DATE	The date, based on Coordinated Universal Time (UTC). Equivalent to &YR2.&MON.&DAY.. Use &LDATE for local date.
&DAY	The day of the month, based on Coordinated Universal Time (UTC). Shown in two decimal digits, 01-31. Use &LDAY for local time.
&DS	The dump section number for multiple data sets in a transaction dump (TDUMP). Shown in three decimal digits, 000-999. &DS is resolved when used by IEATDUMP.
&HHMMSS	The time. Equivalent to &HR.&MIN.&SEC.. Use &LHHMMSS for local time.
&HR	The hour of the day, based on UTC. Shown in two decimal digits, 00-23. Use &LHR for local time.
&JDAY	The Julian day of the year, based on UTC. Shown in three decimal digits, 001-366. Use &LJDAY for local time.
&JOBNAME	The name of the job. Shown in 1-8 characters. In the special case when the job name is *MASTER*, &JOBNAME resolves to #MASTER# to avoid the error of using the asterisk as part of a data set name. **Note:** Under z/OS, it is valid to define a user ID that is all numeric. Under z/OS UNIX, this all-numeric user ID might be propagated as a job name in the case of a fork() or non-local spawn(); this can lead to allocation failures when &JOBNAME is used in substitution for a data set name.
&MIN	The minute of the hour, based on UTC. Shown in two decimal digits, 00-59. Use &LMIN for local time.
&MON	The month, based on UTC. Shown in two decimal digits, 01-12. Use &LMON for local time.
&SEC	The second of the minute, based on UTC. Shown in two decimal digits, 00-59. Use &LSEC for local time.
&SEQ	A sequence number for uniqueness. It is required for names of automatically allocated dump data sets. Shown in five decimal digits, 00000-99999. &SEQ is resolved only when used with DUMPDS.
&TIME	The time that the dump was requested, Equivalent to &HR.&MIN.&SEC.. Use &LTIME for local time.
&WDAY	The day of the week, based on UTC. Shown in three characters: SUN, MON, TUE, WED, THU, FRI or SAT. Use &LWDAY for local time.

Table 5. Dynamic system symbols (continued)

System symbol	Description
&YR2	The year, based on UTC. Shown in two decimal digits, 00-99. Use &LYR2 for local time.
&YR4	The year, based on UTC. Shown in four decimal digits, 0000-9999. Use &LYR4 for local time.
&YYMMDD	The date, based on UTC. Equivalent to &YR2.&MON.&DAY.. Use &LYYMMDD for local time.

Note: You can specify dynamic system symbols in parmlib. However, be aware that the system substitutes text for dynamic system symbols when it processes parmlib members. For example, if you specify &HHMMSS in a parmlib member, its substitution text will reflect the time when the member is processed.

This situation can also occur in other processing. For example, if you specify the &JOBNAME dynamic system symbol in a START command for a started task, the resolved substitution text for &JOBNAME will be the name of the job assigned to the address space that calls the symbolic substitution service, *not* the address space of the started task.

Symbols reserved for system use

When you define additional system symbols in the IEASYMxx parmlib member (see "Step 4. Create an IEASYMxx parmlib member" on page 44), ensure that you do not specify the names shown in Table 6, which are system symbols that are reserved for system use.

Table 6. Names reserved for system use

Symbol Name	Symbol Name	Symbol Name
&DATE	*&DAY*	*&HHMMSS*
&HR	*&JDAY*	*&JOBNAME*
&LDATE	*&LDAY*	*&LHHMMSS*
&LHR	*&LJDAY*	*&LMIN*
&LMON	*&LSEC*	*&LTIME*
&LWDAY	*&LYR2*	*&LYR4*
&LYYMMDD	*&MIN*	*&MON*
&SEC	*&SEQ*	*&SID*
&SYSALVL	*&SYSCLONE*	*&SYSNAME*
&SYSOSLVL	*&SYSPLEX*	*&SYSR1*
&SYSUID	*&TIME*	*&WDAY*
&YR2	*&YR4*	*&YYMMDD*

If you try to define a system symbol that is reserved for system use, the system might generate unpredictable results when performing symbolic substitution.

Setting up a shared parmlib

Before you use system symbols to share parmlib definitions that require unique values, you must evaluate the requirements for system symbols at your installation. Determine which members of parmlib require unique definitions. Then determine what system symbols you need so two or more systems can share those members.

Based on that evaluation, you can then:

- Define substitution texts for system-defined static system symbols, or accept their default substitution texts
- Determine if you need to define additional static system symbols.

The following sections explain how to set up parmlib for sharing:

Table 7. Procedure to set up parmlib for sharing

Step	Location
1. Plan to share parmlib definitions.	"Step 1. Plan to share parmlib members"
2. Determine where to specify system parameters for each system.	"Step 2. Determine where to specify system parameters" on page 42
3. Determine where to specify the system name.	"Step 3. Determine where to specify the system name" on page 44
4. Determine which parmlib members can be managed more easily through the use of system symbols.	"Step 2. Determine where to use system symbols in parmlib" on page 54
5. Create one or more IEASYMxx parmlib members.	"Step 4. Create an IEASYMxx parmlib member" on page 44
6. Code support for system symbols in the LOADxx parmlib member.	"Step 5. Code support for system symbols in LOADxx" on page 49

Step 1. Plan to share parmlib members

When planning to share parmlib members, ask yourself the following questions:

1. **Do two or more systems require identical resource names in the parmlib members to be shared?**

   If so, you need not specify system symbols in the parmlib members; the systems can share the contents of the parmlib members without the benefit of system symbols. However, if the systems require unique names for resources (such as jobnames) specified in those members, you must specify system symbols to generate unique values for those names.

   For example, if every system that shares parmlib uses the same set of APF authorized libraries, the systems can share the PROGxx parmlib member. Because the definitions in PROGxx are identical for each system, there is no need to specify system symbols in PROGxx.

2. **What resource definitions are good candidates for sharing?**

   If your goal is to greatly simplify your operating environment, the answer will be: As many as possible! If two or more systems require different names for a resource, define the resource with a system symbol in parmlib; then define different substitution texts for the system symbol on each system that is to share the parmlib definition. Now you have one convenient place to maintain

the definition. If you follow the same process with all parmlib definitions that require unique values, you can view a multisystem environment as a *single system image* with one point of control.

Be aware that there are also reasons you might *not* want to share certain resource definitions. Perhaps the release level of MVS prevents you from using system symbols on a particular system; or perhaps one or more systems do not require a particular resource. Whatever the case, your installation must examine the resources it defines in parmlib and determine the extent to which they can be shared.

An example of a resource definition that can be shared is a definition for a page data set. Because systems cannot share page data sets, installations must define unique page data sets for each system in a sysplex. You might want to assign unique names to the page data sets so you can easily identify them. Suppose that you had three systems, named SYS1, SYS2, and SYS3, that share the same IEASYSxx member. To specify unique names for the page data sets on the different systems, you could specify the following in the IEASYSxx parmlib member:

```
PAGE=PAGE.&SYSNAME..LOCAL1
```

When each system processes IEASYSxx, it calls into use a predefined page data set with the name that results from the symbolic substitution, as follows:

- PAGE.SYS1.LOCAL1 on system SYS1
- PAGE.SYS2.LOCAL1 on system SYS2
- PAGE.SYS3.LOCAL1 on system SYS3

3. **Do I need to share system parameters that specify unique values in the IEASYSxx parmlib member?**

   When the systems in your environment require unique parmlib members, you can specify system symbols in one or more shared IEASYSxx members to indicate unique member suffixes. Do this only when it is not possible to share the same members. In other words, try to specify system symbols in parmlib *definitions* first, as described in "Step 2. Determine where to use system symbols in parmlib" on page 54. If you cannot share a member with any degree of efficiency, create a new member with a unique suffix; then specify a system symbol in IEASYSxx that resolves to the suffix you want to use on each system.

   For example, suppose systems SYS1, SYS2, and SYS3 require unique CLOCKxx parmlib members. You can take the following steps:

   a. Create three unique members: CLOCKS1, CLOCKS2, and CLOCKS3.
   b. Specify the following in the shared IEASYSxx member:

      ```
      CLOCK=&SYSCLONE.
      ```

   If your installation accepts the default substitution text for &SYSCLONE (the last two characters in the system name), the CLOCK parameter in IEASYSxx will specify the following for each system:

   - CLOCKS1 on system SYS1
   - CLOCKS2 on system SYS2
   - CLOCKS3 on system SYS3

4. **Do I want to provide substitution texts for static system symbols, or do I want to accept the system defaults?**

   The system provides default substitution texts for the &SYSCLONE, &SYSNAME, and &SYSPLEX system symbols. See the descriptions of the default substitution texts in Table 4 on page 35. If the default texts are not acceptable to your installation, you can define your own substitution texts in parmlib. For example, if a system always runs on the same logical partition

(LPAR), the system might be able to accept the default for the &SYSNAME system symbol (because &SYSNAME can default to the LPAR name).

5. **Do I need to define additional static system symbols?**

   The static system symbols that are defined to the system might be sufficient to uniquely identify resources. If your installation requires additional static system symbols, you can define at least 800 additional static system symbols in the IEASYMxx parmlib member.

6. **Do I want to define the &SYSPLEX static system symbol early in system initialization so other parmlib members can use it?**

   You can optionally specify the sysplex name in both the COUPLExx and LOADxx parmlib members. The sysplex name is also the substitution text for the &SYSPLEX system symbol. When you specify the sysplex name in both LOADxx and COUPLExx, the substitution text for &SYSPLEX is defined early in system initialization, which allows all parmlib members to use the defined substitution text. When the sysplex name is defined only in COUPLExx, the system does not define the substitution text for &SYSPLEX until late in system initialization. The system substitutes the text **LOCAL** for any instances of &SYSPLEX that occur before COUPLExx is processed.

   If you plan to use the &SYSPLEX system symbol in parmlib, IBM suggests that you specify the *same* sysplex name in both LOADxx and COUPLExx. See "Step 5. Code support for system symbols in LOADxx" on page 49 for details.

7. **Do I want to ensure that the substitution text for &SYSCLONE is unique on all systems?**

   &SYSCLONE is a 1-2 character shorthand notation for the system name (unless your installation changes the substitution text to another value). IBM suggests that you use &SYSCLONE in places where the substitution text for &SYSNAME is too long, such as parmlib member suffixes.

   Each system in a sysplex must specify a unique SYSCLONE value. Message IXC217I will be issued if the substitution text for the symbol &SYSCLONE is not unique in a sysplex.

8. **Do I plan to use system symbols in other interfaces? If so, are there any considerations to make for those interfaces when I define system symbols in parmlib?**

   Parmlib is not the only place where you can use system symbols to share resources that require unique values. After you read this chapter, see the following books for information about sharing resources in the listed interfaces:

   Application or vendor programs
   See the topics on how to substitute text for system symbols in application programs in *z/OS MVS Programming: Assembler Services Guide*.

   Dynamic allocations
   See the topic on coding dsname allocation text units in *z/OS MVS Programming: Authorized Assembler Services Guide*.

   Job control language (JCL) for started tasks
   See the topic on how to specify system symbols in JCL in *z/OS MVS JCL Reference*.

   System commands
   See the topic on using system symbols in system commands in *z/OS MVS System Commands*.

   Time Sharing Option Extensions (TSO/E) Logon Procedures
   See the topic on how to set up logon processing in *z/OS TSO/E Customization*.

When defining static system symbols in parmlib, consider that the listed interfaces might want to use those system symbols.

Step 2. Determine where to specify system parameters

The IEASYSxx parmlib member specifies the system parameters that the system is to use. The operator can specify system parameters during initialization, or accept the system parameters specified in IEASYSxx. IEASYS00 is the default member; you can specify additional IEASYSxx members as needed.

You can specify the suffixes of IEASYSxx members that the system is to use in any of the following places:

- The LOADxx parmlib member
- The IEASYMxx parmlib member
- The SYSP parameter, specified in response to the SPECIFY SYSTEM PARAMETERS prompt.

When different systems in a multisystem environment require different IEASYSxx members, consider specifying the IEASYSxx suffixes in IEASYMxx. You can specify the IEASYSxx suffixes for all systems in one IEASYMxx member.

The specific advantages to using IEASYMxx over LOADxx are as follows:

- IEASYMxx can specify unique IEASYSxx members for as many systems as you need.
- When you use IEASYMxx, LOADxx can point to IEASYMxx; therefore, you do not have to code unique LOADxx and IEASYSxx members for each system.

For example, the SYSPARM statements in the IEASYMxx member in Figure 2 specify the IEASYSxx members to be used for three different systems:

```
SYSDEF  HWNAME(MVS3090) LPARNAME(SYSCA)
        SYSPARM(01,L)

SYSDEF  HWNAME(MVS3090) VMUSERID(VMSYSCB)
        SYSPARM(03,L)

SYSDEF  HWNAME(SYSCC)
        SYSPARM(01,02,L)
```

Figure 2. Example IEASYMxx parmlib member

See "Step 4. Create an IEASYMxx parmlib member" on page 44 for more information about how to specify system parameters in IEASYMxx.

Depending on where you specify the suffixes of IEASYSxx members that the system is to use, the system overrides or concatenates the members:

- If you specify the suffixes in IEASYMxx or LOADxx, the system will concatenate the members according to the scheme shown in Table 8 on page 43.
- If you specify the suffixes in IEASYMxx or LOADxx and the operator specifies the suffixes at IPL, the suffixes that are specified by the operator overrides suffixes in IEASYMxx or LOADxx.

The system always processes the IEASYS00 member first, regardless of where you specify IEASYSxx suffixes. If the same parameter appears in both IEASYS00 and a specified alternate IEASYSxx list, the value in the alternate list will override the

value in IEASYS00. Also, a parameter value in a later specified IEASYSxx list overrides the same parameter in an earlier specified list. See the description of the IEASYSxx member for more information.

For example, suppose you create the following IEASYSxx members and specify their suffixes in the indicated places:

- LOADxx - 01,02
- IEASYMxx - 03,04
- Console SYSP (WTOR) - 05,06

Table 8 shows how the system concatenates or overrides the suffixes:

Table 8. Precedence of system parameter specifications

LOADxx	IEASYMxx	Console SYSP (WTOR) (See note)	Resultant IEASYSxx noncatenation
None	None	None	00 (default)
(01,02)	None	None	(00,01,02)
(01,02)	None	Null response	(00)
(01,02)	(03,04)	None	(00,01,02,03,04)
(01,02)	(03,04)	(05,06)	(00,05,06)
None	(03,04)	None	(00,03,04)
None	(03,04)	(05,06)	(00,05,06)
None	None	(05,06)	(00,05,06)
Note: "None" means that no IEA101A prompt was issued. This is the case when the IMSI prevents the system from prompting the operator for system parameters. This is different from the case where the IEA101A prompt is issued, but the user response to the message is null. A null response is equivalent to a response of SYSP=00.			

If any of the IEASYSxx suffixes specified in LOADxx, IEASYMxx, or in response to a WTOR are not valid, the system issues message IEA336A to request that you specify system parameters again.

The number of IEASYSxx members that you use for a multisystem environment depends on the desired result from your system parameter concatenation and your preferences about how the information is organized. See Table 9 for more information.

Table 9. Recommended actions for IEASYSxx members

Desired result	Situation	Recommended action
One IEASYSxx member	Different systems require unique system parameters	Code one IEASYSxx member; specify system symbols to represent unique system parameters.
One IEASYSxx member	Different systems use the same system parameters	Code one IEASYSxx member to be shared by all systems.
Multiple IEASYSxx members	Different systems require unique system parameters	Code separate IEASYSxx members, for each system, where appropriate; specify the IEASYSxx members in a shared IEASYMxx member.

Step 3. Determine where to specify the system name

The name of a system is also the substitution text for the &SYSNAME system symbol. You can specify the name of a system in *one or more* of the following places:

- The IEASYMxx parmlib member
- The IEASYSxx parmlib member
- By the operator, in response to write to operator with reply (WTOR) message IEA101A SPECIFY SYSTEM PARAMETERS.

If you define the system name in only *one* of the listed places, the system will use that definition. Otherwise, the system will determine the name as follows:

- If the system name is specified in both IEASYMxx and IEASYSxx, the system will use the name in IEASYMxx.
- If the operator specifies the system name in response to the specify system parameters message, the system will use the name specified on the operator response.

IBM suggests that you specify the system name in IEASYMxx.

Table 10 shows every possible way to specify the system name and the resulting system action. In the table, IEASYMXX indicates that the system name is specified in the IEASYMxx parmlib member, IEASYSXX indicates that it is specified in the IEASYSxx parmlib member, and WTOR indicates that it is specified in response to the specify system parameters prompt:

Table 10. Precedence of system name specifications

IEASYMxx	IEASYSxx	Console SYSNAME (WTOR)	Resultant system name
IEASYMXX			IEASYMXX
IEASYMXX	IEASYSXX		IEASYMXX
IEASYMXX		WTOR	WTOR
IEASYMXX	IEASYSXX	WTOR	WTOR
	IEASYSXX		IEASYSXX
	IEASYSXX	WTOR	WTOR
		WTOR	WTOR

If the system name is not defined in any of the listed places, the system name will become, by default, one of the following:

- The processor name that is defined to HCD (if not in LPAR mode)
- The LPAR name that is defined to the processor (if in LPAR mode)
- A VM userid, if the system is running as a guest of VM/ESA.

Step 4. Create an IEASYMxx parmlib member

The main purpose of IEASYMxx is to provide a single place to specify system parameters for each system in a multisystem environment. IEASYMxx contains statements that take the following steps:

- Define static system symbols
- Specify IEASYSxx parmlib members that contain system parameters.

You can apply the statements in IEASYMxx to any system in your environment. Therefore, only one IEASYMxx member is required to define static system symbols and specify system parameters for all systems.

In IEASYMxx, you can define at least 800 additional static system symbols for each system in a multisystem environment. In other words, you can define as many additional static system symbols in IEASYMxx as you like, so long as the resulting symbol table would not exceed the maximum symbol table size of 32512 bytes.

The LOADxx parmlib member specifies the IEASYMxx member that the system is to use. For information about how to specify the suffix of the IEASYMxx member in LOADxx, see "Step 5. Code support for system symbols in LOADxx" on page 49.

Contents of IEASYMxx

IEASYMxx contains SYSDEF statements that *optionally* take the following steps for each system:

- Define substitution texts for the existing &SYSCLONE and &SYSNAME system symbols, and apply those definitions to one or more systems.
- Define additional static system symbols and apply those definitions to one or more systems (at least 800 additional static system symbols are allowed across all of your IEASYMxx members).
- Control which definitions apply to which systems by specifying values for comparison to the current system hardware name or logical partition (LPAR) name.

See the description of IEASYMxx for descriptions of the parameters that each SYSDEF statement can contain.

Scope of statements in IEASYMxx

You can code the statements in IEASYMxx so only one IEASYMxx member is needed to define static system symbols and system parameters for all systems in a multisystem environment. If you want, you can code additional IEASYMxx members to organize the definitions more clearly.

When coding IEASYMxx, remember the following rules about the scope of the statements:

- The system processes IEASYMxx members from left to right, in the order which they are specified in the LOADxx parmlib member. Definitions that are specified in previous IEASYMxx members remain in effect until they are overridden by subsequent definitions. For example, suppose you specify the following IEASYMxx members in LOADxx:

  ```
  IEASYM   (01,02,L)
  ```

  The system first processes the IEASYM01 parmlib member. Then the system processes IEASYM02. The definitions from IEASYM01 remain in effect unless definitions from IEASYM02 override them.
- At the beginning of IEASYMxx, you can specify **global** statements that apply to all systems that use the IEASYMxx member. Global statements should specify the values of SYSPARM and SYMDEF that are most common to the systems in your sysplex. Global statements do not use the HWNAME, LPARNAME, and VMUSERID parameters to identify specific systems to which definitions apply.

  When specific systems require values that are different from those on global statements, you can override the global statements with *local* statements (see below).

- **Local** statements apply to a subset of systems in your sysplex. Using the HWNAME, LPARNAME, and VMUSERID parameters, local statements identify one or more CPCs or LPARs that are to use specific system symbols and system parameters. The HWNAME, LPARNAME, and VMUSERID values specified on a SYSDEF statement are compared to those of the current system. If they match exactly, the other parameters on the SYSDEF statement will be used. If not, the other parameters will be ignored for this system.
- The system processes IEASYMxx members in the order which they are specified. When two definitions for the same resource appear in IEASYMxx, the most recent definition overrides the previous definition. Keep this rule in mind when specifying global and local statements: If global statements appear *after* local statements in IEASYMxx, the local statements will not override the global statements.

Procedure for coding IEASYMxx: IBM suggests that you code one IEASYMxx member that contains only global statements. Specify that member as the first member in the IEASYMxx concatenation in LOADxx (for example, IEASYM01 in the example above). Then code other IEASYMxx members that specify local statements, which override the global statements specified in the first IEASYMxx member. This practice ensures that global and local statements always appear in the proper order.

Rules for coding IEASYMxx

Follow these rules when coding IEASYMxx:

1. Define new system symbols that are 1- through 8 characters long, *excluding* the required ampersand and the optional period. For example, you can define a system symbol called &PAGESYM3, which contains an 8-character name, using the following SYMDEF statement:

   ```
   SYMDEF(&PAGESYM3='LOCAL3')
   ```

2. Do not define *resolved* substitution texts that are longer than system symbol names (*including* the required ampersand and *excluding* the optional period). Before being resolved, a substitution text can contain other system symbols that extend its length beyond the length of the symbol name. However, the substitution text for *those* symbols must resolve to form a string that is less than the length of the symbol name.

   For example, consider the &PAGESYM3 system symbol definition:

   ```
   SYMDEF(&PAGESYM3='LOCAL3')
   ```

   The LOCAL3 substitution text above is valid because it contains six characters, which is less than the 9 characters in the symbol name, &PAGESYM3 (note that the optional period is not included). The following definition is also valid:

   ```
   SYSNAME(SYS1)
   SYMDEF(&PAGESYM3='LOC&SYSNAME.3')
   ```

   Note that although the LOC&SYSNAME3 substitution text appears longer than the &PAGESYM3 symbol, the &SYSNAME system symbol in LOC&SYSNAME3 resolves to a character string that makes the final resolved substitution text, LOCSYS13, shorter than the symbol name.

   If you specify a substitution text that, when resolved, is longer than a symbol name, the system will prompt for a valid substitution text.

   Recommendation: Define system symbols that are eight characters long (the maximum) so you can define substitution texts that are up to nine characters long (the eight characters in the system symbol name plus the ampersand at the beginning of the name).

3. Do not define new system symbols that begin with the characters **SYS**. Those names are reserved for existing system symbols (like &SYSNAME).

4. You can specify an optional period at the end of a system symbol definition. For example, both of the following statements are valid and define the same system symbol:

   ```
   SYMDEF(&PAGESYM1='LOCAL1')
   SYMDEF(&PAGESYM1.='LOCAL1')
   ```

5. When coding a single quotation mark as part of a substitution text for an installation-defined static system symbol, specify two consecutive single quotation marks. In the following example, &SYMBOL4 is assigned the string **O'HARE**:

   ```
   SYMDEF(&SYMBOL4='O''HARE')    /* &SYMBOL4 is assigned O'HARE  */
   ```

6. If you intend to use a system symbol to represent a parmlib member suffix in IEASYSxx, see the description of IEASYSxx in this book for special considerations.

If the system finds an error in IEASYMxx, the system will issue message IEA013E. If a statement in error applies to the processor or LPAR on which this system is being initialized, the system will then issue message IEA011A to prompt for a new IEASYMxx parmlib member.

Example of coding IEASYMxx: Use the following example as a model when coding IEASYMxx for your installation. The example explains how to code IEASYMxx for the sysplex shown in Figure 4 on page 48.

1. Enter a **global** SYSDEF statement that specifies:
   - The IEASYSxx member, IEASYS01, that specifies system parameters for systems SYSCA, SYSCB, and SYSCD. SYSCC requires two IEASYSxx members, which are specified on a local SYSDEF statement later in the parmlib member.
   - The global definition for the installation-defined system symbol &LNKSYM;

     ```
     SYSDEF  SYSPARM(01,L)
             SYMDEF(&LNKSYM='GLOBALP')
     ```

2. Enter a **local** SYSDEF statement that identifies the LPAR on which system SYSCA is running. The statement specifies:
   - The hardware name and LPAR name (the system name defaults to the LPAR name)
   - A local definition for the installation-defined system symbol &LNKSYM, which is to override the global definition specified earlier. The local definition for &LNKSYM will be used only by system SYSCA.

     ```
     SYSDEF  HWNAME(MVS3090) LPARNAME(SYSCA)
             SYMDEF(&LNKSYM='LOCALP1')
     ```

3. Enter a **local** SYSDEF statement that identifies the CPC on which system SYSCC is running. The statement specifies:
   - The hardware name for the CPC (the system name defaults to the hardware name)
   - The IEASYSxx members that contain the system parameters for system SYSCC. The IEASYSxx members will be used only by system SYSCC. The other systems will use the IEASYSxx members identified in the global definition.

     ```
     SYSDEF  HWNAME(SYSCC)
             SYSPARM(01,02,L)
     ```

In this example, no local SYSDEF statements are necessary for systems SYSCB or SYSCD because they do not require unique definitions for system symbols or system parameters.

The complete IEASYMxx parmlib member is:

```
SYSDEF  SYSPARM(01,L)
        SYMDEF(&LNKSYM='GLOBALP')

SYSDEF  HWNAME(MVS3090) LPARNAME(SYSCA)
        SYMDEF(&LNKSYM='LOCALP1')

SYSDEF  HWNAME(SYSCC)
        SYSPARM(01,02,L)
```

Figure 3. Complete IEASYMxx Parmlib Member

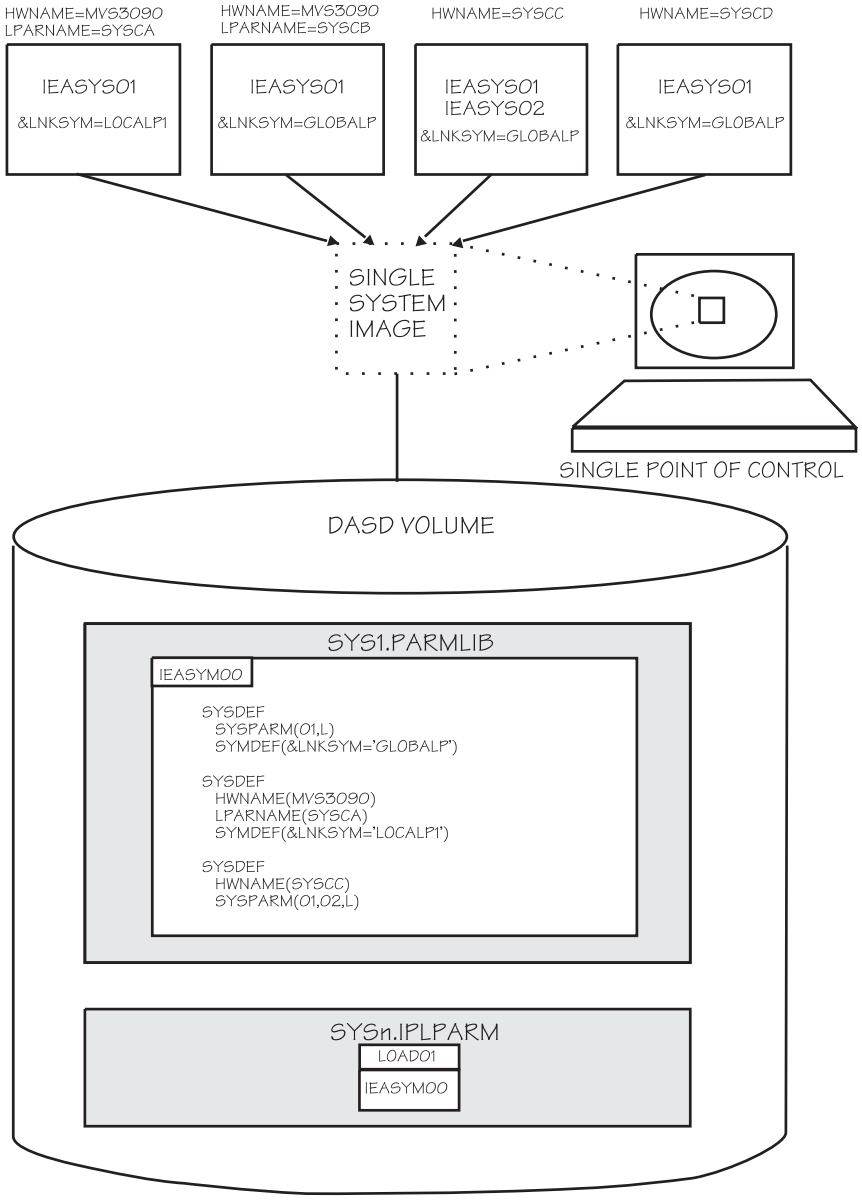

Figure 4. Coding IEASYMxx for a Four-System Sysplex

Figure 5 on page 49 shows an example of how to use one IEASYMxx member to specify global statements (IEASYMA4) and another to specify local statements (IEASYMA5). IEASYMA4 defines system symbols for all systems that share the

member:

```
SYSDEF SYMDEF(&LNKLST1='CS,52')
       SYMDEF(&LNKLST2='44,42,L')
       SYMDEF(&LPALST1='52,CS')
       SYMDEF(&LPALST2='ST,L')
       SYMDEF(&MLPALS1='44')
       SYMDEF(&MLPALS2='43,42')
       SYMDEF(&SVCNAME='44')
       SYMDEF(&MVSRLSE='52')
       SYMDEF(&SMFPARM='44')
```

Figure 5. IEASYMA4 Parmlib Member

IEASYMA5 (shown in Figure 6) defines local system symbols and system parameters for processors J50 and J60.

```
SYSDEF HWNAME(J50)
       SYSPARM(02)
       SYSNAME(J50)
       SYSCLONE(J5)
       SYMDEF(&NODE='55')
       SYMDEF(&LOGCLS='M')
SYSDEF HWNAME(J60)
       SYSPARM(02)
       SYSNAME(J60)
       SYSCLONE(J6)
       SYMDEF(&NODE='56')
       SYMDEF(&LOGCLS='O')
```

Figure 6. Example IEASYMxx Parmlib Member

See Figure 7 on page 50 for a sample LOADxx member that points to IEASYMA4 and IEASYMA5.

Step 5. Code support for system symbols in LOADxx

The LOADxx parmlib member provides sysplex-related definitions early in system initialization so other parmlib members can use those definitions. It optionally specifies the IEASYMxx parmlib member that the system is to use.

If you use IEASYMxx to define names for the systems in your environment, you will not be required to code a unique LOADxx member for each system.

Contents of LOADxx

LOADxx specifies the following parameters that relate to replication:

Parameter	Description
IEASYM	Specifies the suffixes of one or more IEASYMxx members that define system symbols and specify system parameters.
SYSPARM	Specifies the suffixes of one or more IEASYSxx parmlib members that contain system parameters
SYSPLEX	Specifies the name of the sysplex (which is the substitution text for the &SYSPLEX system symbol)

Procedure for coding LOADxx

Take the following steps to code the parameters in LOADxx that relate to the sharing of system resources:

1. Optionally specify the sysplex name on the SYSPLEX parameter in the LOADxx parmlib member. For example, to name a sysplex EXMPLEX, code:

   ```
   SYSPLEX  EXMPLEX
   ```

   Note: LOADxx defines the substitution text for &SYSPLEX early in system initialization so other parmlib members can use it. Therefore, if you plan to use the &SYSPLEX system symbol in parmlib, specify the sysplex name in LOADxx. You can also specify the sysplex name on the SYSPLEX parameter in the COUPLExx parmlib member. To ensure that the name in COUPLExx matches the one in LOADxx, IBM suggests that you specify the following in COUPLExx:

   ```
   SYSPLEX(&SYSPLEX.)
   ```

   If you do not specify the SYSPLEX parameter in LOADxx, the system will temporarily assign the value LOCAL to the &SYSPLEX system symbol. When the system processes the SYSPLEX parameter in COUPLExx, it assigns the value specified on that parameter to &SYSPLEX. If the value is SYSPLEX(&SYSPLEX;), the value of &SYSPLEX will remain LOCAL.
2. On the IEASYM statement, specify the suffixes of the IEASYMxx members that the system is to use.
3. Determine if you need to specify one or more IEASYSxx suffixes on the SYSPARM parameter:
   - If the systems in your environment require unique system parameters, specify the system parameters in IEASYMxx (see "Step 4. Create an IEASYMxx parmlib member" on page 44); do not respecify the system parameters in LOADxx.
   - If the systems in your environment use the same system parameters, they can share the same LOADxx member; specify the system parameters in LOADxx.

   If you specify system parameters in *both* LOADxx and IEASYMxx, the system will use the scheme shown in Table 10 on page 44 to determine which IEASYSxx members to use.

Figure 7 is an example of a LOADxx member that includes support for system symbols. The IEASYM statement specifies that IEASYMxx members IEASYMA4 and IEASYMA5 are to be used. The SYSPLEX statement specifies a sysplex name of UTCPLXJ4.

```
IODF     A1 IODFST   B710     A1
IEASYM   (A4,A5)
SYSPLEX  UTCPLXJ4
SYSCAT   CMNJ4C113CCATALOG.J40CAT
```

Figure 7. Example LOADxx parmlib member

Using system symbols in parmlib

After you set up parmlib for sharing, take the following steps to specify system symbols in parmlib definitions:

1. "Step 1. Know the rules for using system symbols in parmlib" on page 51
2. "Step 2. Determine where to use system symbols in parmlib" on page 54
3. "Step 3. Verify system symbols in parmlib" on page 55

Step 1. Know the rules for using system symbols in parmlib

Follow these rules and recommendations when using system symbols in parmlib:

1. Specify system symbols that:
   - Begin with an ampersand (&)
   - Optionally end with a period (.)
   - Contain 1-8 characters between the ampersand and the period (or the next character, if you do not specify a period).

   Note: Symbols are character strings and can only be used as such. Even though a character string can consist of only numeric characters, it cannot be used as a number. For example, a symbol cannot be used for the `start` or `number` value in a substring.

   If the system finds a system symbol that does not end with a period, it will substitute text for the system symbol when the next character is one of the following:
   - Null (the end of the text is reached)
   - A character that is not alphabetic, numeric, or special (@,#, or $).

   Recommendation: End all system symbols with a period. Omitting the period that ends a system symbol could produce unwanted results under certain circumstances. For example, if the character string `(2)` follows a system symbol that does not have an ending period, the system will process the (2) as substring syntax for the system symbol, regardless of how you intended to use the string in the command. For more information about substringing system symbols, see "Using substrings of system symbols" on page 52.
2. Use a small set of system symbols so they are easy to manage and remember.
3. Code two consecutive periods (..) if a period follows a system symbol. For example, code &DEPT..POK when the desired value is D58.POK and the substitution text D58 is defined to the system symbol &DEPT.
4. When using system symbols in data set name qualifiers, keep the rules for data set naming in mind. For example, if you use &SYSNAME as a data set qualifier, ensure that the substitution text begins with an alphabetic character.
5. Ensure that resolved substitution texts do not extend parameter values beyond their maximum lengths. For example, suppose that the following command is to start CICS:

   ```
   S CICS,JOBNAME=CICS&SYSNAME.,...
   ```

   The resolved substitution text for &SYSNAME cannot exceed four characters because job names are limited to eight characters (the four characters in CICS plus up to four character in &SYSNAME). A substitution text of SYS1 is valid because it resolves to the job name CICSSYS1. However, a substitution text of SYSTEM2 is not valid because it resolves to the job name of CICSSYSTEM2, which exceeds the allowable maximum of eight characters.
6. If you use &SYSNAME, ensure that its substitution text is unique on each system. See "Step 3. Determine where to specify the system name" on page 44 for more information.
7. Do not specify system symbols in the values on the OPI and SYSP parameters in the IEASYSxx parmlib member.
8. Do not specify any system symbols in parmlib members that do not support system symbol substitution. See "Overview of parmlib members" on page 16 for information about the parmlib members that support system symbols.

Using substrings of system symbols

Substringing allows you to specify a subset of characters in a substitution text. This function is particularly useful when specifying a symbol in a field that accepts only a small number of characters.

The syntax for substringing symbols is:
-or-

```
&SYMBOL(start:number).
```

```
&SYMBOL(start).
```

start The character position where the substring is to start. The first character in the original system symbol is position 1, the second is position 2, and so on. If *start* is positive, the system will count from the starting position to the end of the string. If *start* is negative (in other words, a minus sign appears before it), the system will count backwards from the ending position of the string.

number
The number of characters, from the starting position to the ending position of the string, that the substring is to contain. If *number* is not specified, the substring length will default to 1. Do not specify a negative number for *number*.

For example, assume that you want to specify &SYSNAME as the second-level qualifier in a data set name, and you want to limit the qualifier to two characters. If the system defined a four-character value to &SYSNAME, such as SYS1, you can specify the third and fourth characters of SYS1 using the following notation:

```
HILEVELQ.&SYSNAME(3:2)..LOWLVLQ
```

In this case, the system substitutes the third and fourth characters of SYS1 (which are S1) as the second-level qualifier, yielding:

```
HILEVELQ.S1.LOWLVLQ
```

Procedure for substringing symbols: Follow these steps when specifying substrings for symbols:

1. Start the substring notation with a left parenthesis and end it with a right parenthesis.

   If you specify only the left and right parentheses, the "substring" is the entire substitution text. For example, if the system symbol &YR8; is assigned the substitution text 2008 in all cases below, the following system symbols will specify the same substitution text:

Symbol	Resulting Substring
&YR8	2008
&YR8()	2008

2. Specify the character position, from the *start* of the original string, where the substring is to begin. For example:

Symbol	Resulting Substring
&YR8(4)	8

Symbol	Resulting Substring
&YR8(2)	0

Use a minus sign (-) before the start position to indicate that the system is to count backwards from the *end* of the substitution text. For example:

Symbol	Resulting Substring
&YR8(-4)	2
&YR8(-1)	8

3. Specify the number of characters, from the starting position, that the substring is to contain. Separate this number from the start position with a colon (:). For example:

Symbol	Resulting Substring
&YR4(3:2)	08
&YR4(-4:2)	20

Errors in substringing: It is important that you specify substrings carefully. The syntax can become complicated, and it is very easy to make mistakes. When the system finds an error in substring notation, it tries to assign a value whenever possible. For example, when a substring length of zero is specified, the system assigns a length of one.

Table 11 summarizes the errors that might occur in substring syntax. Always, assume that the dynamic system symbol &YR8. is assigned the substitution text **2008**.

Table 11. Summary of errors in substring syntax

System Symbol	Description of Error	System Action	Resulting Substring
&YR8(-5:1)	The start position is not valid (it exceeds the length of the substitution text).	The system assigns start position one.	2
&YR8(0:1)	The start position is zero.	The system assigns start position one.	2
&YR8(4:0)	The length is zero.	The system assigns a length of one.	8
&YR8(5:1)	The start position is beyond the length of the substitution text.	The system assigns a length of zero.	Null
&YR8(3:3)	The length exceeds the length of the substitution text beyond the specified start position.	The system assigns the substring from the start position to the end of the substitution text.	08
&YR8(-2:a)	The “a” character is not valid substring notation.	The system treats the substring notation as normal text.	2008(-2:a)
&YR8.(3:3)	The optional period indicates the end of the string. The system does not apply the substring notation to the string.	The system treats the substring notation as normal text.	2008(3:3)

Using double ampersand notation: You can use double ampersand notation to:

- Defer the processing of a symbol until a later time
- Tell the system to process an ampersand as a literal character (instead of a character that indicates the beginning of a symbol).

When the system finds two consecutive ampersands at the beginning of a valid symbol, the first ampersand is removed and the second is kept in place. A subsequent process can then substitute text for the symbol in later processing, or the second ampersand can remain as a literal character.

Note: Defer the processing of system symbols only if a function specifically states that it supports double ampersand notation (for example, the naming of dump data sets with the DUMPDS command, which is described in *z/OS MVS System Commands*). If a function does not specifically state that it supports double ampersand notation, the system might not perform the desired substitution.

When you code:

```
&&DAY.
```

The system removes the first ampersand:

```
&DAY.
```

Step 2. Determine where to use system symbols in parmlib

System symbols offer the greatest advantage when two or more systems require different data sets, jobs, procedures, or entire parmlib members. This section provides examples of how to specify system symbols when naming certain resources in parmlib.

Data sets: A good example of using system symbols in data set names is the DSNAME parameter in the SMFPRMxx parmlib member, which specifies data sets to be used for SMF recording. Assume that each system in your sysplex requires one unique data set for SMF recording. If all systems in the sysplex use the same SMFPRMxx parmlib member, you can specify the following naming pattern to create different SMF recording data sets on each system:

```
SY&SYSCLONE..SMF.DATA
```

When you IPL each system in the sysplex, the &SYSCLONE system symbol resolves to the substitution text that is defined on the current system. For example, if a sysplex consists of two systems named SYS1 and SYS2, accepting the default value for &SYSCLONE will produce the following data sets:

```
SYS1.SMF.DATA on system SYS1
SYS2.SMF.DATA on system SYS2
```

Note that the use of &SYSCLONE provides unique data set names while establishing a consistent naming convention for SMF recording data sets.

Parmlib members: You can apply the same logic to system images that require different parmlib members. For example, assume that system images SYS1 and SYS2 require different CLOCKxx parmlib members. If both systems share the same IEASYSxx parmlib member, you can specify &SYSCLONE in the value on the CLOCK parameter:

```
CLOCK=&SYSCLONE;
```

When each system in the sysplex initializes with the same IEASYSxx member, &SYSCLONE resolves to the substitution text that is defined on each system. Accepting the default value for &SYSCLONE produces the following:

```
CLOCK=S1    (Specifies CLOCKS1 on system SYS1)
CLOCK=S2    (Specifies CLOCKS2 on system SYS2)
```

Started Task JCL: If JCL is for a started task, you can specify system symbols in the source JCL or in the START command for the task. You cannot specify system symbols in JCL for batch jobs, so you might want to change those jobs to run as started tasks.

If a started task is to have multiple instances, determine if you want the started task to have a different name for each instance. Started tasks that can be restarted at later times are good candidates. The different names allow you to easily identify and restart only those instances that require a restart. For example, you might assign different names to instances of CICS because those instances might be restarted at later points in time. However, instances of VTAM, which are generally not restarted, might have the same name on different systems.

When you start a task in the COMMNDxx parmlib member, you can specify system symbols as part of the job name. Assume that system images SYS1 and SYS2 both need to start customer information control system (CICS). If both system images share the same COMMNDxx parmlib member, you can specify the &SYSNAME system symbol on a START command in COMMNDxx to start unique instances of CICS:

```
S CICS,JOBNAME=CICS&SYSNAME;,...
```

When each system in the sysplex initializes with the same COMMNDxx member, &SYSNAME resolves to the substitution text that is defined on each system. If &SYSNAME is defined to SYS1 and SYS2 on the respective systems, the systems will start CICS with the following jobnames:

```
CICSSYS1 on system SYS1
CICSSYS2 on system SYS2
```

Note that the *resolved* substitution text for &SYSNAME is eight characters long, which is the maximum length for jobnames.

Step 3. Verify system symbols in parmlib

IBM provides you with tools to verify symbol usage in parmlib. The parmlib symbolic preprocessor tool allows you to test symbol definitions before you IPL the system to use them. This tool shows how a parmlib member will appear after the system performs symbolic substitution. For information about setting up and using the parmlib symbolic preprocessor tool, see Appendix B, “Symbolic Parmlib Parser,” on page 795.

If you only need to verify a new parmlib member's use of the current system symbols, you can run the IEASYMCK sample program to see how the contents of the parmlib member will appear after symbolic substitution occurs.

IEASYMCK is located in SYS1.SAMPLIB. See the program prolog for details.

Changing System Symbols

System symbols can be changed, added or deleted by the SETLOAD xx,IEASYM command (see *z/OS MVS System Commands*. for full details) and can be changed or added by the IEASYMU2 program. Updated symbols may not be transported to other systems and therefor may not be usable, for example, for job JCL interpretation or automatic restart manager (ARM) processing. Update symbols with care. If you are not careful in how you update symbols, 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. One safe use of updated symbols is for symbols that are used only in catalog aliases.

When you update symbols, the entire system symbol table is re-built, and the old system symbol table remains in storage; this storage is never freed, for system integrity reasons. As a result, you can avoid wasting common storage if you do a group of updates together instead of doing individual updates. The system symbol table is in subpool 245 (ESQA) and may be as large as 32512 bytes. The size depends upon the number of symbols that you have defined. In general, the number of bytes in the table can be calculated as the sum of:

- 4
- Number_Of_Symbols multiplied by 16
- Length of every symbol name (including the leading ampersand and trailing period)
- Length of the value for every symbol

The size of the current system symbol table can be viewed by the DISPLAY SYMBOLS,SUMMARY command. An example of the command output is:

```
SY1  IEA994I STATIC SYSTEM SYMBOL INFO
SYMBOLS DEFINED: 12
CURRENT TABLE SIZE: 344 BYTES
MAX TABLE SIZE: 32512 BYTES
```

Also, z/OS V2R1 provides CHECK(IBMSUP,SUP_SYMBOL_TABLE_SIZE) as part of the IBM Health Checker for z/OS to let you view the current size and to let you set a threshold so that you will be alerted if/when the current size approaches the maximum size too closely.

The IEASYMU2 program is provided in SYS1.LINKLIB in z/OS V2R1 as a replacement for the IEASYMUP program, which had been provided only by OBJ in SYS1.SAMPLIB and with documentation only in the IBM redbook SG24-5451. As of z/OS V2R1, IEASYMUP must not be used. If the IEASYMUP OBJ provided in SYS1.SAMPLIB is re-linked and then executed, it will indicate that you must use either the new SETLOAD xx,IEASYM command or the IEASYMU2 program. Do not use a pre-z/OS V2R1 IEASYMUP on a z/OS V2R1 system.

IBM recommends that you not use both the IEASYMU2 program and the SETLOAD xx,IEASYM command unless you do so intentionally and carefully. Updates made by the IEASYMU2 program are temporary and will be lost if you subsequently issue the SETLOAD xx,IEASYM command unless you have made corresponding updates in the appropriate IEASYMxx parmlib members. You could use RACF profiles to enforce that either IEASYMU2 or SETLOAD xx,IEASYM not be used. The SETLOAD xx,IEASYM command is protected by the

MVS.SETLOAD.IEASYM entity in the OPERCMDS class; the IEASYMU2 program is protected by IEASYMUP.symbolname entities in the FACILITY class.

To execute the IEASYMU2 program, you would run the following JCL (noting that IEASYMU2 will be in the LNKLST so no joblib or steplib should be used):

```
//IEASYMU2 JOB ...
//IEASYMU2 EXEC PGM=IEASYMU2,PARM='...'
```

The input parameter is specified using the PARM keyword on the EXEC card, in the format:

```
PARM='SYMNAME1=VALUE1 SYMNAME2=VALUE2 ...SYMNAMEN=VALUEN'
```

where:

- All updates by IEASYMU2 are controlled by RACF facility class entity IEASYMUP.symbolname (without the leading ampersand and trailing period). If the security product does not grant UPDATE access to the requestor for every symbol requested by the job, the job does not complete successfully and the symbol table is not updated.
- The name of each "SymnameN" should not contain the leading "&" or the trailing ".".
- The first symname=value specification must begin at the first character of the parameter. Each symname=value specification must be separated from the preceding one by exactly one blank.
- There must be no trailing blanks at the end of the parm string.
- The value must not have blanks.
- No translation to uppercase is done by the processing, so be careful to put everything in uppercase as needed.
- You can change the values of symbols (including the length).
- You can add new symbols.
- You cannot have an "&" as part of the symbol name
- You can use symbolics in defining the value if you invoke this as a started task or started job. In this case, the symbols are resolved by the system before IEASYMU2 runs.
- When there are symbolics used in defining the value that are not resolved by the system before IEASYMU2 runs (perhaps because you ran this as a batch job):
  - They must be resolvable by the system symbol table that existed before this running of IEASYMU2.
  - The input parameter area may be modified by the IEASYMU2 processing.
- The specified symbol name must be no longer than 8 characters (as it does not contain the leading "&" or the trailing ".").
- You may not specify the system-defined symbols SYSPLEX, SYSNAME, SYSCLONE, SYSR1, SYSALVL, SYSOSLVL.
- The specified value must be no longer than the specified symbol name plus one.
- Less syntax checking of operands may be done for the IEASYMU2 program than is done for the SETLOAD xx,IEASYM command.
- If the parameter string exceeds 100 bytes (or even if it does not), you can use the PARMDD (also known as LONGPARM) support of z/OS V2R1 to specify the parameters within the DD identified by using PARMDD=ddname on the EXEC PGM= statement of the JCL. Care must be taken if using PARMDD, due to the PARMDD rule of stripping off all trailing blanks on a given line. Thus:
  - The first symname=value pair must start in column 1 of the first line

  - You should not split a symname=value pair across lines (primarily for clarity)
  - When continuing onto a second or subsequent line, begin that line with the symname=value pairs starting in column 2
  - For example, (the first line of the example is not part of your input; it is just used for showing the starting column)

```
1--------------------------------
SYMNAME1=VALUE1 SYMNAME2=VALUE2
 SYMNAME3=VALUE3
 SYMNAME4=VALUE4 SYMNAME5=VALUE5
```

- You may not provide more than 255 input symbols
- The resulting system symbol table must not exceed 32512 bytes.
- Output messages are written by WTO with routing code 11 and MCSFlag indicating hardcopy-only.
- As with the SETLOAD xx,IEASYM command, a change to the system symbol table results in sending ENF signal 73 and also re-running CHECK(IBMSUP,SUP_SYMBOL_TABLE_SIZE).
- Messages IEASYMU01I, IEASYMU02I, IEASYMU03I, and IEASYMU04I might be issued.

IEASYMU2 return codes

In the return code descriptions in Table 12, "*xxxxxx*" identifies which input symbol is the source of the error ("000001" corresponds to the first input symbol, and so on).

Table 12. IEASYMU2 return codes

Return code	Explanation
0	Success
00000008	Missing parameter. No parameters were provided. Correct the input.
xxxxxx0C	Symbol name contains ampersand The symbol name has an ampersand; this is not allowed. Correct the input.
xxxxxx10	Bad symbol name length A symbol name length is not in the range 1-8. Correct the input.
xxxxxx14	Bad symbol value length A symbol value length is longer than the symbol name length plus one. Correct the input.
00010018	Too many symbols More than 255 input symbols were specified. Correct the input.
xxxxxx1C	Reserved symbol An attempt to define or change a reserved symbol (one of the symbols defined by z/OS - SYSPLEX, SYSNAME, SYSCLONE, SYSR1, SYSALVL, SYSOSLVL). Correct the input.
xxxxxx20	RACF denied access RACF denied access to this symbol. Correct the input or contact the security administrator to gain access.

Table 12. IEASYMU2 return codes (continued)

Return code	Explanation
xxxxxx24	Symbol value contains ampersand after substitution The symbol value still has an ampersand after all symbol substitution has been done; this is not allowed. Correct the input.
xxxxxx28	Too much symbol data The resulting symbol table would exceed the maximum symbol table size. Correct the input.
00000FFF	Improper release The IEASYMU2 program may not be used on this release of z/OS. Use IEASYMU2 on z/OS V2R1 or later.

Displaying static system symbols

You can enter the DISPLAY SYMBOLS operator command to display the static system symbols and associated substitution texts that are in effect for a system. See *z/OS MVS System Commands* for information about how to enter DISPLAY SYMBOLS.

Diagnosing problems with static system symbols

You can use the SYMDEF interactive problem control system (IPCS) subcommand to diagnose problems in static system symbol definitions. See *z/OS MVS IPCS Commands* for information about how to use the SYMDEF subcommand.

Indirect volume serial support

Indirect volume serial support allows the system to dynamically resolve volume and device type information for non-VSAM data sets that reside on either the system residence volume (SYSRES) or one or more logical extensions to the SYSRES volume. If all the SYSRES data sets do not fit on a single DASD volume, you can use additional volumes as "logical extensions" to SYSRES and refer to them indirectly. This allows you to change the volume serial number or device type of SYSRES or its logical extension volumes without having to recatalog the non-VSAM data sets on that volume.

The indirect volume serial is specified by one of two methods, either six asterisks (******) or a system symbol. The method used depends on whether the data set will be cataloged on the SYSRES volume or on one of the SYSRES logical extension volumes.

- For data sets that will reside on the SYSRES volume, you can specify either:
  - a string of six asterisks (******) in place of the volume serial, or
  - the &SYSR1 static symbol in place of the volume serial. (See “Restrictions” on page 61.)

  The system dynamically resolves ****** or &SYSR1 to the volume serial of the volume from which the system was IPLed (the SYSRES volume).
- For data sets that will reside on the logical extensions to SYSRES, you must specify a static system symbol in place of the volume serial. This static symbol must be defined in the IEASYMxx parmlib member.

Note: If you establish a naming convention for your SYSRES volume and associated logical extensions, you can define a system symbol in IEASYMxx that allows you to derive the logical extensions from whatever SYSRES volume is IPLed. This eliminates the need to update the IEASYMxx parmlib member if your installation IPLs with different SYSRES volumes. A naming convention example is shown in the following section.

Using indirect volume serial support

The following information is a guide to help you use indirect volume serial support:

- Use access method services (IDCAMS) DEFINE NONVSAM to catalog the data sets that will reside on SYSRES or its logical extension volumes.

  For a data set that will be cataloged on the SYSRES volume, specify ****** or the &SYSR1 static symbol in the VOLUMES parameter and a value of 0000 on the DEVICETYPES parameter.

  For a data set that will be cataloged on a SYSRES logical extension volume, specify a static system symbol in the VOLUMES parameter. IBM suggests using &SYSR2 for the first logical extension volume, &SYSR3 for the second, and so on. If you decide to use symbols with names different than &SYSR2, &SYSR3, and so on, keep in mind that the symbol name must be no smaller than the volume serial it represents, and no longer than six characters. This length includes the & but not the period.

- Define static symbols &SYSR2, &SYSR3, and so on, in the IEASYMxx parmlib member. (The system sets &SYSR1 to the volume serial of the IPL volume. You cannot define &SYSR1.)

  For example, if the logical extension for the SYSRES volume used to IPL your production system is P1RES2 and the logical extension for the SYSRES volume used to IPL your test system is TSTRS2, your IEASYMxx member might contain statements such as:

```
/* Symbols for production system, P1 */
SYSDEF .
       .
       .
       SYMDEF(&SYSR2='P1RES2')   /* Logical extension volume for P1 */
/* Symbols for test system, TST      */
SYSDEF .
       .
       .
       SYMDEF(&SYSR2='TSTRS2')   /* Logical extension volume for TST */
```

  If your installation IPLs with different SYSRES volumes and you establish a naming convention for the SYSRES and its logical extension volumes, you can create a single IEASYSMxx member that can be used regardless of which SYSRES volume is used to IPL the system. To do this, use substrings of the SYSRES volume serial (&SYSR1) in defining the symbols for the extension volume serials. For example, if you have the following SYSRES volumes and logical extensions:

SYSRES	**Extensions**
S01RES	S01RS2,S01RS3
S02RES	S02RS2,S02RS3
DEVRES	DEVRS2,DEVRS3

  you can refer to them using a single IEASYMxx parmlib member with the following statements:

```
SYSDEF .
       .
       SYMDEF(&SYSR2='&SYSR1(1:3).RS2') /* second SYSRES logical
                                           extension */
       SYMDEF(&SYSR3='&SYSR1(1:3).RS3') /* third SYSRES logical
                                          extension  */
       .
       .
```

- If you use parmlib member PROGxx to specify authorized libraries, and any of those libraries are on SYSRES and its logical extensions, you can use static system symbols to identify the volumes. If, however, you still use parmlib member IEAAPFxx for this purpose, symbols will **not** work.

  Note: You can use the CSVLNKPR tool to convert IEAAPFxx to PROGxx.
- In RACF, you may be able to remove the volser from PROGRAM profiles that refer to SYSRES and its logical extensions. See *z/OS Security Server RACF Security Administrator's Guide* for more information.
- In DFSMShsm, a recovered or recalled data set catalog entry will not reflect system symbols used for indirect volume serials and may not return to its original volume. You may want to prevent migration of these data sets. See APAR OW24928 for more information.

Restrictions

- Indirect volume serial support can only be used for data sets residing on SYSRES and its logical extensions. These are typically data sets that are installed as part of a system build process. Use with user or application data sets is not supported.
- Cataloged data sets must be non-VSAM and non-SMS-managed.
- The volume must be mounted and online when a request is made to retrieve information from the catalog.
- You are responsible for managing volumes where the data sets reside. If you move a data set from one volume to another, you must make the related changes. Changes include:
  - setting up parmlib, proclib, and JCL.
  - establishing system management procedures such as security, backup, and recovery.
- The following **cannot** be catalogued with a system symbol:
  - SYS1.PARMLIB
  - Any parmlib data set listed in LOADxx without a volume name.

  Any parmlib data set (including SYS1.PARMLIB) **can** be catalogued with six asterisks (******).

Part 2. Members of SYS1.PARMLIB

Chapter 3. ADYSETxx (dump suppression)

ADYSETxx allows an installation to control dump analysis and elimination (DAE) processing, which suppresses dumps that it considers unnecessary because they duplicate previously taken dumps. DAE suppresses ABEND dumps that would be written to a SYSMDUMP data set (SYSMDUMPs), Transaction dumps (IEATDUMP), and SVC dumps, when the symptom data of a dump duplicates the symptom data of a dump of the same dump type previously taken. DAE uses the ADYSETxx parmlib member to determine the actions DAE is to perform.

To change the ADYSETxx parmlib member specification, enter the SET DAE command.

DAE can take the following actions for SYSMDUMP dumps, Transaction dumps, and SVC dumps in either a single system or in a sysplex:

- Matching, which means that DAE compares each dump occurrence for a dump type (SVC dump, or SYSMDUMP and Transaction dumps) to dumps previously recorded in the DAE data set.
- Updating, which means that DAE records either a unique symptom or an occurrence of a duplicate symptom string in the DAE data set.
- Suppressing, which means that DAE prevents a dump from being taken if the dump's symptom data duplicates the symptom data of a previous dump (of the same dump type) recorded in the DAE data set. Dump suppression must also be permitted through one of the following:
  - The SUPPRESS option is specified in the ADYSETxx parmlib member, and the VRADAE key is contained in the variable recording area of the SDWA.
  - The SUPPRESSALL option is specified in the ADYSETxx parmlib member, and the VRANODAE key is absent from the variable recording area of the SDWA.

For more information, see the following references:

- *z/OS MVS Diagnosis: Tools and Service Aids*, for information about DAE.
- *z/OS MVS System Commands*, for information about the SET DAE command.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for ADYSETxx

The following rules apply to the creation of ADYSETxx:

- Use one of the following rules for comments:
  - Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
  - An asterisk (*) in column one indicates a comment record. Note that comment records cannot be continued and that comments cannot appear on a parameter record.
- DAE= must appear first on a parameter record, and START or STOP must appear on the record. (Specify one or the other; START and STOP are mutually exclusive parameters.)

- Use commas or blanks to separate parameters and to separate subparameters. If you use a parameter (such as RECORDS or SVCDUMP) that requires a subparameter, you must enclose the subparameter in parentheses.

Syntax format of ADYSETxx

```
DAE={START[,RECORDS(400|n)]                                                  }
    {     [,SVCDUMP([MATCH][,SUPPRESS|SUPPRESSALL][,UPDATE]
[,NOTIFY(3,30)])]}
    {     [,SYSMDUMP([MATCH][,SUPPRESS|SUPPRESSALL][,UPDATE])]              }
    {     [,SHARE(DSN[,OPTIONS])]                                           }
    {     [,DSN(datasetname)]                                             }
    {     [,GLOBAL([DSN][,OPTIONS])]                                        }
    {STOP [,GLOBALSTOP]                                                     }
```

Syntax examples:

```
DAE=START,RECORDS(400),SVCDUMP(MATCH,UPDATE)
DAE=START,SYSMDUMP(MATCH,UPDATE,SUPPRESS),RECORDS(600)
DAE=START,SVCDUMP(MATCH,UPDATE,SUPPRESSALL,NOTIFY(5,60))
DAE=START,SVCDUMP(MATCH,SUPPRESSALL),SHARE(DSN,OPTIONS),DSN(SYS1.DAESHARE)
DAE=START,SVCDUMP(UPDATE,SUPPRESS),SHARE(DSN,OPTIONS),
          DSN(SYS1.DAESHARE2),GLOBAL(DSN,OPTIONS)
DAE=STOP
DAE=STOP,GLOBALSTOP
```

IBM-supplied defaults for ADYSETxx

IBM supplies three ADYSETxx parmlib members:

- ADYSET00 automatically starts DAE. It contains:

  ```
  DAE=START,RECORDS(400),SVCDUMP(MATCH,SUPPRESSALL,UPDATE,NOTIFY(3,30))
      SYSMDUMP(MATCH,UPDATE)
  ```

  Note: At system initialization, the ADYSET00 member is automatically in effect (DAE is active) because the IEACMD00 parmlib member contains the command SET DAE=00. If you do not want automatic activation of DAE, modify ADYSET00 to specify DAE=STOP.
- ADYSET01 allows the operator to stop DAE processing by issuing the command SET DAE=01. ADYSET01 contains:

  ```
  DAE=STOP
  ```

  Note: If the systems in a sysplex are sharing the DAE data set, IBM suggests that you update the ADYSET01 member as follows:

  ```
  DAE=STOP,GLOBALSTOP
  ```

  This will stop DAE on each system in the sysplex that shares the DAE data set. For more on dump suppression, see *z/OS MVS Diagnosis: Tools and Service Aids*.
- ADYSET02 allows the operator to start DAE processing by issuing the command SET DAE=02. ADYSET02 contains the same options as ADYSET00.

Statements/parameters for ADYSETxx

DSN(*datasetname***)**
Specifies the data set name to be used as the DAE data set. The data set must be accessible from each system that wants to share the data set.

The *datasetname* cannot be longer than 20 characters. SYS1.DAE is the default if you are not sharing the DAE data set. If you specify SHARE(DSN) and omit the DSN parameter, the DAE data set name is the data set already in use in the sysplex. For example, if the systems of the sysplex are sharing the DAE data set named SYS1.DAESHARE and the next system to join the sysplex specifies SHARE(DSN) without specifying the DSN parameter, that system will share the DAE data set SYS1.DAESHARE already in use by the sysplex.

If you share the DAE data set across the systems in a sysplex, IBM suggests that you use a data set name other than SYS1.DAE.

GLOBAL([DSN][,OPTIONS])
Specify GLOBAL when you want to change either the data set name, the DAE options (MATCH/SUPPRESS|SUPPRESSALL/UPDATE), or both for each system in the sysplex sharing the DAE data set.

When you specify GLOBAL(DSN), you must also specify SHARE(DSN) and DSN with the name of the DAE data set the sysplex is to use. DAE then begins to use the new DAE data set on all systems in the sysplex sharing the DAE data set. If you specify GLOBAL(DSN) without specifying DSN or without specifying SHARE(DSN), the system issues error message ADY002I.

When you specify OPTIONS, you can change the following for each system sharing the DAE options:

- The SYSMDUMP dump options: MATCH, SUPPRESS, SUPPRESSALL, or UPDATE.
- The SVCDUMP dump options: MATCH, SUPPRESS, SUPPRESSALL, NOTIFY, or UPDATE.

When you specify GLOBAL(OPTIONS), you must also specify SHARE(OPTIONS). If you specify GLOBAL(OPTIONS) without specifying SHARE(OPTIONS) and SVCDUMP and/or SYSMDUMP, the system issues error message ADY002I.

GLOBALSTOP
Specifies that DAE is to be stopped on each system in the sysplex currently sharing the DAE data set. This parameter is valid only when both of the following are true:

1. You specify STOP in this parmlib member.
2. You specified SHARE(DSN) when DAE was started.

If you do not specify GLOBALSTOP, you must specify STOP on each system in the sysplex.

MATCH
Specifies that DAE is to compare the symptoms from the current dump to those that have already been recorded in the DAE data set. (Coding MATCH does not indicate that DAE will suppress duplicate dumps or update the DAE data set.)

NOTIFY(*nnnn*,*tttt*)
Specifies the threshold at which an event notification facility (ENF) signal (event code 47) will be generated to notify a listener about SVC dumps completed, or suppressed, for a particular symptom string. The values, *nnnn* and *tttt* are decimal digits ranging from 1 to 9999 that specify the number of dumps and time interval in minutes. The time interval is measured from successive dump completions or suppressions and not dump initiations. The default is 3 dumps in 30 minutes for a particular symptom string.

Note:

1. NOTIFY is only valid when used with SVCDUMP.
2. UPDATE must be specified in order for NOTIFY to be effective.

RECORDS(*nnnn*)
Specifies the maximum number of symptom records to be placed in virtual storage, where *nnnn* is a decimal digit in the range 1 through 9999. The default is 400 records.

Note that the system obtains records in multiples of 20. Therefore, if you specify RECORDS(23), the system will round up the number of records you specified to the next multiple of 20, and place 40 records in storage instead of the 23 specified.

Each record includes information about each symptom, including:

- The date of the first and last occurrence of an error.
- The time of the first and last occurrence of an error.
- The system name of the first and last system to encounter an error.
- The actual symptom.

SHARE(DSN[,OPTIONS])
Specify SHARE when you want each system in the sysplex to share either:

- The DAE data set or
- The DAE data set and options for SVCDUMP and SYSMDUMP.

When you specify OPTIONS, you can set up the following for each system sharing the DAE options:

- The SYSMDUMP dump options: MATCH, SUPPRESS or SUPPRESSALL, and UPDATE.
- The SVCDUMP dump options: MATCH, SUPPRESS or SUPPRESSALL, UPDATE, and NOTIFY(3,30).

Specify OPTIONS only with DSN.

START
Specifies that DAE is to be started. If already active, DAE is first stopped and then started again.

Note: If you specify only DAE=START, the system does not issue an error message. However, unless you also specify SVCDUMP and/or SYSMDUMP, DAE does not perform dump processing and elimination.

STOP
Specifies that DAE is to be stopped and that all storage used by DAE is to be freed. This includes closing and unallocating the DAE data set.

SUPPRESS
Specifies that duplicate dumps are to be suppressed when all other criteria for matching and suppressing dumps are met, including that the VRADAE key is present in the variable recording area of the SDWA. (Coding SUPPRESS indicates that DAE will also match symptoms from the current dump to those already recorded; it does not indicate that DAE will update the DAE data set with any new symptoms.)

SUPPRESS and SUPPRESSALL are mutually exclusive. If you specify both SUPPRESS and SUPPRESSALL, the system issues message ADY001I.

If you specify SUPPRESSALL, DAE will suppress more dumps because the VRADAE key in the SDWA is not required to suppress a dump. If you omit both SUPPRESS and SUPPRESSALL, DAE will not suppress any dumps.

SUPPRESSALL
Specifies that duplicate dumps are to be suppressed when all criteria for matching and suppressing dumps are met with the exception of the VRADAE key. When SUPPRESSALL is specified, the system does not require the VRADAE key to be present in the variable recording area of the SDWA.

If SUPPRESSALL is specified and the VRANODAE key is present in the variable recording area of the SDWA, DAE will not suppress the dump. (Coding SUPPRESSALL indicates that DAE will also match symptoms from the current dump to those already recorded; it does not indicate that DAE will update the DAE data set with any new symptoms.)

SUPPRESSALL and SUPPRESS are mutually exclusive. If you specify both SUPPRESS and SUPPRESSALL, the system issues message ADY001I.

If you specify SUPPRESSALL, DAE will suppress more dumps because the VRADAE key in the SDWA is not required to suppress a dump. If you omit both SUPPRESS and SUPPRESSALL, DAE will not suppress any dumps.

SVCDUMP(*dump options*)
Requests that DAE process SVC dumps, using one or more of the following subparameters:

- MATCH
- SUPPRESS or SUPPRESSALL
- UPDATE
- NOTIFY

NOTIFY(3,30) is the default if UPDATE is specified with SVCDUMP.

SYSMDUMP(dump options)
Requests that DAE process SYSMDUMPs and transaction dumps using one or more of the following subparameters; there is no default.

- MATCH
- SUPPRESS or SUPPRESSALL
- UPDATE

To enable suppression of duplicate SYSMDUMPs and transaction dumps, specify

```
SYSMDUMP(MATCH,SUPPRESS,UPDATE)
                or
SYSMDUMP(SUPPRESSALL,UPDATE)
```

then enter,

```
SET DAE=xx
```

at the MVS console.

To suppress all SYSMDUMPs and transaction dumps, enter the following from the MVS console:

```
CHNGDUMP SET,SYSMDUMP,NODUMP
```

UPDATE
Specifies that the DAE data set is to be updated with the results of matching. The update can be a new record for a new set of symptoms or an increase to the count of occurrences for a duplicate set of symptoms. (Coding UPDATE

indicates that DAE will also match symptoms from the current dump to those already recorded; it does not indicate that DAE will suppress any duplicate dumps.)

If you do not specify UPDATE, DAE will not update the DAE data set. Specify UPDATE if you plan to suppress dumps, or to be notified when a threshold is reached for SVC dumps completed or suppressed for a particular symptom string.

Chapter 4. ALLOCxx (allocation system defaults)

Use the ALLOCxx member of SYS1.PARMLIB to define installation defaults for:
- Unit names (dynamic allocation, unit-affinity-ignored, and redirection from TAPE)
- Space attributes
- TIOT size
- Handling allocation requests
- Catalog error policies.

These installation defaults for handling allocation requests can be overridden by installation exit routines specified in the EXITxx parmlib member. For information about the allocation exit routines, see *z/OS MVS Installation Exits*. After IPL, use the SETALLOC command to change any of the defaults (except for 2DGT_EXPDT).

Parameter in IEASYSxx (or supplied by the operator)

```
ALLOC=   {aa        }
         {(aa,bb...)}
```

The two-character identifier (*aa*, *bb*, and so forth) is appended to ALLOC to identify the ALLOCxx member. Multiple members can be specified. If you specify a parameter in more than one member, the system uses the value in the first member, and ignores the values in the subsequent members.

Note: The system does not set default to ALLOC00. If neither the IEASYSxx member nor the operator specifies an ALLOC= parameter, the system uses the defaults identified in "Statements/parameters for ALLOCxx" on page 75.

Syntax rules for ALLOCxx

The following syntax rules apply to ALLOCxx:
- Use columns 1 through 71. Do not use columns 72 through 80 for data; these columns are ignored.
- At least one delimiter (space or comma) is required between a statement and keyword. Delimiters are not required between keywords.
- Comments may appear in columns 1-71 and must begin with /* and end with */.

Syntax format of ALLOCxx

```
SPACE      PRIMARY(nnnnnnn)
           SECONDARY(nn)
           DIRECTORY(n)
                    {TRK                                  }
           MEASURE  {CYL                                  }
                    {AVEBLK    BLKLNGTH(nnnn) {ROUND  }   }
                    {                         {NOROUND}   }

                    {CONTIG}
           PRIM_ORG {MXIG  }
                    {ALX   }

           {RLSE  }
           {NORLSE}

UNIT       NAME(unit-name)
           UNITAFF(unit-name)

           REDIRECTED_TAPE {TAPE}
                           {DASD}

TIOT       SIZE(nn)

SDSN_WAIT  WAITALLOC {NO  }
                     {YES }

VOLUME_ENQ POLICY  {WTOR  }
                   {CANCEL}
                   {WAIT}

                   {WTOR  }
VOLUME_MNT POLICY  {CANCEL}

                  {WTOR   }
SPEC_WAIT  POLICY {WAITHOLD}
                  {WAITNOH}
                  {CANCEL }

           MAXNWAIT(nnn)

           POLICYNW   {WTOR  }
                      {CANCEL}

                  {WTOR   }
ALLC_OFFLN POLICY {WAITHOLD}
                  {WAITNOH}
                  {CANCEL }

           MAXNWAIT(nnn)

           POLICYNW   {WTOR  }
                      {CANCEL}
CATLG_ERR  FAILJOB {YES}
                   {NO }

           ERRORMSG {YES}
                    {NO }

2DGT_EXPDT POLICY {ALLOW}
                  {WARN}
                  {FAIL}

VERIFY_VOL POLICY {YES}
                  {NO}
```

```
SYSTEM     IEFBR14_DELMIGDS(LEGACY|NORECALL)
           TAPELIB_PREF(EQUAL|BYDEVICES)
           REMIND_INTV(xxx)
           TEMPDSFORMAT(UNIQUE|INCLUDELABEL)
           VERIFY_UNCAT(FAIL|TRACK|MSGTRACK|LOGTRACK)
           MEMDSENQMGMT(ENABLE|DISABLE)
           BATCH_RCLMIGDS(SERIAL|PARALLEL)
           OPTCDB_SPLIT(EXPLICIT|CATALOG)
```

Syntax example of ALLOCxx

```
2DGT_EXPDT    POLICY(ALLOW)        /*Allow data set allocation with two-digit date*/
SPACE         PRIMARY(10)          /*Primary Space Quantity*/
              SECONDARY(50)        /*Secondary Space Quantity*/
              BLKLNGTH(1000)       /*Block Length*/
              DIRECTORY(0)         /*Default to Sequential*/
              MEASURE(AVEBLK)      /*Average Block Length*/
              PRIM_ORG(CONTIG)     /*Contiguous Organization*/
              RLSE                 /*Release Unused Space*/

UNIT          NAME(SYSALLDA)       /*SYSALLDA is Default*/
              UNITAFF(CART)        /*Cartridge is the default
                                     esoteric for tape*/
              REDIRECTED_TAPE(DASD) /*Treat unopened
                                     batch-allocated DASD
                                     data sets, which have
                                     been redirected from
                                     TAPE as DASD. Default
                                     is TAPE.*/

TIOT          SIZE(32)             /*32K TIOT Size*/

SDSN_WAIT     WAITALLOC(NO)        /*Do not wait for
                                       special data sets*/

VOLUME_ENQ    POLICY(CANCEL)       /*Always cancel job*/

VOLUME_MNT    POLICY(WTOR)         /*Always issue the WTOR*/

SPEC_WAIT     POLICY(WAITNOH)      /*Wait while
                                       not holding resources*/
              MAXNWAIT(7)          /*7 "wait nohold"
                                     decisions allowed*/
              POLICYNW(CANCEL)     /*Cancel if wait is
                                     not allowed*/

ALLC_OFFLN    POLICY(WAITNOH)      /*Wait
                                      while not holding resources*/
              MAXNWAIT(7)          /*7 "wait nohold"
                                     decisions allowed*/
              POLICYNW(CANCEL)     /*Cancel if wait is
                                     not allowed*/
CATLG_ERR     FAILJOB(YES)         /*Fail the job*/
              ERRORMSG(YES)        /*Issue the WTO*/

VERIFY_VOL POLICY(YES)            /* Requests verification of premounted
                                    or PASSed/RETAINed volumes on
                                    AutoSwitchable (AS) Tape devices. */
SYSTEM        IEFBR14_DELMIGDS(LEGACY) /*Indicates that the system is to
                                       automatically recall HSM-migrated
                                       data sets before deletion.*/
              TAPELIB_PREF(EQUAL)      /*Indicates that for non-specific
                                       tape library requests, such
                                       as scratch tape requests, all
                                       tape libraries must be treated
                                       as equal, and receive an
                                       equal share of the non-specific
                                       tape requests.*/
              REMIND_INTV(90)          /*90 seconds between each IEF882I
                                       or IEF883I reminder message.*/
              TEMPDSFORMAT(UNIQUE)     /*Creates unique
                                       temporary data
                                       set names.*/
              VERIFY_UNCAT(FAIL)       /*Indicates that UNCATLG
                                       requests fail if the data set
                                       information is not retrieved
                                       from the catalog.*/
              MEMDSENQMGMT(ENABLE)     /*Allows jobs and subsystems to use the
                                       memory-based ENQ management
```

```
OPTCDB_SPLIT(EXPLICIT)   /*Indicates that the
                         system should treat
                         multi-volume data sets
                         in system-managed tape
                         libraries as separate
                         DD statements only
                         when DCB=OPTCD=B is
                         is specified with
                         explicit volume serial
                         numbers.*/
```

IBM-supplied default for ALLOCxx

The system does not set default to ALLOC00. If neither the IEASYSxx member nor the operator specifies an ALLOC= parameter, the system uses all of the defaults identified in "Statements/parameters for ALLOCxx." If an ALLOCxx member is specified, but any individual parameters are not specified in that ALLOCxx member, the system uses the default for that parameter identified in "Statements/parameters for ALLOCxx."

Statements/parameters for ALLOCxx

2DGT_EXPDT

Identifies the action to be taken if a new data set allocation request specifies a two-digit year (yyddd) Expiration Date using one of the following:

- EXPDT — Batch JCL two-digit year Expiration Date Keyword.
- DALEXPDT — Dynamic Allocation Expiration Date specification short form.

Note:

1. A two digit year is always treated as 19yy (yyddd).
2. Dates of 00000, 98000, 99000, 99365, and 99366 are allowed regardless of the Policy in force.

POLICY(ALLOW)

Allow the data set allocation with no Expiration Date related message.

POLICY(WARN)

Allow the data set allocation but issue a warning message. If a 2-digit year is specified on a JCL DD statement (EXPDT), warning message IEF428I is written to both the System Message and JCL Message portions of the Job Log. If the 2-digit year is specified on a Dynamic data set allocation (DALEXPDT), warning message IEF405I is written to the operator and Dynamic code '0054'x is returned to the SVC 99 caller.

POLICY(FAIL)

Fail the data set allocation and issue a failure message. If a 2-digit year is specified on a JCL DD statement (EXPDT), failure message IEF429I is written to both the System Message and JCL Message portions of the Job Log. If the 2-digit year is specified on a Dynamic data set allocation (DALEXPDT), failure message IEF406I is written to the operator and Dynamic Allocation Error Return Code '000C'x with Class 3 Reason Code '03B8'x is returned to the SVC 99 caller.

Default: POLICY(ALLOW)

SPACE

Specifies the installation defaults for some space allocation parameters. These

defaults apply to only dynamic allocation and VIO requests. Understand that space allocations specified on JCL (for VIO requests), on dynamic allocation, or in SMS data classes take precedence over the values coded on this statement.

PRIMARY(nnnnnnn)
Specifies one of the following:

- For TRK, the number of tracks to be allocated.
- For CYL, the number of cylinders to be allocated.
- For AVEBLK, the number of average data blocks in the data set. Use the BLKLNGTH parameter to specify the length of the average data block.

When you specify TRK or CYL for a partitioned data set (PDS), the primary quantity includes the space for the directory. When you specify a block length for a PDS, the primary quantity does not include the directory space; the system assigns the directory space outside the primary space assignment.

One volume must have enough available space for the primary quantity. If you request a particular volume and it does not have enough space available for your request, the system ends the job step. Allow for track overflow when computing track requirements.

To request an entire volume, specify in the primary quantity the number of tracks or cylinders on the volume minus the number used by the volume table of contents (VTOC). The volume must not contain other data sets.

Value Range: 0 - 16777215

Default: 4

SECONDARY(nnnnnnn)
Specifies the number of additional tracks, cylinders, blocks, or records to be allocated, if more space is needed. The system does not allocate additional space until it is needed.

If PRIMARY specifies the average block length, the system computes the number of tracks for the secondary quantity from the SECONDARY value multiplied by one of the following, in order:
1. The SPACE average block length subparameter.
2. The block length in the BLKSIZE field of the data control block.

When you specify SECONDARY and the data set requires additional space, the system allocates the specified quantity:
1. In contiguous tracks or cylinders, if available.
2. If not, in up to five extents.

The system can allocate up to 16 extents for a data set on a volume. An extent is space that may or may not be contiguous to other space allocated to the data set. The extents for a data set include the primary quantity space and user-label space.

Note: BDAM data sets cannot be extended.

When your program has filled the allocated space on a volume for a sequential data set, the system determines where the following data is written as follows:

- If the disposition of the data set is NEW or MOD and the limit on the number of extents on a volume has not been reached, the system attempts to allocate the secondary quantity on the same volume.

- If the disposition of the data set is OLD or SHARE, the system examines the next volume specified for the data set.
  - If space has been allocated on the next volume for the data set, the next volume is used for the data set.
  - If space has not been allocated on the next volume for the data set, secondary space is allocated on the next volume for the data set.
  - If there is not another volume specified for the data set, the system attempts to allocate the secondary quantity on the current volume.

  Note that your program should not write with a disposition of DISP=SHR unless you take precautions to prevent other programs from writing at the same time.

If the requested volumes have no more available space and if at least one volume is demountable, the system asks the operator to mount scratch (nonspecific) volumes until the secondary allocation is complete. If none of the volumes are demountable, the system abnormally ends the job step.

Value Range: 0 - 16777215

Default: 24

DIRECTORY(nnnnnnn)
Specifies the number of 256-byte records needed in the directory of a PDS.

Note: When creating a PDS, you must request space for a directory.

Value Range: 0 - 8388607

Default: 0

MEASURE([TRK|CYL|AVEBLK])
Specifies the unit of measure of the space allocation as one of the following:

TRK Requests that space be allocated in tracks.

CYL Requests that space be allocated in cylinders.

AVEBLK
Requests that the system is to decide how many tracks to allocate based on the average block size. The size of the average block is specified using the BLKLNGTH parameter, and the number of blocks is specified using the PRIMARY parameter.

BLKLNGTH(nnnnn)
Specifies, in bytes, the average block length of the data. The system computes how many tracks to allocate.

Value Range: 0 - 65535

Default: 8192

ROUND|NOROUND
When MEASURE(AVEBLK) is specified, requests whether (ROUND) or not (NOROUND) space allocated to the data set must be equal to an integral number of cylinders.

Default: NOROUND

Default: AVEBLK

PRIM_ORG([CONTIG|MXIG|ALX])
Specifies the organization of the primary space allocation as one of the following:

CONTIG
Requests that space allocated to the data set be contiguous. If CONTIG is specified and contiguous space is not available, the system ends the job step.

MXIG Requests that space allocated to the data set must be (1) the largest area of available contiguous space on the volume and (2) equal to or greater than the value specified on the PRIMARY parameter.

Caution: IBM suggests that you use extreme care when coding this parameter. Large amounts of storage could be allocated, depending on how much free space is available at the time the request is made. If you code this parameter, IBM suggests that you also code the RLSE parameter to release any unused space.

Note:
1. Do not code MXIG for an indexed sequential data set.
2. MXIG can also be specified in a job's JCL.

ALX Requests that up to five of the largest separate areas of available contiguous space are to be allocated to the data set, and each area must be equal to or greater than the value specified on the PRIMARY parameter.

For example, assume the following space extents (in tracks) are available: 910, 435, 201, 102, 14, 12, and 8.

If your job requests 14 tracks as its primary allocation, and ALX is in effect, the job receives the following 5 extents: 910, 435, 201, 102, and 14.

However, if the job requests 15 tracks as its primary allocation, it would received 4 extents: 910, 435, 201, and 102. The job does not receive the 14-track extent because it is less than the primary space allocation.

Caution: IBM suggests that you use extreme care when coding this parameter. Large amounts of storage could be allocated, depending on how much free space is available at the time the request is made. If you code this parameter, IBM suggests that you also code the RLSE parameter to release any unused space.

Note: ALX can also be specified in a job's JCL.

Default: None

RLSE|NORLSE
Requests whether (RLSE) or not (NORLSE) space allocated to an output data set, but not used, is to be released when the data set is closed, and the CLOSE macro does not specify TYPE=T. Unused space is released only if the data set is open for output and the last operation was a write. Coding RLSE for primary allocation does not prohibit use of secondary allocation. The secondary request for space is still in effect.

The system ignores a request to release unused space when a data set is closed if:

- Another job is sharing the data set.

- Another task in the same job is processing an OPEN, CLOSE, EOV, or FEOV request for the data set.
- Another data control block is open for the data set.
- The data set is an indexed sequential data set.

Default: RLSE

UNIT

Specifies the installation default for the device on which the system is to place data sets.

REDIRECTED_TAPE(TAPE|DASD)

Allows the installation to specify whether unopened batch-allocated DASD data sets that were redirected from tape should be treated as DASD or TAPE.

Specifying REDIRECTED_TAPE(TAPE) causes unopened batch allocated data sets that have been redirected from TAPE to DASD to be deleted during final disposition processing. These unopened redirected data sets are deleted regardless of the disposition requested.

Specifying REDIRECTED_TAPE(DASD) causes unopened batch allocated data sets that have been redirected from TAPE to DASD to be processed according to the original disposition, as they would have been if they had been directed to DASD and not redirected to DASD from TAPE.

Note: Dynamic allocation of SMS DASD data sets that were redirected from TAPE will continue to be treated as DASD during dynamic allocation.

Default: TAPE

NAME(groupname)

Requests a group of devices to place data sets on. The installation must have assigned the name to the devices(s) during system initialization or IBM must have assigned the name. This default applies only to dynamic requests.

A group-name can identify a single device or a group of devices. A group can consist of devices of the same or different types. For example, a group can contain both direct access storage devices (DASD) and tape devices.

The system assigns any available device from the group. If a group consists of only one device, the system assigns that device. If the group consists of more than one device type, the units requested are allocated from the same device type. For example, if GPDA contains 3330 Disk Storage and 3350 direct access storage devices (DASD), a request for two units would be allocated to two 3330s or to two 3350s.

If a data set that was created using the group-name subparameter is to be extended, the system allocates additional devices of the same type as the original devices. However, the additional devices may not necessarily be from the same group.

Dynamic Allocation Consideration: If a time sharing user's dynamic allocation request does not include unit information, the system obtains a unit description from:

- The PDF_DEFAULT_UNIT parameter for certain ISPF/PDF data sets. See Data set allocation settings in *z/OS ISPF Planning and Customizing* for specific information.

- The SYS1.UADS entry for all other ISPF/PDF data sets and for all non-ISPF/PDF TSO data sets.

If the user is not a time sharing user, or if the SYS1.UADS entry does not contain a unit description, the system uses the unit name specified on the UNIT keyword of the ALLOCxx parmlib member as the default.

The unit description you supply in your dynamic allocation request can override the unit type for a cataloged data set. The unit description from the SYS1.UADS, however, cannot override the unit information in the catalog.

Value Range: 1 to 8 alphanumeric characters.

Default: SYSALLDA, which contains all DASD defined to the system.

UNITAFF(unit-name)

Specifies the installation default for the unit name on which the system is to place data sets when the following conditions are true:

- The data set for the referencing DD, that is, the DD that specifies UNIT=AFF, DISP=NEW or DISP=MOD (MOD treated as NEW), and is not SMS-managed.
- The data set for the referenced DD, that is, the DD statement pointed to by the UNIT=AFF subparameter, is SMS-managed.
- The allocation is not part of a data set collection involving data set stacking.
- The system cannot obtain a unit name from the primary DD statement in the unit affinity chain.

The installation must have assigned the name to the device(s) during system initialization, or IBM must have assigned the name.

Unit-name can be a group name. A group-name can identify a single device or a group of devices. A group can consist of devices of the same or different types. For example, a group can contain both direct reel tape devices (3400) and cartridge tape devices (3480).

The system assigns any available device from the group. If a group consists of only one device, the system assigns that device. If the group consists of more than one device type, the units requested are allocated from the same device type. For example, if TAPEGRP contains both 3400 devices and 3480 devices, a request for two units would be allocated to two 3400s or to two 3480s.

If a data set that was created using the UNITAFF subparameter is to be extended, the system allocates additional devices of the same type as the original devices. However, the additional devices might not necessarily be from the same group.

If the name specified by UNITAFF does not exist in the eligible devices table (EDT), the system default is used instead and a warning message is issued.

Note: If the UNITAFF subparameter is not specified for any given system, make sure the device preference order is the same on all systems. In this case, a system-derived default is used, that is, the tape generic highest in the device preference table.

Value Range: 1 to 8 alphanumeric characters.

Default: Tape generic highest in the device preference table. This generic must be available on every IODF used on the system between this IPL and the next IPL.

TIOT

Specifies the installation defaults for the task I/O table (TIOT).

SIZE(nn)

Specifies the size of the TIOT. The TIOT contains an entry for each DD statement. The size of the TIOT controls how many DDs are allowed per jobstep. By specifying an integer from 16 to 64 as the value of this parameter, the installation controls the default DD allowance. Table 13 shows the relationship between the size of the TIOT and the maximum number of DDs allowed.

Table 13. Relationship size of TIOT and maximum number of DDs allowed

Dec	Hex	Size of TIOT	Maximum number of single unit DD Allowed	Maximum number of DDs allowed when every DD requests the maximum number of units (59)
16	10	16384 (16K)	816	64
17	11	17408 (17K)	867	68
24	18	24576 (24K)	1225	97
25	19	25600 (25K)	1277	101
32	20	32768 (32K)	1635	129
40	28	40960 (40K)	2045	162
48	30	49152 (48K)	2454	194
56	38	57344 (56K)	2864	227
64	40	65536 (64K)	3273	259

Note:

1. Your calculations need to take into account that the size of a TIOT entry, for a DD statement or a Dynamic Allocation, increases by four (4) bytes for every SMS Candidate volume assigned (e.g., by your DATACLAS), regardless of whether they're guaranteed space.
2. For a VSAM KSDS the number of 4-byte entries in the TIOT for the data set depends on whether or not the data set is defined as reusable. The count of entries in the TIOT is the count of candidate volumes for the data and index components plus:
   - For a reusable data set - the number of volumes used by the data component plus the number of volumes used by the index component.
   - For a nonreusable data set - the number of volumes in the set of volumes used by the data and index component.
3. Use the following to calculate the maximum number of DDs allowed per Job Step:
   a. The TIOT Prefix, Header, and Trailer consume sixty (60) ('3C'x) bytes of the total TIOT space available to a Job Step.
   b. A DD statement requesting a single unit requires twenty (20) bytes ('14'x) of TIOT space. The TIOT space requirement for entire step is 80 bytes.

```
//TAPEJOB   JOB
//STEP1     EXEC   PGM=IEFBR14
//DD1       DD     UNIT=3490     ** DD requires 20 bytes *
```

c. A DD statement requesting two (2) units requires twenty four (24) bytes ('18'x) of TIOT space. Twenty bytes for the basic information for the first unit and an additional four bytes for the second unit. The TIOT space requirement for entire step is 84 bytes.

```
//TAPEJOB   JOB
//STEP1     EXEC   PGM=IEFBR14
//DD1       DD     UNIT=(3490,2) ** DD requires 24 bytes *
```

d. A DD requesting the maximum number of units allowed, fifty nine (59), utilizes two hundred fifty two (252) bytes ('FC'x) of TIOT space. The TIOT space requirement for entire step is 312 bytes.

```
//TAPEJOB   JOB
//STEP1     EXEC   PGM=IEFBR14
//DD1       DD     UNIT=(3490,59) ** DD requires 252 bytes *
```

e. A Job Step with three (3) DD statements and each DD requesting one more unit than the previous DD would use the following TIOT space; TIOT space requirement for entire step is 132 bytes.

```
//TAPEJOB   JOB
//STEP1     EXEC   PGM=IEFBR14
//DD1       DD     UNIT=3490      ** DD requires 20 bytes *
//DD2       DD     UNIT=(3490,2)  ** DD requires 24 bytes *
//DD3       DD     UNIT=(3490,3)  ** DD requires 28 bytes *
```

Value Range: 16 - 64 kilobytes

Default: 32 kilobytes

SDSN_WAIT

Specifies the installation policy for batch jobs that must wait to enqueue on special types of data set names.

WAITALLOC([NO|YES])

Specifies whether to cancel jobs that must wait to enqueue on the following types of data set names:

- GDG absolute generation data set name (unless the absolute generation data set name is specified on the JCL). See *z/OS MVS JCL Reference* for a discussion of how the system generates absolute generation names and how the system enqueues on absolute generation names when a relative generation name is specified.
- Real data set name (when its corresponding alias data set name is specified on the JCL).

When YES is specified, and a batch job's enqueue request cannot be satisfied, the system issues messages IEF861I, IEF863I and IEF458D. The job waits, holding any resources it might have acquired. The system operator can choose to cancel the job in response to message IEF458D, or allow the job to continue waiting until the enqueue becomes available. If the operator cancels the job, the system writes an informational message (IEF330I) to the job log.

When NO is specified, the system cancels the job, releases its resources, and issues message IEF211I.

Note:

1. Use caution when specifying YES. Allowing jobs to wait for data set availability can cause deadlocks with other jobs in the system.

2. When you specify YES, the system does not allow the job to wait for a data set when both of the following conditions are true:
   a. This job plus one (or more) other jobs have the data set allocated as DISP=SHR
   b. This job requests that its use of the data set be upgraded from DISP=SHR to DISP=OLD. The system ends this job and issues message IEF211I.
3. The WAITALLOC option only applies to batch allocation requests (that is, allocation requests specified in the job's JCL).

Default: NO

VOLUME_ENQ
Specifies the installation policy for enqueuing on volumes when an allocation request has to wait for a volume or a series of volumes.

POLICY ([WTOR|CANCEL|WAIT])
Specifies the default action to take. An installation exit can override the policy.

WTOR
The installation policy is to issue the message and let the operator make the decision about the allocation request. The system displays one of the following messages on the operator's console:

- IEF690I - The following volumes are unavailable to <jobname>...
- IEF235D - <jobname> is waiting for volumes. To cancel wait, reply no.

In addition, the system issues message IEF369D (invalid reply) in response to an invalid reply to IEF235D.

CANCEL
The installation policy is to cancel a job that needs an unavailable volume. The system cancels the job, releases its resources, and issues message IEF251I.

WAIT
The installation policy is to let a job that needs an unavailable volume wait until the volume is available.

Caution: When WAIT is used as the default, deadlocks with other jobs in the system might arise for tape volumes.

Default: WTOR

VOLUME_MNT
Specifies the installation policy for mounting a volume when an allocation request requires a volume to be mounted. MVS invokes the exit when processing mount requests for single volumes or the first volume of a multi-volume request. MVS does not invoke the exit for tape mount requests that specify UNIT=DEFER nor second and subsequent volumes of a multi-volume request. Use the EOV exit routine to handle second and subsequent volumes (see *z/OS DFSMS Installation Exits* for information).

POLICY ([WTOR|CANCEL])
Specifies the default action to take. An installation exit can override the policy.

WTOR
The installation policy is to issue the message and let the operator

make the decision about the volume mount. The system displays one or more of the following messages on the operator's console:

- IEF233A - Mount volume <ser>
- IEF233D - Mount volume <ser> or respond to IEF455D message
- IEF455D - Mount <ser> on <device> for <jobname> or reply no.
- IEF488I - Must wait for a unit, or volume on unit.

In addition, the system issues message IEF369D (invalid reply) in response to an invalid reply to IEF455D.

CANCEL
The installation policy is to cancel a job that needs a volume mounted. The system cancels the job, releases its resources, and issues message IEF251I.

Default: WTOR

SPEC_WAIT
Specifies the installation policy to be followed when an allocation request must wait for a specific volume or unit.

POLICY ([WTOR|WAITHOLD|WAITNOH|CANCEL])
Specifies the default action to take. An installation exit can override the policy.

WTOR
The installation policy is to issue the message and let the operator make the decision about the wait request. The system displays one or more of the following messages on the operator's console:

- IEF238D - Reply [device name] [,] ['wait'] or 'cancel'
- IEF244I - Unable to allocate <nnn> units(s). At least <nnn> allocated or offline units are needed.
- IEF433D - Wait requested — reply hold or nohold
- IEF488I - Must wait for a unit, or volume on unit.

In addition, the system issues one or more of the following messages in response to an invalid reply to the preceding messages.

- IEF434D - Invalid reply (to message IEF433D). Reply hold or nohold.
- IEF490I - Invalid reply (to message IEF238D) for one of the following reasons:
  - Device is not accessible (no paths available, boxed, or cannot be assigned)
  - Required system managed volume is not available
  - Required volume is not available
  - Replied device is not eligible
  - Device is found in an offline library.
  - Coupling facility error

WAITHOLD
The installation policy is for the system to not release any of the devices that have already been allocated to this job before it waits for the required units or volumes. The system issues message IEF289E.

Be aware that using the WAITHOLD policy might cause a deadlock situation, particularly when the device is being used by a job that is going to wait. The system does not release any non-sharable devices

(that is, non-DASD) that have already been allocated to the job before it waits for required units and volumes. To avoid this problem, do not specify WAITHOLD.

When devices for a job are held during a wait, and a device that was eligible for allocation to the job becomes ineligible for allocation (because of its use by a system utility, for example), the job might fail because it does not have enough devices to complete successfully. Message IEF700I in the job log identifies this failure. Refer to message IEF700I for information about how to respond to this failure.

WAITNOH
The installation policy is to let the job wait while not holding the obtained resources. The system will release those devices that have been allocated to this job, but that cannot be shared with other jobs. The system issues message IEF289E.

For an example of the WAITHOLD versus WAITNOH options, consider that JOBA owns an automatically switchable device and is waiting for a printer. JOBB owns the printer JOBA needs and is waiting for the automatically switchable device JOBA owns.

- If the reply is WAITHOLD for each job, the two jobs will wait until one job is canceled. This deadlock can be even more complex depending on the number of jobs waiting.
- If the reply is WAITNOH for each job, allocation responds on a first-come, first-served basis. After the first job finishes using a resource, it is available to the second.

CANCEL
The installation policy is to cancel the allocation request. If a TSO/E user issued the allocation request, the user receives an error message. If a batch job or started task issued the request, the system cancels the job or task, releases its resources, and issues message IEF251I.

Default: WTOR

MAXNWAIT(nnn)
Specifies the number of "WAITNOH" decisions allowed to be made for the specific volume or unit allocation request before the default specified on the POLICYNW parameter will take effect. The WAITNOH decisions counted are those that are specified either through the default on the POLICY parameter or through an installation exit specified in the EXITxx parmlib member. "WAITNOH" decisions made by the operator are not included in the MAXNWAIT count.

Value Range: 1 - 255

Default: 5

POLICYNW(CANCEL|WTOR)
Specifies how the system should handle the allocation request under the following circumstances:

- Either WAITHOLD or WAITNOH is specified on the POLICY parameter and the system does not allow the job to wait for resources.
- The maximum number of "WAITNOH" decisions (specified on the MAXNWAIT parameter) has been exceeded.

The system is to either cancel the allocation request (CANCEL) or issue a message (WTOR). When WTOR is selected, the system issues the messages

listed earlier under WTOR. When CANCEL is selected, the system cancels the allocation request depending on how the request was issued. If a TSO/E user issued the allocation request, the user receives an error message. If a batch job or started task issued the request, the system cancels the job or task, releases its resources, and issues message IEF251I.

Default: WTOR

ALLC_OFFLN
Specifies the installation policy to be followed when an allocation request needs a device that is offline, or must wait for a non-specific volume or unit.

Note that if all eligible devices are offline, they cannot be brought online without operator intervention. In this case, the system ignores the WAITHOLD and WAITNOH options and issues the WTOR immediately.

POLICY ([WTOR|WAITHOLD|WAITNOH|CANCEL])
Specifies the default action to take. An installation exit can override the policy.

WTOR
The installation policy is to issue the message and let the operator make the decision about the needed device. The system displays one or more of the following messages on the operator's console:

- IEF157E - <jobname> needs <nnn> units. All eligible units are currently allocated.
- IEF238D - Reply [device name] [,] ['wait'] or 'cancel'
- IEF244I - Unable to allocate <nnn> units(s). At least <nnn> allocated or offline units are needed
- IEF433D - Wait requested — reply hold or nohold.

In addition, the system issues one or more of the following messages in response to invalid replies to the preceding messages:

- IEF434D - Invalid reply (to message IEF433D). Reply hold or nohold.
- IEF490I - Invalid reply (to message IEF238D) for one of the following reasons:
  - Device is not accessible
  - Required system managed volume is not available
  - Required volume is not available
  - Replied device is not eligible
  - Device could not be found in the configuration
  - Device found in an offline library.

WAITHOLD
The installation policy is for the system to not release any of the devices that have already been allocated to this job before it waits for the required units or volumes. The system issues message IEF289E.

Be aware that using the WAITHOLD policy might cause a deadlock situation, particularly when the device is being used by a job that is going to wait. The system does not release any non-sharable devices (that is, non-DASD) that have already been allocated to the job before it waits for required units and volumes. To avoid this problem, do not specify WAITHOLD.

When devices for a job are held during a wait, and a device that was eligible for allocation to the job becomes ineligible for allocation (because of its use by a system utility, for example), the job might fail because it does not have enough devices to complete successfully.

Message IEF700I in the job log identifies this failure. Refer to message IEF700I for information about how to respond to this failure.

WAITNOH
The installation policy is to let the job wait while not holding the obtained resources. The system will release those devices that have been allocated to this job, but that cannot be shared with other jobs. The system issues message IEF289E.

For an example of the WAITHOLD versus WAITNOH options, consider that JOBA owns an automatically switchable device and is waiting for a printer. JOBB owns the printer JOBA needs and is waiting for the automatically switchable device JOBA owns.

- If the reply is WAITHOLD for each job, the two jobs will wait until one job is canceled. This deadlock can be even more complex depending on the number of jobs waiting.
- If the reply is WAITNOH for each job, allocation responds on a first-come, first-served basis. After the first job finishes using a resource, it is available to the second.

CANCEL
The installation policy is to cancel the allocation request. If a TSO/E user issued the allocation request, the user receives an error message. If a batch job or started task issued the request, the system cancels the job or task, releases its resources, and issues message IEF251I.

Default: WTOR

MAXNWAIT(nnn)
Specifies the number of "WAITNOH" decisions allowed to be made for the allocation request before the default specified on the POLICYNW parameter will take effect. The WAITNOH decisions counted are those that are specified either through the default on the POLICY parameter or through an installation exit in the EXITxx parmlib member. "WAITNOH" decisions made by the operator are not included in the MAXNWAIT count.

Value Range: 1 - 255

Default: 5

POLICYNW(CANCEL|WTOR)
Specifies how the system should handle the allocation request under the following circumstances:

- Either WAITHOLD or WAITNOH is specified on the POLICY parameter and the system does not allow the job to wait for the needed device.
- The maximum number of "WAITNOH" decisions (specified on the MAXNWAIT parameter) has been exceeded.

The system is to either cancel the allocation request (CANCEL) or issue a WTOR. When WTOR is selected, the system issues the messages listed earlier under WTOR. When CANCEL is selected, the system cancels the allocation request depending on how the request was issued. If a TSO/E user issued the allocation request, the user receives an error message. If a batch job or started task issued the request, the system cancels the job or task, releases its resources, and issues message IEF251I.

Default: WTOR

CATLG_ERR
Specifies the installation policy for handling certain types of errors that might

occur when the system processes the disposition of batch unallocated data sets (data sets that have been unallocated at step termination time). The CATLG_ERR statement applies when the system is unable to:

- Catalog a new data set for which the user specified a disposition of CATLG.
- Catalog an old uncataloged data set for which the user specified a disposition of CATLG.
- Recatalog an old cataloged data set for which the volume list was extended, and for which the user specified a disposition of CATLG, KEEP or PASS.
- Roll an SMS-managed generation data set into the GDG base.

The CATLG_ERR statement does not apply when the user unallocates a data set before step termination through the following methods:

- Dynamic deallocation (DYNALLOC macro)
- Having previously specified FREE=CLOSE on the allocation request (DYNALLOC macro or DD statement in the job's JCL).
- CATLG, UNCATLG, KEEP, PASS or DELETE a new or old SMS-managed data set, including when the SMS subsystem is not available.
- CATLG, UNCATLG, KEEP, PASS or DELETE a new or old VSAM data set, including when the SMS subsystem is not available.

The CATLG_ERR statement does not apply when the user unallocates a data set before step termination through the following methods:

- Dynamic deallocation (DYNALLOC macro)
- Having previously specified FREE=CLOSE on the allocation request (DYNALLOC macro or DD statement in the job's JCL).

FAILJOB(YES|NO)
Specifies whether the system is to terminate the job if one of the preceding errors occurs. When a job is terminated by **FAILJOB(YES)**, the termination is considered a post-execution error. Post-execution errors, which include but are not limited to **FAILJOB(YES)** terminations, are indicated by a 1 in the SMF30SYE bit in the SMF30STI field of the SMF30 subtype4 record.

Note:

1. The setting of the condition code is not affected.
2. The job is NOT abnormally ended, unless the step that encountered the error had itself previously abnormally ended. Terminated means that subsequent steps will not be taken.
3. The normal disposition for data sets is taken, unless the step that encountered the error had already abnormally ended, in which case the abnormal or conditional disposition is taken.

Default: NO

ERRORMSG(YES|NO)
Specifies whether the system is to issue an error message to the operator if one of the preceding errors occurs. When you specify ERRORMSG=YES, the system issues message IEF377I only if no other error messages have been issued. If FAILJOB(YES) is also specified, the system terminates the job, releases its resources, and issues message IEF378I.

Default: NO

VERIFY_VOL
Specifies the installation policy for verifying premounted or PASSed/RETAINed volumes on AutoSwitchable (AS) tape devices.

Note that the OPEN, FEOV and CLOSE macros allow the specification of a positioning parameter, and that the LEAVE option of these macros is treated the same as RETAIN®.

An AS Tape device that is connected, and possibly used, outside of this allocation's tape management scheme can be "stolen" for temporary use by allocation on a system outside this scheme and cause the volume status for the device to change, unbeknownst to this allocation's scheme. In this case, if the volume had been premounted, or is PASSed/RETAINed, this allocation scheme could then be causing inadvertent read/write activity on a volume. This can result in data loss or data integrity exposures.

If, for a given DD statement, MVS allocation selects an AS tape device and the UCB for that device shows any currently mounted volume to be the required volume for that DD, the system can optionally cause volume verification to occur when an OPEN is performed for that DD.

This becomes a significant consideration if:

1. AS Tape devices within the scope of this allocation's tape management scheme are also connected to, and usable by, systems outside of that scheme.
2. Allocation on this system might encounter volumes that are premounted onto an AS tape device or PASSed/RETAINed via JCL while mounted on an AS device.

Some examples of AS Tape devices that are connected outside the scope of allocation's tape management scheme include:

1. AS tape devices online to LPARs running other operating systems, such as zLinux or zVM.
2. AS tape devices online to at least one system in a sysplex and also online to one or more distinct, standalone systems.
3. AS tape devices online to systems in two or more distinct sysplexes.

POLICY(YES|NO)
Specifies whether the system is to perform volume verification of an apparently premounted or PASSed/RETAINed AS tape device at open time.

YES
Volume verification is to be done by OPEN for Standard, ISO/ANSI Version 1, or ISO/ANSI/FIPS Version 3 labeled tape volumes that are premounted or PASSed/RETAINed on an AS tape device. Both volume serial and tape position are verified.

NO No specific volume verification is to be done by OPEN on premounted or PASSed/RETAINed volumes on an AS tape device. Only select this policy option if it is certain there is no exposure to the "stolen" AS device for premounted or PASSed/RETAINed volumes as described in the previous scenarios. For example, specify POLICY(NO) if all systems within a sysplex are z/OS V1R2 or above, and tape devices that are defined as AS to systems within that sysplex are NOT shared with any systems, or sysplexes, outside of that sysplex.

Default: YES

SYSTEM
Specifies the system defaults.

IEFBR14_DELMIGDS(LEGACY|NORECALL)
Specifies the policy on whether to recall a migrated data set when you use an IEFBR14 JCL program with DD DISP=(x,DELETE) to delete the data set. The recall is, in most cases, unnecessary, as the data set is being deleted anyway.

LEGACY
Indicates that the system is to recall HSM-migrated data sets before deletion.

NORECALL
Indicates that the system can delete (through HSM HDELETE processing) the data set without first recalling the data set to the primary storage.

Default: LEGACY

TAPELIB_PREF(EQUAL|BYDEVICES)
Specifies the policy for balancing non-specific tape library requests (for example, scratch tape requests) across multiple tape libraries.

EQUAL
Indicates that for non-specific tape library requests, all tape libraries must be treated as equal, and receive an equal share of the requests.

BYDEVICES
Indicates that non-specific tape library requests must be balanced across all tape libraries according to the number of online tape devices in the tape library. Tape libraries with more online tape devices will receive more non-specific tape requests than libraries with fewer online devices when all devices have the same attributes.

This setting is useful when there are multiple tape libraries with different numbers of online tape devices. A tape library with, for example, 512 online tape devices would be able to handle more requests than a tape library with only 16 online tape devices. This setting allows MVS Device Allocation to better balance the requests across dissimilar libraries.

Default: EQUAL

REMIND_INTV(*xxx*)
Specifies the number of seconds for how often the message IEF882E and IEF883E are displayed, letting an operator know of an outstanding IEF238D, IEF433D, or IEF434D message. This interval indicates how many seconds between reminder messages, or 0 to disable them.

Value Range: 0 or 10-999

Default: 90

TEMPDSFORMAT(UNIQUE|INCLUDELABEL)
Specifies how the system generates data set names for temporary data sets that include '&&label' as the specified data set name. This statement affects only the data sets that specify DSN=&&mydsn, but not the data sets that do not specify DSNAME at all.

UNIQUE
Indicates that when the system processes JCL that includes temporary data sets with DSN=&&LABEL, the generated data

set name will be in the form "SYSyyddd.Thhmmss.RA000.jjobname.Rggnnnnn", which does not include the &&label specified in the JCL. All references to "&&mydsn" throughout the JCL correctly refer to the same data set. Using TEMPDSFORMAT=UNIQUE ensures that jobs with the same jobname running simultaneously do not create temporary data sets with the same names. See *z/OS MVS JCL Reference* for more information.

INCLUDELABEL
Indicates that when the system processes JCL that includes temporary data sets with DSN=&&LABEL, the generated data set name will include the &&label specified in the JCL. See *z/OS MVS JCL Reference* for more information.

Note: When this parameter is specified and the Job Entry Subsystem (JES) allows multiple jobs with the same job name to execute at the same time, jobs with the same name, executing simultaneously, might fail with a duplicate data set name error.

Default: INCLUDELABEL

VERIFY_UNCAT(FAIL|TRACK|MSGTRACK|LOGTRACK)
Specifies the policy for handling UNCATLG requests in JCL and dynamic allocation when the data set information is not retrieved from the catalog.

FAIL Fails the request. This option prevents a job from accidentally uncataloging a cataloged data set with the same name as the data set that is allocated.

TRACK
Allows the data set to be uncataloged, and tracks the request in the tracking facility. Specifying this option allows a job to uncatalog a cataloged data set with the same name as the data set that is allocated. This option is provided as a migration option to assist the installation in finding and correcting the existing JCL that might have been uncataloging data sets incorrectly.

Note: This option is provided for migration purposes. It may be removed from the system in the future.

The tracking facility must be active for UNCATLG requests to be tracked. Whenever a job attempts to uncatalog a data set for which the data set information is not retrieved from the catalog, the job is tracked with the Tracking information containing the string `IEFALC 01`, followed by the step name and the DD name.

- For batch allocated DDs, the program name contains `IEFIIC`.
- For dynamically allocated DDs, the program name is the offending program if it can be determined, or `SVC-099` if it cannot be determined.
- The VALUE field is unused, and will contain zeros.

Instances of the `IEFALC 01` event should be corrected by the installation rather than reported to IBM. For more information about the tracking facility, see *z/OS MVS Planning: Operations*.

MSGTRACK

Allows the data set to be uncataloged, tracks the request in the tracking facility, and issues a message. Specifying this option allows a job to uncatalog a cataloged data set with the same name as the data set that is allocated. This option is provided as a migration option to assist the installation in finding and correcting the existing JCL that might have been uncataloging data sets incorrectly.

Note: This option is provided for migration purposes. It may be removed from the system in the future.

The tracking facility must be active for UNCATLG requests to be tracked. Whenever a job attempts to uncatalog a data set for which the data set information is not retrieved from the catalog, the job is tracked with the Tracking information containing the string `IEFALC 01`, followed by the step name and the DD name.

- For batch allocated DDs, the program name contains `IEFIIC`.
- For dynamically allocated DDs, the program name is the offending program if it can be determined, or `SVC-099` if it cannot be determined.
- The VALUE field is unused, and will contain zeros.

Instances of the `IEFALC 01` event should be corrected by the installation, rather than reported to IBM. For more information about the tracking facility, see *z/OS MVS Planning: Operations*.

In addition to tracking the event, message IEF384I indicates that the data set is uncataloged, but the volume information is not retrieved from the catalog. This message informs the submitter of the job that the JCL should be corrected. For details, see *z/OS MVS System Messages, Vol 8 (IEF-IGD)*.

LOGTRACK

Allows the data set to be uncataloged, tracks the request in the tracking facility, and issues a message. Specifying this option allows a job to uncatalog a cataloged data set with the same name as the data set that is allocated. This option is provided for migration purposes to assist you in finding and correcting the existing JCL that might have been uncataloging data sets incorrectly. It might be removed from the system in the future.

The tracking facility must be active for UNCATLG requests to be tracked. Whenever a job attempts to uncatalog a data set for which the data set information is not retrieved from the catalog, the job is tracked with the Tracking information containing the string `IEFALC 01`, followed by the step name and the DD name.

- For batch allocated DDs, the program name contains `IEFIIC`.
- For dynamically allocated DDs, the program name is the offending program if it can be determined, or `SVC-099` if it cannot be determined.
- The VALUE field is unused, and will contain zeros.

Instances of the IEFALC 01 event should be corrected by the installation, rather than reported to IBM. For more information about the tracking facility, see *z/OS MVS Planning: Operations*.

In addition to tracking the event, message IEF384I indicates that the data set is uncataloged, but the volume information is not retrieved from the catalog. This message informs the submitter of the job that the JCL should be corrected. For details, see *z/OS MVS System Messages, Vol 8 (IEF-IGD)*.

Default: FAIL

MEMDSENQMGMT(ENABLE|DISABLE)
Specifies if the MEMDSENQMGMT feature is available for exploitation by jobs and subsystems.

ENABLE
Allows jobs and subsystems to use memory-based data set ENQ management for dynamically allocated data sets. Memory-based data set ENQ management is faster than the other option, SWA-based data set ENQ management, for jobs that allocate a large number of data sets. In addition to the parmlib setting, a job or subsystem that is to use the new management system must enable the feature using the IEFDDSRV service (for example, IEFDDSRV MODIFY TYPE=FEATURE,DSENQMGMT=MEMORY). Note that this feature makes the job or subsystem non-restartable through the checkpoint/restart interface.

DISABLE
Disables jobs and subsystems from using memory-based data set ENQ management for dynamically allocated data sets.

Default: DISABLE

BATCH_RCLMIGDS
Specifies how migrated data sets will be recalled.

SERIAL
The system allows the CATALOG LOCATE function to recall data sets on its behalf, resulting in serial recall processing.

PARALLEL
The system notes which data sets are migrated and recalls them in parallel; this may reduce job execution time.

Default: SERIAL

OPTCDB_SPLIT(EXPLICIT|CATALOG)
Indicates that the system should treat multi-volume data sets in system-managed tape libraries as separate DD statements only when DCB=OPTCD=B is specified with explicit volume serial numbers.

EXPLICIT|CATALOG
Specifies volume serial numbers or catalog.

Default: EXPLICIT

Chapter 5. ANTMIN00 (ANTMAIN control parameters)

Use the ANTMIN00 member to specify ANTMAIN control parameters. ANTMAIN is the DFSMS concurrent copy address space. For more information, see the documentation on the ANTMIN00 member in *z/OS DFSMS Advanced Copy Services*.

Chapter 6. ANTXIN00 (XRC services)

Use the ANTXIN00 member to tailor DFSMS extended remote copy (XRC) services. XRC services provide backup and recovery of data if a disaster occurs to your data center. For more information, see the documentation on the ANTXIN00 member in *z/OS DFSMS Advanced Copy Services*.

Chapter 7. APPCPMxx (define APPC/MVS configuration)

The APPCPMxx parmlib member contains a combination of statement types that define or modify the APPC/MVS configuration. These statements define APPC/MVS local LUs, indicate whether they are associated with a transaction scheduler, and name their associated administrative VSAM files. For more information about using APPC, see *z/OS MVS Planning: APPC/MVS Management* .

An installation can control its APPC/MVS configuration with different versions of the APPCPMxx parmlib member. One member might contain startup values while other members contain customized values.

To initialize the APPC address space and set up the APPC/MVS configuration, specify APPCPMxx with a START APPC command. To modify the APPC/MVS configuration after initialization, specify APPCPMxx with the SET APPC operator command, or use the SETAPPC command to modify the configuration without using parmlib. See *z/OS MVS System Commands* for further information regarding the usage of this command.

You can specify more than one APPCPMxx parmlib member on a START or a SET APPC command. If you specify more than one member on the START or SET command, APPC processes the statements in the order specified and creates a cumulative configuration. If you specify one or more members that do not exist, APPC/MVS issues an informational message. For more information about the START and SET commands, see *z/OS MVS System Commands*.

If you do not specify a parmlib member with the START command, the system searches for an APPCPM00 member, by default. If APPCPM00 does not exist, APPC/MVS issues an informational message.

Changing values

An installation can change configuration values established by an APPCPMxx parmlib member by creating another APPCPMxx parmlib member containing LUDEL statements that delete previous statements, or the parmlib member can contain LUADD and SIDEINFO statements with new parameter values to modify previous statements. Examples of parmlib members used to delete and modify configurations are in *z/OS MVS Planning: APPC/MVS Management*.

Note: When modifying previous statements, the parmlib statements have a cumulative effect, and any one parmlib member might not reflect the current configuration.

In addition to the SET APPC command, the SETAPPC command allows the installation to perform the same functions, but without having to update the APPC parmlib member. See *z/OS MVS System Commands* for further information.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for APPCPMxx

- Use column 1 through 71. Do not use columns 72-80, because the system ignores these columns.
- Comments may appear in columns 1-71 and must begin with /* and end with */.
- A statement must begin with a valid statement name followed by at least one blank.
- A statement ends with the beginning of the next valid statement name or End Of File (EOF).
- A statement can be continued even though there is no explicit continuation character.
- A statement contains only uppercase characters. Lower case is not accepted.
- Multiple occurrences of a statement are accepted.
- Keywords must be separated by valid delimiters. Valid delimiters are a comma, a blank, or column 71.
- Multiple occurrences of a delimiter are accepted but treated as one.
- Keyword values must be set off by parenthesis.
- Do not use blanks, commas, or comments in the middle of a parameter, between the parameter and the left parenthesis before the value, or in the middle of a value.

Syntax format of APPCPMxx

```
LUADD    ACBNAME (luname)
         [SCHED(schedname )|NOSCHED]
         [BASE            ]
         [PSTIMER(value)   ]
         [TPDATA(dsname)   ]
         [TPLEVEL{(SYSTEM)} ]
         [       {(GROUP)}  ]
         [       {(USER)}   ]
         [ALTLU  {(scheduler-supplied value)}   ]
         [USERVAR{(scheduler-supplied value)}   ]
         [GRNAME{(genericname)}                 ]
         [NQN | NONQN]

LUDEL    ACBNAME {(luname)}
         {PERSIST | NOPERSIST }

SIDEINFO DATASET(dsname)
```

IBM-supplied default for APPCPMxx

There is no default APPCPMxx parmlib member supplied by IBM. A sample parmlib member, APPCPMXX, is provided in SYS1.SAMPLIB.

Statements/parameters for APPCPMxx

The statement types for the APPCPMxx parmlib member are listed below and explained in more detail later.

LUADD

The **LUADD** statement defines a local APPC/MVS LU that is to be added to the APPC configuration. The LUADD statement contains:

- The LU name
- An indication of whether the LU is associated with a transaction scheduler
- The name of the transaction scheduler, if one is to be associated with this LU
- The amount of time the LU's sessions will persist in the event the LU becomes unavailable
- The TP profile file associated with the LU
- The level of TP profile from which the LU starts to search
- Optional values to be passed to an alternative transaction scheduler, or to any other member of the APPC XCF group, such as an APPC/MVS server
- A VTAM generic resource name to associate with the LU
- An indication of whether the LU is enabled to support network-qualified names for its partner LUs.

Each LU managed by APPC/MVS must be defined with an LUADD statement.

When an installation uses the ASCH transaction scheduler exclusively, only one LU is required. If other transaction schedulers are used, each scheduler requires a separate LU. An installation might choose to define additional LUs to isolate TPs for security or testing.

An installation can also define LUs that are not associated with transaction schedulers. These LUs handle work that is processed by APPC/MVS servers, rather than scheduled by a transaction scheduler. Such LUs are indicated by coding the NOSCHED keyword on LUADD. Installations can also use NOSCHED LUs when they want to flow outbound allocate requests without having a transaction scheduler active. (Note that APPC/MVS servers can also run under LUs that are associated with transaction schedulers.)

You can modify an LU by overriding a previous LUADD statement with another LUADD statement that names the existing LU and changes the parameters to be modified. The only parameters you cannot modify with an overriding LUADD are the SCHED, NOSCHED, ALTLU, USERVAR, GRNAME, NQN and NONQN parameters. To change these parameters, you must first delete the LU with an LUDEL statement and then re-identify the LU with a new LUADD that changes the parameters.

Example: The following example defines LU MVSLU01 to be associated with the transaction scheduler provided with APPC/MVS.

```
LUADD
  ACBNAME(MVSLU01)
  SCHED(ASCH)
  TPDATA(SYS1.APPCTP)
  TPLEVEL(USER)
```

ACBNAME(luname)

The required name of the LU that APPC/MVS is to remove. If this LU was defined to VTAM, its association with VTAM is terminated after active conversations end.

Value range: A one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@,$,#) and must begin with an alphabetic or national character.

Note: The SNA LU 6.2 architecture defines a network-qualified LU name to be up to 17 bytes in length and in the form *network_id.network_LU_name*, where *network_id* is the optional 8-byte id of the network and *network_LU_name* is the

8-byte local LU name. SAA CPI Communications allows the full 17-byte network-qualified LU name. However, for the ACBNAME keyword, specify only the 8-byte local LU name.

Default: None, this parameter is required.

SCHED(ASCH|schedname)
NOSCHED

An optional parameter that indicates whether the LU is to be associated with a transaction scheduler. LUs associated with a transaction scheduler cannot become active until that scheduler identifies itself to APPC/MVS. LUs not associated with a transaction scheduler become active as soon as APPC/MVS becomes active.

SCHED indicates that the LU is associated with a transaction scheduler. *schedname* must match the name the transaction scheduler specifies when it calls the Identify service. For more information about the Identify service and its scheduler_name parameter, see *z/OS MVS System Messages, Vol 3 (ASB-BPX)*.

NOSCHED indicates that the LU is not to be associated with a scheduler. When NOSCHED is specified, the LU becomes active as soon as APPC/MVS becomes active. Installations can use NOSCHED LUs to isolate work from schedulers when the work is to be processed by APPC/MVS servers. Installations can also use NOSCHED LUs to flow outbound allocate requests without having a transaction scheduler active.

Value Range: For *schedname*, the value is a one- to eight-byte character string and each character must be an uppercase letter (A-Z) or a numeral (0-9).

Note: SCHED and NOSCHED are mutually exclusive keywords; you cannot specify both SCHED and NOSCHED in a single LUADD statement. Doing so causes the system to ignore the statement and issue message ATB041I to the system operator.

Default: When you omit both SCHED(schedname) and NOSCHED, the default is SCHED(ASCH).

BASE

An optional parameter that designates the LU as the base LU. Base LUs are default LUs assigned to handle outbound work. A base LU can be the default LU associated with a particular transaction scheduler or a NOSCHED LU.

When a NOSCHED LU is defined with the BASE option, the LU becomes the *system base LU*. That means the LU is to be the default LU used for outbound allocate requests from MVS programs, such as batch jobs, TSO/E users, started tasks, and other work requests that attempt to enter the network without being associated with a scheduler or an LU.

Example: The following example defines a NOSCHED LU, MVSLU02, to be the system base LU.

```
LUADD
  ACBNAME(MVSLU02)
  NOSCHED
  BASE
  TPDATA(SYS1.APPCTEST)
  TPLEVEL(SYSTEM)
```

If you do not define a NOSCHED LU as a base LU, the base LU defined for the APPC/MVS transaction scheduler (ASCH) becomes the system base LU. If no system base LU exists, APPC/MVS rejects conversations allocated by MVS programs that are not associated with a scheduler or an LU.

IBM suggests that you define one LU per transaction scheduler as the base LU for the scheduler. In addition, define a NOSCHED LU as the system base LU if you want to allow outbound requests from the system when no transaction schedulers are active.

When more than one LU is defined as the base LU, the one most recently defined is the base.

PSTIMER(value)

An optional parameter that sets the maximum amount of time for which the LU's sessions **persist** (are maintained) during interruptions in APPC/MVS or a transaction scheduler's service.

When you specify a valid value other than NONE, the LU's sessions persist when the APPC address space is canceled, forced, terminated, or automatically restarted. The sessions also persist during interruptions in scheduler service. Any conversations that were active at the time of the interruption are lost. When APPC service is resumed, the conversation partners can re-establish these conversations, if desired.

Sessions do not persist in the event the LU is deleted.

Value Range:

- 0 or INDEFINITE (Sessions persist indefinitely)
- 1 - 86400 (Number of seconds the sessions can persist)
- NONE (Sessions are not to persist)

Default: NONE

TPDATA(dsname)

An optional parameter that specifies the name of the VSAM key-sequenced data set that contains TP profiles, along with an optional data base token for the LU. The data base token is used for verifying access authority to TP profiles. If this LU is a NOSCHED LU, APPC/MVS uses only the data set's data base token, if any. The data set specified on TPDATA must be cataloged in either a user catalog or the master catalog.

Value range: Up to 44 characters in length consisting of one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@,$,#) and must begin with an alphabetic or national character.

Default: SYS1.APPCTP

TPLEVEL({SYSTEM|GROUP|USER})

An optional parameter that identifies the level of TP profiles for which the LU searches in response to an inbound allocate request. TPLEVEL limits the search to the levels desired.

Each TP can have different levels of TP profiles with scheduling characteristics associated with a user, a group of users, or all users (system). The TPLEVEL parameter tells the LU which of those levels of TP profile to search.

Value range:

SYSTEM means that the LU searches for system-level TP profiles only (NOT for a specific user or group of users).

GROUP means that the LU searches for TP profiles associated with (1) a specific group of users and (2) system-level TP profiles, in that order.

USER means that the LU searches for TP profiles associated with (1) a specific user, (2) a group of users, and (3) system-level TP profiles, in that order.

Note: If you specify NOSCHED, TPLEVEL must be SYSTEM. Also, TP profile entries in the data set specified in TPDATA are not used for NOSCHED LUs — only the data base token is used.

Default: SYSTEM

ALTLU(scheduler-supplied value)
This parameter allows optional, installation-supplied data to be passed to a member of the APPC XCF group, such as an alternative transaction scheduler or an APPC/MVS server. If specified, the data is passed to the APPC XCF group member at the activation and deactivation of the associated LU. For information about the APPC XCF group, see *z/OS MVS System Messages, Vol 3 (ASB-BPX)*.

Value Range: A one- to eight-byte character string of uppercase letters A through Z, numerics 0-9, or national characters (@, $, #), with the exception that the first character cannot be numeric (0-9).

Default: None

USERVAR(scheduler-supplied value)
This parameter allows optional, installation-supplied data to be passed to a member of the APPC XCF group, such as an alternative transaction scheduler or an APPC/MVS server.

If specified, the data is passed to the APPC XCF group member at the activation and deactivation of the associated LU. For information about the APPC XCF group, see *z/OS MVS System Messages, Vol 3 (ASB-BPX)*.

Value Range: A one- to eight-byte character string of uppercase letters A through Z, numerics 0-9, or national characters (@, $, #), with the exception that the first character cannot be numeric (0-9).

Default: None

GRNAME(genericname)
This optional parameter specifies a VTAM generic resource name to be associated with the LU. The LU may be one of multiple LUs in the same generic resource group, represented by *genericname*. This parameter cannot be dynamically modified or added to an existing LU definition.

See *z/OS MVS Planning: APPC/MVS Management* for advice and restrictions about selecting a generic resource name, and deciding which LUs should become members of a generic resource group.

Value range: A one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@, $, #) and must begin with an alphabetic or national character.

Default: None. If the GRNAME parameter is not specified, the LU is activated but is not part of a generic resource group.

NQN
NONQN
An optional parameter that specifies whether the APPC/MVS LU is enabled to use a network-qualified partner LU name when first allocating outbound conversations. If you specify NQN, APPC/MVS uses the 17-byte network-qualified LU name when both verifying the partner LU, and sending the outbound Allocate request to the partner LU. If you specify NONQN (or allow the system to use the default), APPC/MVS uses the entire name when verifying the partner, but only the 8-byte network-LU-name portion when sending the outbound Allocate request.

See *z/OS MVS Planning: APPC/MVS Management* for the requirements for enabling APPC/MVS LUs to support network-qualified names.

Default: NONQN

LUDEL

The **LUDEL** statement deletes a local APPC/MVS LU from the APPC configuration. One LUDEL statement must be specified for each LU to be deleted. The LUDEL statement contains:

- The LU name
- An indication of whether APPC/MVS should keep all persistent sessions active between this LU and all of its partners

When an LUDEL statement is processed, incoming allocation requests to the named LU are rejected; however, all active conversations are allowed to continue until completed. The LU is removed only after all active conversations have ended.

ACBNAME(luname)

The required name of the LU that APPC/MVS is to remove. If this LU was defined to VTAM, its association with VTAM is terminated after active conversations end.

Value Range: A one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@,$,#) and must begin with an alphabetic or national character.

Note: The SNA LU 6.2 architecture defines a network-qualified LU name to be up to 17 bytes in length and in the form *network_id.network_LU_name*, where *network_id* is the optional 8-byte id of the network and *network_LU_name* is the 8-byte local LU name. SAA CPI Communications allows the full 17-byte network-qualified LU name. However, for the ACBNAME keyword, specify only the 8-byte local LU name.

Default: None; this parameter is required.

PERSIST | NOPERSIST

An optional parameter that specifies whether APPC/MVS will deactivate all sessions between this LU and its partners when the LU is deleted. If you specify PERSIST, and if the LU was previously enabled to support persistent sessions with the PSTIMER keyword on the LUADD statement, APPC/MVS does not deactivate sessions between the LU and its partners. VTAM keeps these sessions active as long as the LU is re-added to the APPC configuration on the same z/OS image within the PSTIMER time limit (single-node persistent sessions) or in any z/OS image in the sysplex within the PSTIMER time limit (multi-node persistent sessions). See *z/OS MVS Planning: APPC/MVS Management* for further information. If you specify NOPERSIST (or allow the system to use the default), APPC/MVS deactivates all sessions between this LU and its partners when the LU is deleted.

Default: NOPERSIST

SIDEINFO

The **SIDEINFO** statement names the VSAM key sequenced data set that contains side information. Only one side information file is allowed per MVS system.

DATASET(dsname)

An optional parameter that specifies the name of the VSAM key sequenced data set that contains side information. The file must be cataloged in either a user catalog or the master catalog.

Value range: Up to 44 characters in length consisting of one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@,$,#) and must begin with an alphabetic or national character.

Default: SYS1.APPCSI

Response to errors in APPCPMxx

If a syntax error is found in an APPCPMxx parmlib member, a message is sent to the originating console and the statement(s) in error are rejected. All syntactically valid statements are processed normally.

When such an error is reported, the system programmer should create a parmlib member containing corrected statements and retry the operation.

Chapter 8. ASCHPMxx (APPC/MVS transaction scheduler)

The ASCHPMxx parmlib member contains scheduling information for the ASCH transaction scheduler. The statements define classes of transaction initiators and provide default scheduling information when it is missing from a TP profile.

For more information about using APPC, see *z/OS MVS Planning: APPC/MVS Management*.

An installation can control scheduling characteristics with different versions of the ASCHPMxx parmlib member. One member might contain startup values and other members contain customized values.

To start the ASCH address space and set up classes and defaults for the ASCH transaction scheduler, specify ASCHPMxx with a START ASCH command. To modify classes and defaults after initialization, specify ASCHPMxx with the SET ASCH operator command. More than one ASCHPMxx parmlib member can be specified on a START or a SET ASCH command. If you specify more than one member on the START or SET command, APPC processes the statements in the order specified and creates a cumulative configuration. For more information about the START and SET commands, see *z/OS MVS System Commands*.

If you do not specify a parmlib member with the START command, the system uses the default member, ASCHPM00. If ASCHPM00 does not exist, APPC/MVS issues an informational message.

Changing values

An installation can change scheduling information established by an ASCHPMxx parmlib member by creating another ASCHPMxx parmlib member containing CLASSDEL statements that delete previous statements, or the parmlib member can reissue statements with new parameter values to modify previous statements.

Examples of deleting and modifying statements are in *z/OS MVS Planning: APPC/MVS Management*.

Attention: When modifying previous statements, the parmlib statements have a cumulative effect, and any one parmlib member might not contain the current scheduling characteristics.

Default values

When parmlib statements are initially specified, omitted parameters receive default values. When, however, parmlib statements are respecified, omitted parameters in the CLASSADD statement assume the defaults, but omitted parameters on the OPTIONS and TPDEFAULT statements do not assume the defaults.

For example, in the CLASSADD statement, the MAX parameter has a default value of 1. When you initially define a class with the START ASCH command and omit MAX, the class defaults to the maximum of 1 initiator. If, however, you initially set MAX to 10, when the class is modified with a SET ASCH command and MAX is not specified, the maximum of 10 is overridden with the maximum of 1 default.

Support for system symbols

You can specify static symbolics in ASCHPMxx. For information about how to use static symbolics, see Chapter 2, “Sharing parmlib definitions,” on page 33.

Parameter in IEASYSxx (or supplied by the operator)

None

Syntax rules for ASCHPMxx

- Use column 1 through 71. Do not use columns 72-80, because the system ignores these columns.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- A statement type consists of 1-10 characters.
- A statement must begin with a valid statement type followed by at least one blank.
- A statement ends with the beginning of the next valid statement type or End Of File (EOF).
- A statement can be continued even though there is no explicit continuation character.
- Multiple occurrences of a statement type are accepted.
- A statement contains only uppercase characters. Lower case is not accepted.
- Operands must be separated by valid delimiters. Valid delimiters are a comma, a blank, or column 71.
- Multiple occurrences of a delimiter are accepted but treated as one.
- Keyword values must be set off by parenthesis.
- Do not use blanks, commas, or comments in the middle of a parameter, between the parameter and the left parenthesis before the value, or in the middle of a value.

Syntax format of ASCHPMxx

```
CLASSADD   CLASSNAME(classname)
           [MAX(nnnnn)        ]
           [MIN(nnnnn)        ]
           [RESPGOAL(nnnnnnnn)]
           [MSGLIMIT(nnnnn)   ]

CLASSDEL   CLASSNAME(classname)
           WORKQ    {(PURGE|DRAIN)}

OPTIONS    DEFAULT(classname)
           SUBSYS(ssname)

TPDEFAULT  REGION   {(nnnnK)}
                    {(nnnnM)}

           TIME     {(NOLIMIT)        }
                    {(minutes,seconds)}

           MSGLEVEL(1,n)
           OUTCLASS(n)
```

IBM-supplied default for ASCHPMxx

There is no default ASCHPMxx parmlib member. A sample parmlib member, ASCHPMxx, is provided in SYS1.SAMPLIB.

Statements/parameters for ASCHPMxx

The statements for the ASCHPMxx parmlib member are listed below and explained in more detail later.

CLASSADD

The **CLASSADD** statement identifies a class of transaction initiators to the APPC/MVS transaction scheduler configuration. The CLASSADD contains the class name, the maximum and minimum number of transaction initiators to assign to the class, the response time goal, and the message limit. Each class for the APPC transaction scheduler must be defined with a CLASSADD statement.

At least one class definition is required for work assigned to the APPC/MVS transaction scheduler. If work comes in for a class that is not defined by a CLASSADD statement, the work is rejected.

You can modify a class by overriding a previous CLASSADD statement with another CLASSADD statement that names the existing class and changes the parameter values to be modified. When more than one CLASSADD statement exists for the same class, the most recently processed statement is in effect.

CLASSNAME(classname)

A required parameter that specifies the name of a class of transaction initiators. When a class name is specified in a TP profile, the APPC/MVS transaction scheduler assigns the TP to run in that class. If no class name is specified in a TP profile, the default class from the OPTIONS statement is used. If the class in the TP profile does not match any class name specified on a CLASSNAME statement and there is no default class, the TP cannot run.

SRM uses the class of transaction initiators as one of the items to determine dispatching priority.

Value range: A one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@, $, #) and must begin with an alphabetic or national character.

Default: None

MAX(nnnnn)

An optional parameter that specifies the maximum number of transaction initiators that are allowed for a particular class of transaction initiators. After this limit is reached, no new address spaces are created and incoming requests are queued to wait until existing initiator address spaces become available. The value should not exceed the maximum number of address spaces allowed by your installation, and you should be aware of competing products on the system that will also require address spaces.

If this value is too high, system resources that might be needed elsewhere are not available. If this value is set too low, transaction programs will wait on the queue until an existing initiator becomes available, jeopardizing the response time goal for the class. To optimize performance while still meeting resource goals, specify a value that is a percentage of the total number of possible transactions running in the class at a time. You can

determine the exact percentage after considering the number of transaction initiators available for all the classes, and experimenting with various values until one meets performance requirements.

Value Range: 1 - 64000

Default: 1

MIN(nnnnn)
An optional parameter that specifies the minimum number of transaction initiators that are brought up for a particular class of transaction initiators when the ASCH address space starts or changes dynamically with a SET command. If the MIN value exceeds the MAX value, the MIN value is set to the MAX value. The number of transaction initiators available in each class never goes below the MIN number. In setting this value, consider the type of transaction programs that will run in this class. You may want certain classes to have initiators that are always available, while others classes require fewer initiators.

If this value is set too high, system resources that may be needed elsewhere are left idle. If this value is too small, the scheduler may waste time and resources creating and deleting transaction initiators. To optimize performance while still meeting resource goals, specify a value that is a percentage of the total number of possible transactions running in the class at a time. You can determine the exact percentage after considering the number of transaction initiators available for all the classes, and experimenting with various values until one meets performance requirements.

Value range: 0 - 64000

Default: 0

RESPGOAL(nnnnnnnn)
An optional parameter that specifies the response time goal for TPs executing within this class. The RESPGOAL value is the total time in seconds that an installation wants to allow for queueing and running a transaction in this class. To meet this goal, APPC/MVS can create additional transaction initiators for the class until the MAX value is reached.

To determine how well response time is being met, you can see transaction run times in the RMF Monitor I Workload Activity Report. TP run times can also be obtained via SMF reports.

The response time goal should be set by determining the average run time for transactions in a class and adding in an allowable queue delay time. The additional queue delay time provides the APPC transaction scheduler some control to attempt to optimize overall system overhead associated with creating and deleting transaction initiators.

Value range: 0.000001 - 31536000 seconds (365 days)

Default: 1

MSGLIMIT(nnnnn)
An optional parameter that, for all TPs running within the class, specifies the maximum number of messages written to the TP's message log each time the TP runs. For more information about selecting a MSGLIMIT value, see the topic about logging transaction program processing in *z/OS MVS Planning: APPC/MVS Management*.

Value range: 1-15000 messages

Default: 500 messages

CLASSDEL

The **CLASSDEL** statement deletes a class of transaction initiators from the APPC/MVS transaction scheduler configuration. One CLASSDEL statement must be specified for each class of transaction initiators that is to be deleted.

CLASSNAME(classname)

A required parameter that specifies the name of an existing class of transaction initiators to be deleted.

Value Range: A one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@,$,#) and must begin with an alphabetic or national character.

Default: None. A value must be specified.

WORKQ({PURGE|DRAIN})

A parameter that specifies whether a class work queue is to be drained or purged when the work class is deleted.

When you specify PURGE on the WORKQ keyword, all work that is queued up for this class is rejected. An error message is returned to the issuers of the ALLOCATE request.

However, work that is currently running is allowed to complete its processing.

When you specify DRAIN on the WORKQ keyword, new work for the class is rejected; however, work that was assigned before the class was deleted is allowed to finish.

Value range: PURGE or DRAIN

Default: DRAIN.

OPTIONS

The **OPTIONS** statement defines class options for the APPC transaction scheduler. Options include naming a default class to be used when a TP profile does not specify a class, and naming a subsystem to which transaction initiators are assigned.

You can modify these options by overriding a previous OPTIONS statement with another OPTIONS statement that changes parameter values to be modified. When more than one OPTIONS statement exists, the most recently processed parameter values are in effect.

DEFAULT(classname)

An optional parameter that specifies the default class of transaction initiators in which to run a TP when a class name is not specified in a TP profile.

If the TP profile does not specify a class name, and there is no default defined by this parameter, the request to run the TP is denied. If the DEFAULT parameter names a class that does not exist, an error message is displayed on the console.

To delete the previously specified default class, specify a null value for DEFAULT. The format for a null value is: DEFAULT()

Value range: Null or a one- to eight-byte character string of uppercase letters A through Z, numerals 0-9, national characters (@, $, #) and must begin with an alphabetic or national character.

Default: None.

SUBSYS(ssname)
An optional parameter that specifies the name of the subsystem under which all newly created APPC/MVS transaction initiators are started. This value applies to all defined classes of transaction initiators.

If you specify a JES subsystem, it must be defined to the system in an IEFSSNxx parmlib member.

When running APPC transaction programs that do not require JES services (such as SYSOUT processing), you can specify SUBSYS(MSTR). Also, if you are using a version of JES2 lower than 4.2.0, or a version of JES3 lower than 4.2.1., specify SUBSYS(MSTR) because APPC/MVS does not support JES services at these lower levels.

To delete the previously specified subsystem, specify a null value for SUBSYS. The format for a null value is: SUBSYS()

Value range: Null or one- to four-characters. The name must begin with an alphabetic or national character (@, $, #), and the remaining characters (if any) can be alphanumeric or national.

Default: Primary JES subsystem.

TPDEFAULT
The **TPDEFAULT** statement supplies default information when it is missing from the scheduler JCL section of a TP profile scheduled by the APPC/MVS transaction scheduler. Defaults include the region size of the TP, time limit for running the TP, the level of messages to appear in the message log, and the output class for TP SYSOUT.

You can modify defaults by overriding a previous TPDEFAULT statement with another TPDEFAULT that changes parameter values to be modified. When more than one TPDEFAULT statement exists, the most recently processed parameter values are in effect.

REGION({nnnnK|nnnnM})
An optional parameter that specifies the default region size assigned to TP profiles that do not specify a region size.

Value range: 0K - 9999K or 0M - 2047M

Default: 2M

TIME(NOLIMIT|mmmm[,ss])
An optional parameter that specifies the default step time limit assigned to TPs that do not specify a time limit. The time limit is of the following format:

```
TIME(minutes,seconds) or TIME(NOLIMIT) or TIME(minutes) or
TIME(,seconds)
```

When time is in minutes only, do not include the comma. When time is in seconds only, include the comma before the seconds. TIME(NOLIMIT) is equivalent to TIME(1440).

Value range:

- minutes: 1 - 1440

- seconds: 1 - 59
- NOLIMIT

Default: TIME(1440)

MSGLEVEL(1,n)
An optional parameter that specifies the level of messages generated for TPs that do not specify MSGLEVEL in the form MSGLEVEL(1,n). This MSGLEVEL is similar to the JCL MSGLEVEL parameter, but it behaves differently in APPC/MVS. TP profile JCL is processed in two phases, which are reflected in the two sub-parameters (1,n).

The first sub-parameter 1 controls the listing of statements, procedure statements, and substitution JCL messages, which occur during TP profile add and modify processing. These statements and messages are listed in the APPC administration utility output file (SYSPRINT). The value for this parameter must be 1.

The second sub-parameter n controls the generation of messages that occur when the TP profile is accessed to run a TP. If you specify 0 for this sub-parameter, allocation/termination messages are generated only if the TP abnormally terminates. If you specify 1 for this sub-parameter, allocation/termination messages are always generated.

This sub-parameter works in conjunction with the KEEP_MESSAGE_LOG parameter of the TP profile. Messages are generated according to the MSGLEVEL parameters and are written to the TP message log according to the KEEP_MESSAGE_LOG parameter. If the value of KEEP_MESSAGE_LOG is *error* and MSGLEVEL is (1,0) or (1,1), messages are written to the log on error. If the value of KEEP_MESSAGE_LOG is *always* and MSGLEVEL is (1,1), messages are always written to the TP message log. (Note when KEEP_MESSAGE_LOG is *always* and MSGLEVEL is (1,0), allocation/termination messages are generated only when the TP abnormally terminates.) If the value of KEEP_MESSAGE_LOG is *never*, no messages are written regardless of the value of MSGLEVEL.

The value for the first sub-parameter must be 1, which results in the following format:

```
MSGLEVEL=(1,messages)
```

Value range for *messages*:

- (0) Allocation/termination messages are generated only if the TP abnormally terminates.
- (1) Allocation/termination messages are always generated.

Default: MSGLEVEL(1,0)

OUTCLASS(n)
An optional parameter that specifies the default value for MSGCLASS when MSGCLASS is not specified on the JOB statement in the TP profile.

Note: In APPC/MVS, MSGCLASS does not assign the output class for a job log (TP message log). However, MSGCLASS can have an effect on SYSOUT processing. For more information, see the MSGCLASS keyword in *z/OS MVS JCL Reference*.

Value range: A - Z, 0 - 9

Default: A

Chapter 9. AUTORxx (auto-reply policy specifications)

Use the parmlib member AUTORxx to activate auto-reply processing on a system. The member AUTOR00 contains the auto-reply policy suggested by IBM. You can modify the member (which is not recommended), or define another AUTORxx member to customize the auto-reply policy. The member AUTOR00 also contains comments that provide the message text for each WTOR and the rule used to select the WTOR.

Parameter in IEASYSxx (or supplied by the operator)

```
AUTOR=(xx,yy,...)
```

Syntax rules for AUTORxx

The following syntax rules apply to the AUTORxx parmlib member:

- Data is specified in columns 1-71.
- Comments can start in any column and can span lines, and must start with /* and end with */. Comments may appear in any place where a blank is accepted, except within quoted strings.
- Generally, syntax errors cause the auto-reply changes to be rejected, while in the following cases, syntax errors are ignored and allow the changes to become active:
  1. The maximum number of message IDs is reached.
  2. Duplicate message IDs are specified.
  3. NOTIFYMSGS is specified in different members.

Syntax format of AUTORxx

```
[NOTIFYMSGS({HC|CONSOLE})]
[MSGID(']msgid[']) {NOAUTORREPLY}
                   {DELAY(nnn{M|S}) REPLY([']replytext['] [,[']replytext[']...])[RATELIMIT(nnn)]}]
```

Syntax example of AUTORxx

```
notifymsgs(hc)
/*********************************************************************/
/* $HASP811 REPLY Y TO CONTINUE OR N TO TERMINATE START PROCESSING  */
   Msgid(?HASP811)   Delay(30S) Reply(Y)
/*********************************************************************/
/* ANTU2220D "READY FOR FLASHCOPY. REPLY 'I' TO INITIATE, 'C' TO    */
/*           CANCEL"                                                 */
   Msgid(ANTU2220D)  Delay(60S) Reply(C)
```

To set the rate limit value to 30 replied WTORs per second per message ID, you can specify RATELIMIT(030) on the MSGID statement.

```
NOTIFYMSGS(HC)
/*********************************************************************/
/* ABD123D IS THE SKY BLUE?                                          */
/* REPLY 'Y'-YES OR 'N'-NO                                           */
Msgid(ABD123D) Delay(0S) Reply(Y) RateLimit(030)
/*********************************************************************/
/* ABD456D READY FOR WORK. REPLY 'B' TO BEGIN, 'C' TO CANCEL         */
Msgid(ABD456D) Delay(60S) Reply(C)
```

IBM-supplied default for AUTORxx

The IBM-supplied default for AUTORxx is AUTOR00.

Statements/parameters for AUTORxx

NOTIFYMSGS

May only appear once in a parmlib member. If multiple members are specified in the SET AUTOR= command, the first NOTIFYMSGS value is accepted. The other specifications are ignored.

HC Indicates that the auto-reply notification messages (CNZ2605I, CNZ2606I and CNZ2608I) only appear in the hardcopy log. If NOTIFYMSGS is not specified, HC is the default value.

CONSOLE
Indicates that the auto-reply notification messages (CNZ2605I, CNZ2606I and CNZ2608I) are displayed on consoles receiving routing codes 2 (operator information) or 10 (system programmer information) and also appear in the hardcopy log.

MSGID()

The first keyword of the message definition.

msgid Must be in the range of 1 to 10 characters. The message ID must be enclosed in quotes if it contains non-alphanumeric characters, like equal signs, parenthesis, etc. The quotes do not count as part of the message ID. If enclosed in quotes, the *msgid* is not converted to uppercase.

Wildcards are partially supported in the *msgid*:

- The question mark "?" is supported, because some messages (e.g., JES2 messages) start with an installation-specified character (not always $). For instance, ?HASP1234 is allowed.
- The asterisk "*" is not treated as a wildcard in a *msgid*. If a *msgid* contains both a question mark and an asterisk, the message definition is rejected.
- Multiple question marks in the *msgid* is supported. For instance, eight question marks ???????? would match on any eight character message ID.

Note: The maximum number of message IDs that do not contain wildcards is limited to 10,413. The maximum number of message IDs that contain wildcards is limited to 1,500.

NOAUTORREPLY
Can be specified to cause subsequent MSGID() specifications of

this message ID (in this member or subsequent members) to be ignored. With the NOAUTOREPLY option, you can have your own AUTORxx member and use it to remove messages specified in the AUTOR00 member without actually removing the AUTOR00 statements. No auto-reply processing is performed for this message ID.

DELAY(*nnn***{M|S})**
Can appear in any order and on different lines from REPLY() and MSGID(). *nnn* is the minimum amount of time, in minutes (M) or seconds (S), to delay after the WTOR is issued but before the system issues a reply. Only three digits are supported, which gives a limit of 999 minutes (16.65 hours) or 999 seconds (16.65 minutes). The delay value of 0 is supported, but specifying the value of 0 can prevent automation or an operator from providing a reply.

REPLY(['*replytext***'] [,['***replytext***']...])**
Can appear in any order and on different lines from DELAY() and MSGID(). A null reply can be provided by specifying REPLY(' ').

replytext is limited to 64 characters. If the reply contains blanks, non-alphanumeric characters, or is not to be folded to uppercase, enclose the reply in single quotes. The single quotes count towards the total of 64 characters. If the reply is too long for the WTOR requestor, the notification message CNZ2608I is issued when the WTOR is issued. Where message CNZ2608I appears depends on the setting of NOTIFYMSGS. It is displayed on consoles receiving routing codes 2 (operator information) or 10 (system programmer information), and also appears in the hardcopy log.

- To specify quotes in the reply, specify two contiguous single quotes, like REPLY('That''s all folks.'). The reply may contain system symbolics that are resolved when the parmlib member is processed. If you want to have the symbolic resolved when the REPLY command is issued, specify two contiguous ampersands, like REPLY('Here is the ','&&SYSNAME','.'). The symbolic must be in uppercase and enclosed in quotes along with the two contiguous ampersands &&.
- If the auto-reply policy entry contains a symbolic (like &&name) that is to be resolved when the reply is issued, auto-reply is unable to validate that the reply length would be acceptable by the WTOR issuer. If the reply with the symbolic resolved is longer than what the WTOR issuer expects, when auto-reply processing issues the reply, the REPLY command rejects the reply with:

  ```
  IEE700I REPLY xxxx IGNORED; REPLY TOO LONG FOR REQUESTOR
  ```

  Auto-reply no longer monitors this WTOR, and the response to the D AUTOR,WTORS command (CNZ2604I) indicates that the WTOR has been replied to.
- Do not have the reply text span parmlib source lines. Otherwise, the use of symbolics might cause problems. For example, if you specify

```
REPLY('system=&SYSNAME.,option1
,option2')
```

and &SYSNAME resolves to SY1, the reply text is

```
'system=SY1,option1     ,option2'
```

To prevent the substitution from causing problems, split the reply text into separate strings. A correct specification is:

```
REPLY('system=&SYSNAME.',
            ',option1,option2')
```

and the reply text is

```
'system=SY1,option1,option2'
```

- Auto-reply processing can concatenate the strings specified in REPLY. For example, if you specify REPLY('String1'), the quotes are part of the reply 'String1'. If you specify REPLY('String1','String2'), auto-reply processing concatenates the strings and the reply is 'String1String2'. The leading and trailing quotes are part of the reply.

  You can also have a mixture of strings. By specifying

  ```
  REPLY(Word1,'String1',Word2,'String2')
  ```

  you will get a reply of

  ```
  WORD1'String1'WORD2'String2'
  ```

- The REPLY command accepts only one quoted string if the first character of the reply is a quote. For example:

  ```
  'This is the reply' but not this
  ```

  The REPLY command will only pass

  ```
  This is the reply
  ```

  to the WTOR issuer. The WTOR issuer does not see

  ```
  but not this
  ```

  Although it is possible that auto-reply can build this type of string, you should be aware that it may not yield the results you wanted.

RATELIMIT(*nnn*)

nnn is the rate limit value that can be set to indicate (for a specific message ID) the number of replies that can be issued per second. After 1 second has passed, the message count is reset. Only three digits are supported, which gives a limit of 999 replies per second for the specified message ID. The rate value of 001 is the lowest value supported. The default value for rate limit has been set to 020. RATELIMIT is only valid when DELAY is set to 0; otherwise it will be rejected.

Note:

1. Both DELAY() and REPLY() are required if NOAUTOREPLY is not specified.
2. Duplicate DELAY() or REPLY() keywords for an MSGID() are rejected and cause the MSGID() specification to be rejected.

3. Specification of the same message ID in different message definitions results in the first definition being used.

Chapter 10. AXRxx (system REXX options)

The AXRxx PARMLIB member specifies options for system REXX. SYS1.SAMPLIB contains a copy of the AXRxx member that you can copy to SYS1.PARMLIB and modify.

Parameter in IEASYSxx (or supplied by the operator)

```
AXR={aa         }
     {(aa,bb,...)}
```

The 2-character identifier (aa, bb, and so forth) is appended to AXR to identify the AXRxx member of SYS1.PARMLIB. If AXR= is not provided in IEASYSxx, AXR00 is used.

Syntax rules for AXRxx

- Data must be contained in columns 1-71
- Comments may appear in columns 1-71 and must be delimited with /* and */
- If a parameter is specified multiple times, the first valid instance is accepted
- System symbolics may be used anywhere.

IBM-supplied default for AXRxx

The IBM-supplied default for AXRxx is AXR00, and the AXRxx member is optional.

Statements/parameters for AXRxx

```
CPF('<cpf value>',SYSTEM|SYSPLEX)
AXRUSER(<Userid>)
REXXLIB ADD DSN(<Rexxlib data set name>) VOL(<volser>)
or
REXXLIB ADD DSNAME(<Rexxlib data set name>)
MaxWorkerTasks(nn)
```

Note:

1. VOLSER (or VOL) is optional when the data set is cataloged.
2. The <volser> can be a quoted string.

CPF defines a 1-8 character command prefix value for System REXX that can be used instead of specifying the MODIFY AXR command. This prefix may be defined as either SYSTEM or SYSPLEX in scope.

- SYSTEM scope indicates that the CPF will only be recognized on the system it is defined on.
- SYSPLEX scope indicates that it will be recognized throughout the SYSPLEX and the command will be routed to the system on which it is defined.

The default value of CPF is:

```
CPF('REXX&SYSCLONE.',SYSPLEX)
```

For more information about CPF, see the topic on Using the Command Prefix Facility in *z/OS MVS Planning: Operations*.

Example: To obtain information about System REXX, enter @SYSREXX STATUS instead of MODIFY AXR,SYSREXX STATUS.

```
CPF('@',SYSTEM)
```

The AXRUSER parameter specifies a 1-8 character user ID that is used to define the security environment that an exec initiated using the AXREXX macro when SECURITY=BYAXRUSER is specified. The exec will run with the level of authorization associated with the specified user ID.

The installation needs to provide the user ID with SAF access to the resource SYSREXX.<userid>. There is no default for this parameter. If it is omitted, any subsequent AXREXX invocation using AXRUSER will be rejected. Also, the installation security product must be active when system REXX initializes; otherwise, system REXX initialization is delayed.

The REXXLIB concatenation is an ordered list (by appearance in AXRxx) of data sets specified by REXXLIB ADD. When the AXREXX macro is called directly or indirectly from the operator console, System REXX searches this list for the specified exec. If the cataloged SYS1.SAXREXEC is not among the data sets specified, it is appended to the end. All data sets specified in the REXXLIB concatenation must have the same record type and record length as SYS1.SAXREXEC. The data set must either be a PDS or PDSE.

Adding data sets to the REXXLIB concatenation: The number of data sets that you can add to form the REXXLIB concatenation is limited by the total number of DASD extents the data sets will occupy. A partitioned data set extended (PDSE) counts as one extent. The total number of extents must not exceed 255 that is the current system limit.

The system will concatenate as many of the REXXLIB data sets as possible until the system limit of 255 extents is reached. If each data set in the REXXLIB concatenation occupies just a single extent, 255 data sets can be concatenated. The effective size of the concatenation is reduced if data sets in the concatenation occupy secondary extents. Each secondary extent reduces the concatenation size by one. The total reduction is the sum of all secondary extents. When the system limit is exceeded, message AXR0115E is issued, and no further requests are processed by System REXX.

The MaxWorkerTasks(*nn*) parameter indicates the maximum number of worker tasks that are running TSO=NO work in the AXR address space, where *nn* is a value in the range of 4 to 64. The default value is 32.

For more information, see *z/OS DFSMS Using Data Sets*.

Chapter 11. BLSCECT (formatting exits for dump and trace analysis)

The BLSCECT member specifies, through embedded parmlib members, the names of IBM-supplied exit routines. The routines analyze and format dump and trace data. While analyzing a dump or trace, you can enter interactive problem control system (IPCS) subcommands to run the routines.

IPCS accesses the BLSCECT parmlib member through the IPCSPARM DD statement. If you allocate FILE(IPCSPARM), IPCS uses this data set for BLSCECT and the embedded parmlib members. If IPCSPARM is not allocated, the current concatenation is allocated, used, and then freed.

For information about IPCS, see the following references:

- *z/OS MVS IPCS User's Guide* for information about using IPCS.
- *z/OS MVS IPCS Commands* for the IPCS commands and subcommands.
- *z/OS MVS IPCS Customization* for information about formatting exits.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for BLSCECT

IPCS processes each statement as one or more lines of TSO/E Command Language. See *z/OS TSO/E Command Reference* and *z/OS TSO/E User's Guide*.

Comments may appear in columns 1-71 and must begin with `/*` and end with `*/`.

IBM-supplied default for BLSCECT

The default member is BLSCECT. When the system reads BLSCECT, it processes the parmlib members specified on IMBED statements in the order in which the statements appear.

The IBM-supplied BLSCECT member contains an IMBED statement for the BLSCECTX parmlib member. BLSCECTX contains formatting exit routines for IBM components other than the MVS base control program.

Statements/parameters for BLSCECT

The statements and parameters used in BLSCECT and in the parmlib members embedded by its IMBED statements are described for the BLSCUSER parmlib member; see "Statements/parameters for BLSCUSER, BLSCECT, and embedded parmlib members" on page 127.

IBM suggests that:

- You use the IBM-supplied BLSCECT without changing it or adding to it.
- You use the IBM-supplied BLSCECTX without changing it or adding to it.
- You specify your customization in BLSCUSER.

In this way, you can upgrade to a new release without having to move your customization from the old BLSCECT to the new BLSCECT and without changing BLSCECTX.

The IMBED statements in the IBM-supplied BLSCECT and BLXCECTX members are required.

Chapter 12. BLSCUSER (installation customization for dump and trace analysis)

The BLSCUSER member contains installation-supplied customization to the interactive problem control system (IPCS). For example, it can contain:

- The name of an installation-supplied routine to analyze a data area or structure.
- Specification of an installation-supplied analysis dialog to be added to the IPCS dialog.
- The name of an installation-supplied exit routine that can analyze and format dump or trace data.
- The name of an installation-supplied parmlib member to be embedded.
- Specification of a message.
- The definition of an installation-supplied input or help panel to be added to the IPCS dialog.
- The definition of a special symbol.
- The name of the sysplex dump directory, if the default of SYS1.DDIR is not used.
- A TSO/E command or CLIST to be run when the parmlib member is being activated.

During system initialization, the BLSCUSER member supplies guidance on formatting support for SNAP and ABEND dumps formatted by the system at the time of error. During system initialization it also supplies the name of the sysplex dump directory to be used by SDUMP to log unformatted system dumps. ENVIRONMENT options on various statements permit filtering of information, for example, whether environments that exploit freeway architecture are supported.

During initialization of an IPCS session, the BLSCUSER member supplies guidance for a broad range of analysis and formatting actions that may take place during that session. ENVIRONMENT options here permit filtering of information only intended for use by SNAP. During initialization of a session, IPCS does not know it whether it will need to analyze dumps of systems running in ESA mode, z/Architecture® mode, or both. As a result, BLSCUSER processes information needed to handle dumps from both modes.

The BLSCUSER member must be in the same parmlib as the BLSCECT member. See Chapter 11, “BLSCECT (formatting exits for dump and trace analysis),” on page 123 for more information.

The statements in BLSCUSER, BLSCECT, and embedded parmlib members generate standard IPCS return codes. These codes can be tested by CLISTs. Any serious or terminating error in processing any of the statements will prevent the system from using the parmlib data for a SNAP or ABEND dump and will cause an IPCS session to be abnormally ended.

For information about IPCS, see:

- *z/OS MVS IPCS User's Guide* for information about using IPCS
- *z/OS MVS IPCS Commands* for the IPCS commands and subcommands
- *z/OS MVS IPCS Customization* for information about formatting exits

Parameter in IEASYSxx (or issued by the operator)

None.

Syntax rules for BLSCUSER

IPCS processes each statement as a line of TSO/E Command Language. See *z/OS TSO/E Command Reference* and *z/OS TSO/E User's Guide*.

- All statements are optional.
- BLSCUSER can contain all statements, except SYSDDIR, more than once.
- Comments may appear in columns 1-71; they must begin with `/*` and end with `*/`.

Syntax format of BLSCUSER

```
CTRACE COMPONENT(component-name)
       FORMATTABLE(table-epname)
       ENVIRONMENT(ALL|{ESAME|ZARCHITECTURE}|ESA)

DATA  {AREA(namelist)      }
      {STRUCTURE(namelist) }
      [FIND(epname)                    ]
      [FORMAT(name[,level])            ]
      [PROCEDURE(procname) ]
      [GROUP(groupname)                ]
      [MODEL(modelname)                ]
      [SCAN(scanname[,level])          ]
      [ENVIRONMENT(ALL) ]
      [ENVIRONMENT([IPCS|SNAP]  ] [{ESAME|ZARCHITECTURE}|ESA]) ]

DIALOG NAME(dialogname)
      [ABSTRACT('text') ]
      [PARM('text') ]
      [ENVIRONMENT([{ESAME|ZARCHITECTURE}|ESA]) ]
END
EXIT  EP(pgmname[,level])
      [ANALYZE      ]
      [ASCB         ]
      [CBSTAT(name) ]
      [FORMAT(name) ]
      [TCB          ]
      [VERB(name) ]
      [ABSTRACT('text') ]
      [AMASK(X'00FFFFFF' | X'7FFFFFFF') ]
      [ENVIRONMENT(ALL)                    ]
      [ENVIRONMENT([IPCS|SNAP] [{ESAME|ZARCHITECTURE}|ESA]) ]
      [HELP(name) ]
      [PARM('text') ]

IMBED MEMBER(membername)
      [REQUIRED ]
      [ENVIRONMENT(IPCS | SNAP | ALL) ]
```

```
NOTE  ['text' ]
      [CAPS | ASIS ]
      [PAGE | NOPAGE ]
      [SPACE[(count)] | NOSPACE | OVERTYPE ]
      [TOC([indentation | 1] [toc-text]) | NOTOC ]
      [ FLAG(severity) ]
      [ PRINT | NOPRINT ]
      [ TERMINAL | NOTERMINAL ]
      [ TEST | NOTEST ]

PANDEF SUBCOMD(CTRACE)
      [COMPONENT(component name) ]
      {INPUT(panel name)  }
      {HELP(panel name) }
      [ENVIRONMENT(ALL | {ESAME|ZARCHITECTURE} | ESA) ]

SYMBOL [PREFIX(prefixname) ]
       [SUFFIX(suffix) ]
       [NAME(name) ]
       [AREA(name)      ]
       [STRUCTURE(name) ]
       [ENVIRONMENT([{ESAME|ZARCHITECTURE} | ESA])

SYSDDIR dsname
       [ENVIRONMENT([{ESAME|ZARCHITECTURE} | ESA])

TSO [ [ [%]clistnm | [%]

rexxnm | tsocmd ] [operands] ]
```

IBM-supplied default for BLSCUSER

None.

Statements/parameters for BLSCUSER, BLSCECT, and embedded parmlib members

Note: These statements and their parameters can be used in the BLSCUSER member, the BLSCECT member, and all parmlib members embedded in these members.

CTRACE statement

Enables components to collect CTRACE data without necessarily registering, get the data into an unformatted dump, and have the IPCS subcommand "CTRACE COMP(component-name)..." accepted to locate the component's CTRACE data in a dump.

COMPONENT(*component-name*)

Specifies the name of the component.

FORMATTABLE(*table-epname*)

Specifies the entry point of the format table containing the component-supplied find routine.

ENVIRONMENT(ALL)
ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)

Specifies that IPCS should support the data type in environments that exploit freeway architecture and those that do not.

DATA statement

Associates specific types of areas or structures with an exit routine, a model, or a group of areas or structures. Use one DATA statement for each type of data specification. For information about the exit routines, see *z/OS MVS IPCS Customization*.

AREA(*namelist***)**
STRUCTURE(*namelist***)**

Identifies one or more names of areas or structures. Each name in the *namelist* is 1 to 31 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase. Separate names with blanks or commas, or in other ways used in TSO/E commands.

FIND(*epname***)**

Specifies the name of the entry point in a find exit routine that is used to locate the data types within this group. (See the description of the GROUP parameter.) See the find exit routine in *z/OS MVS IPCS Customization*.

The *epname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

FORMAT(*name***[,***level***])**

Specifies the name of the control block formatter exit routine to be used to format the data types within this group. See the control block formatter exit routine in *z/OS MVS IPCS Customization*.

The *name* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

The *level* is a function systems mode ID (FMID), which indicates a version and release of the MVS system and which associates the formatter with that level application programming interface (API) support. The *level* option may be specified as one of the following values:

- **JBB2125** indicates that the formatter expects the structure that it formats will reside in 31-bit virtual or real storage, storage that can be described without the use of BLSRESSY structures.
- **HBB3310** indicates that the formatter expects the structure that it formats will reside in 31-bit storage. The description may be provided by one or more 31-bit BLSRESSY structures or using the JBB2125 API.
- **HBB7703** indicates that the formatter expects the structure that it formats will reside in 64-bit storage. The description may be provided by one or more 31-bit or 64-bit BLSRESSY structures or using the JBB2125 API.

Default: HBB3310

PROCEDURE(*procname***)**

An alternative data definition to FORMAT that specifies the designated data type. FORMAT and PROCEDURE are mutually-exclusive keywords. When the CBFORMAT subcommand or service routine is asked to format the designated data type during the session, issue the procedure command with the syntax: `%procname X`.

GROUP(*groupname***)**

Specifies the name of a control block group, such as RBs and UCBs. Pointers in other control blocks (such as TCBs) may address one of these blocks, and IPCS may need to record the existence of the block in the

dump directory before it is determined whether the block is, in fact, a PRB instead of another type of RB or a UCBTAPE instead of another type of UCB.

The *groupname* specifies the type of control blocks to be associated with each type of data being defined by this DATA statement. Do not reference an area or structure as a *groupname* if the AREA or STRUCTURE parameter references another group data type. If an attempt is made to establish such a relationship, IPCS will detect an error, and the group data type associated with the explicitly-referenced group will be used instead.

The *groupname* is 1 to 31 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

MODEL(*modelname***)**
Specifies a model to format data types within this group. See format models in *z/OS MVS IPCS Customization*.

The *modelname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

SCAN(*scanname***[,***level***])**
Specifies the name of a scan routine that is to verify each instance of a data type in this group. See the scan exit routine in *z/OS MVS IPCS Customization*.

The *scanname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

The *level* is a function systems mode ID (FMID), which indicates a version and release of the MVS. The *level* option may be specified as one of the following values:

- Omission of *level* indicates that the scan routine expects the structure that it scans will reside in 31-bit storage. The description will be provided to the scan routine by a 31-bit BLSRSASY structure, and the results of the scan should be returned in that format.
- **HBB7703** indicates that the scan routine expects the structure that it scans will reside in 64-bit storage. The description will be provided to the scan routine by a 64-bit BLSRSASY structure, and the result of the scan should be returned in that format.

  Note: The BLSRSASY structure passed to a scan routine contains a field that indicates whether it is in 31-bit or 64-bit format. If a scan routine needs to be written for use with the HBB7703 API as well as the earlier API, it can use this information to determine the format of the BLSRSASY structure.

ENVIRONMENT(ALL)
ENVIRONMENT(IPCS)
ENVIRONMENT(SNAP)
ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that IPCS should support the data type in IPCS and SNAP environments and in environments that exploit freeway architecture and those that do not.

Default Value: ALL

DIALOG statement
Specifies an analysis dialog that a user may select from the IPCS MVS Dump Component Data Analysis panel of the IPCS dialog. See option 2.6 in the *z/OS MVS IPCS User's Guide*.

NAME(*dialogname*)
Specifies the name for the analysis dialog. The *dialogname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase. Each *dialogname* must be unique. It must not duplicate another dialog name on a DIALOG statement or a verb name on an EXIT statement specified in BLSCECT and its embedded parmlib members.

ABSTRACT('*text*')
Specifies text to appear on the IPCS MVS Dump Component Data Analysis panel. The text describes the analysis dialog processing.

The format of the line displayed on the panel is:

```
dialogname - text
```

The *text* is 1 to 60 alphabetic characters and must be enclosed in single quotation marks. Uppercase and lowercase letters are accepted; both cases are shown on the panel.

PARM('*text*')
Specifies data to be passed to the ISPF SELECT service when the analysis dialog is active. The *text* is 1 to 32,767 alphanumeric characters and must be enclosed in single quotation marks. Lowercase alphabetic characters are accepted and treated as uppercase.

ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that IPCS naming conventions for symbols should apply in environments that exploit freeway architecture and those that do not.

END statement
Ends processing of an embedded parmlib member. Any statement following an END statement in an embedded parmlib member is not processed. Processing of the member containing the IMBED statement continues.

If a parmlib member contains a CLIST, the system considers the CLIST as part of the member. If a CLIST generates an END statement, the CLIST generating the END statement, any other CLIST invoked by the currently embedded member, and the currently embedded member are ended.

An END statement is useful to prevent processing of a block of statements that you want IPCS to ignore for the time being.

EXIT statement
Specifies an exit routine. The user can invoke the exit routine during IPCS processing by a subcommand:

- By an ANALYZE subcommand if ANALYZE is specified
- By an ASCBEXIT subcommand if ASCB is specified
- By a CBSTAT STRUCTURE(*name*) subcommand if CBSTAT(*name*) is specified
- By a TCBEXIT subcommand if TCB is specified
- By a VERBEXIT *verbname* subcommand if VERB(*verbname*) is specified

The ABSTRACT, HELP, and PARM parameters tell IPCS how to make the component analysis from your exit routine available through the IPCS dialog.

EP(*pgmname***[,***level***])**
The *pgmname* option specifies the name of the entry point for the exit routine. The *pgmname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @).

The *level* is a function systems mode ID (FMID), which indicates a version and level of the MVS system. The *level* option pertains to CBSTAT and post-formatting exits and may be specified as one of the following values:

- Omission of *level* indicates that the scan routine expects the structure of interest will reside in 31-bit storage. The description will be provided by a 31-bit structure.
- **HBB7703** indicates that the routine expects the structure of interest will reside in 64-bit storage. The description will be provided by a 64-bit structure.

Note: The structure passed contains a field that indicates whether it is in 31-bit or 64-bit format. If a routine needs to be written for use with the HBB7703 API as well as the earlier API, it can use this information to determine the format of the structure.

ANALYZE
Specifies that IPCS invoke the exit routine during contention analysis in response to an ANALYZE subcommand. The system invokes the ANALYZE exit routines in the order they are specified in BLSCECT and its embedded parmlib members. See the ANALYZE exit routine in *z/OS MVS IPCS Customization*.

If VERB(*verbname*) is also specified, the exit routine will also be invoked by the VERBEXIT *verbname* subcommand. If ABSTRACT is also specified, the exit routine will also be invoked through the IPCS MVS Dump Component Data Analysis panel of the IPCS dialog.

In the following example, the exit routine analyzes GRS ENQ contention.

```
EXIT EP(ISGDCONT) ANALYZE
```

ASCB
Specifies that IPCS invoke the exit routine when an ASCBEXIT subcommand is entered. The system invokes the ASCB exit routines in the order they are specified in BLSCECT and its embedded parmlib members. See the ASCB exit routine in *z/OS MVS IPCS Customization*.

If VERB(*verbname*) is also specified, the exit routine will also be invoked by the VERBEXIT *verbname* subcommand.

CBSTAT(*name***)**
Specifies that IPCS invoke the exit routine when a CBSTAT STRUCTURE(*name*) subcommand is entered and, for CBSTAT exit routines associated with units of work, when an ANALYZE subcommand is entered for the unit of work. See the ANALYZE and CBSTAT exit routines in *z/OS MVS IPCS Customization*.

The *name* is 1 to 31 alphanumeric characters and must begin with an EBCDIC letter (A-Z). Lowercase alphabetic characters are accepted and treated as uppercase.

This example is for a RTM ASCB status exit.

```
EXIT EP(IEAVTRCA) CBSTAT(ASCB)
```

FORMAT(*name***)**
Specifies that IPCS invoke the exit routine when the following are entered:

- A SUMMARY subcommand
- A CBFORMAT subcommand with an EXIT parameter and a STRUCTURE(*cbname*) parameter.

See the post-formatting exit routine in *z/OS MVS IPCS Customization*.

For the SUMMARY subcommand, the system invokes the FORMAT exit routines in the order they are specified in BLSCECT and its embedded parmlib members.

The *name* is 1 to 31 alphanumeric characters and must begin with an EBCDIC letter (A-Z). Lowercase alphabetic characters are accepted and treated as uppercase.

TCB
Specifies that IPCS invoke the exit routine when a TCBEXIT subcommand is entered. The system invokes the TCB exit routines in the order they are specified in BLSCECT and its embedded parmlib members. See the TCB exit routine in *z/OS MVS IPCS Customization*.

If VERB(*verbname*) is also specified, the exit routine will also be invoked by the VERBEXIT *verbname* subcommand.

The following example provides COMM TASK TCB exit for WTORs.

```
EXIT EP(IEAVG701) CBSTAT(TCB)
```

VERB(*verbname*)
Specifies that IPCS invoke the exit routine when a VERBEXIT *verbname* subcommand is entered. See the verb exit routine in *z/OS MVS IPCS Customization*.

If ANALYZE, ASCB, or TCB is also specified, the exit routine will also be invoked by the ANALYZE, ASCBEXIT, or TCBEXIT subcommand.

The *verbname* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase. Each *verbname* must be unique. It must not duplicate another verb name on an EXIT statement or a dialog name on a DIALOG statement specified in BLSCECT and its embedded parmlib members.

To specify more than one *verbname* for the same EP *pgmname*, code a separate EXIT statement for each *verbname*.

The following example specifies a VERBEXIT VLFDATA subcommand, which will analyze VLF control blocks.

```
EXIT EP(IGVSFMAN) VERB(VLFDATA)
```

ABSTRACT('*text*')
Specifies text to appear on the IPCS MVS Dump Component Data Analysis panel. The text describes the exit routine processing. Specify this parameter when VERB is also specified.

The format of the line displayed on the panel is:

```
verbname - text
```

The *text* is 1 to 60 alphabetic characters and must be enclosed in single quotation marks. Uppercase and lowercase letters are accepted; both cases are shown on the panel. If an exit routine is invoked by multiple verb names, the first verb name associated with the exit in the parmlib member(s) is used on the panel.

AMASK(X'00FFFFFF' | X'7FFFFFFF')
Specifies a logical AND mask. The exit routine uses the mask when it passes storage addresses to the dump access service. Any TSO/E INTEGER specification equivalent to these values is accepted. A TSO/E integer may be specified as decimal, hexadecimal, or binary. Hexadecimal is recommended.

The following example is for VTAM.

```
EXIT EP(VTAMMAP) AMASK(X'00FFFFFF')
```

Default Value: X'7FFFFFFF'

ENVIRONMENT(ALL)
ENVIRONMENT(IPCS)
ENVIRONMENT(SNAP)
ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that IPCS should support the data type in IPCS and SNAP environments and in environments that exploit freeway architecture and those that do not.

Default Value: ALL

HELP(*name***)**
Specifies the name of the help panel that is to be displayed when a user enters a question mark in the selection field of a exit selection panel in the IPCS dialog. The help panel should be an online description of the content of the report generated by the exit routine. Specify this parameter when ABSTRACT is also specified.

The *name* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

PARM('*text***')**
Specifies a parameter that IPCS is to pass to the exit routine when it is invoked from the IPCS MVS Dump Component Data Analysis panel. Specify this parameter when ABSTRACT is also specified.

The *text* is 1 to 32,767 alphanumeric characters and must be enclosed in single quotation marks. Mixed upper and lowercase alphabetic characters will be accepted and passed as is to the routine.

IMBED statement
Embeds other parmlib members. The embedded parmlib members can embed additional parmlib members.

MEMBER(*membername***)**
Specifies the name of the parmlib member that IPCS should process before the processing of the current parmlib member is resumed.

The *membername* is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

REQUIRED
Specifies that IPCS treat a failure to locate the parmlib member as an error. If you omit the REQUIRED parameter, IPCS accepts failure to locate the member and issues no messages.

ENVIRONMENT(IPCS)
ENVIRONMENT(SNAP)

ENVIRONMENT(ALL)
Specifies that IPCS should support exit routines in the parmlib members in an IPCS or SNAP environment or in both. For an ABEND or SNAP dump to be processed correctly, specify ALL.

Default Value: IPCS

NOTE statement
Transmits messages to the terminal, to FILE(IPCSPRNT), or to both. The statement is a problem determination aid for construction and maintenance of parmlib members that are embedded in BLSCECT and in members embedded in BLSCECT.

NOTE statement syntax is identical to the syntax of the IPCS NOTE subcommand.

PANDEF statement
Specifies an input panel or help panel that is to be displayed in the IPCS dialog.

In the parameters on the PANDEF statement, each name is 1 to 8 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

SUBCOMD(CTRACE)
Specifies the name of the IPCS subcommand with which the panel is to be associated; only the CTRACE subcommand supports the PANDEF statement.

COMPONENT(*component name***)**
Specifies the name of the component that defines the options to be captured (input panel) or explained (help panel).

INPUT(*panel name***)**
Specifies the name of the input panel used to obtain user specifications of component-specific options.

HELP(*panel name***)**
Specifies the name of the help panel displayed to describe component-specific options to the user.

ENVIRONMENT(*ALL***)**
ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that IPCS should support the data type in environments that exploit freeway architecture and those that do not.

SYMBOL statement
Defines a group of special symbols to IPCS.

PREFIX(*prefixname***)**
Specifies the initial characters, *prefixname*, for a group of special symbols, such as ASCBnnnnn, TCBnnnnnaaaaa, UCBxxxx, or ASTnnnnn.

The *prefixname* is 1 to 30 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are treated as uppercase. The number of characters allowed in the *prefixname* depends on the associated suffix:

SUFFIX Value	Maximum Prefix Length
COUNT0	1 to 26 characters

SUFFIX Value	Maximum Prefix Length
COUNT1	1 to 26 characters
COUNT1NAME	1 to 25 characters
CPU	1 to 29 characters
DUALCOUNT	1 to 21 characters
NAME	No longer than 31 characters minus the length of the prefix
UNIT	1 to 26 characters

SUFFIX(*suffix*)
Specifies the syntax requirements for the final characters for a group of special symbols. The values for the *suffix* are:

COUNT0
Specifies a decimal suffix of 0 - 99999. An example is the suffix of 0 in the symbol AST0.

COUNT1
Specifies a decimal suffix of 1 - 99999. Some examples are the suffixes of 5 in the symbol ASCB5 and of 534 in ASXB534.

COUNT1NAME
Specifies a suffix of a decimal number of 1 - 99999 plus a name. The name consists of alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, @, #). An example is the suffix of 75PSFA in the symbol WTRFSCB75PSFA.

CPU
Specifies a suffix of a decimal number of 0 - 99. Some examples are the suffixes of 0 in the symbol LCCA0, of 15 in PCCA15, and of 7 in PSA7.

DUALCOUNT
Specifies a suffix of 2 numbers:

- First, a decimal number of 1 - 99999
- Second, a number of 1 - 5 EBCDIC letters (A-Z), which is a base-26 value in which leading A's may be omitted.

Some examples are the suffixes of 34C in the symbol JSABA34C, of 1A in PGT1A, and of 5F in TCB75F.

NAME
Specifies a suffix that is a name. The name consists of 1 or more alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, @, #). An example is the suffix of ABC in the symbol JOBABC.

UNIT
Specifies a suffix of a hexadecimal number of 0-1FFFF and is intended to be a device number. An example is the suffix of 01D0 in the symbol UCB01D0.

NAME(*name*)
Specifies the complete name of the symbol. The *name* is 1 to 30 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, @, #). Lowercase alphabetic characters are accepted and treated as uppercase.

Examples are: CVT, GDA, or PRIVATE.

AREA(*name*)

STRUCTURE(*name***)**
Specifies the data type for the symbol. The *name* is 1 to 31 alphanumeric characters and must begin with an EBCDIC letter (A-Z) or national character ($, #, @). Lowercase alphabetic characters are accepted and treated as uppercase.

This parameter allows an IPCS user to enter a shortened version of a command. For example, an IPCS user can enter:

```
list cvt
```

This subcommand is the same as:

```
list cvt structure(cvt)
```

ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that IPCS name conventions for symbols should apply in environments that exploit freeway architecture and those that do not.

SYSDDIR *dsname* **statement**
Identifies the name of the data set for the sysplex dump directory. The *dsname* must be fully qualified. The name can be specified with or without apostrophes enclosing it.

Default: SYS1.DDIR

ENVIRONMENT({ESAME|ZARCHITECTURE})
ENVIRONMENT(ESA)
Specifies that the same sysplex dump directory is used by systems that exploit freeway architecture and those that do not.

TSO *command* **statement**
Invokes a TSO/E command, CLIST, or REXX exec during the processing of the parmlib member. TSO statement syntax is identical to the syntax of the IPCS TSO subcommand.

Use the TSO statement when the TSO/E command, CLIST name, or REXX exec name duplicates the name of a BLSCUSER statement. A TSO/E command with a name that does not duplicate a BLSCUSER statement name can be specified without TSO before it.

A CLIST that generates BLSCUSER statements can also be specified in the parmlib member; place a percent sign (%) before the CLIST name.

TSO/E commands and REXX execs may use the TSO/E stack to leave BLSCUSER statements for processing after the command or exec ends processing.

TSO/E commands supplied with IPCS, IPCS primary commands and IPCS subcommands try to lighten this burden in two ways: Some keywords that you tend to use often, e.g. CHARACTER, support explicit, short aliases, C in this case. All keywords can be truncated, entering just enough of their characters to make what you have entered unambiguous. For example, LENGTH is a keyword accepted on many subcommands. Entering LEN instead of LENGTH is currently unambiguous on them all. If you are composing a command procedure that you hope will remain useful for a long time, do not truncate keywords in it. As IPCS responds to new demands, new keywords can be introduced that make truncations ambiguous. Support for truncation is solely intended to make manual entry of commands and writing command procedures intended for brief use faster and easier.

Chapter 13. BPXPRMxx (z/OS UNIX System Services parameters)

BPXPRMxx contains the parameters that control the z/OS UNIX System Services (z/OS UNIX) environment and the file systems.

To make it easier to migrate from one release to another, especially when using the ServerPac method of installation, use two BPXPRMxx parmlib members. One member defines the values to be used for system setup and the other member defines the file systems.

To specify which BPXPRMxx parmlib member to start with, the operator can include OMVS=xx in the reply to the IPL message or OMVS=xx in the IEASYSxx parmlib member. The two alphanumeric characters, represented by xx, are appended to BPXPRM to form the name of the BPXPRMxx parmlib member.

If OMVS=xx is not specified in the reply to the IPL message or is not in the IEASYSxx member, or if OMVS=DEFAULT is specified, defaults are used for each parameter and the kernel services are started in minimum mode. For more information about running in minimum mode and full function mode, see *z/OS UNIX System Services Planning*. If the operator specifies OMVS=xx in the reply to the IPL message, it overrides the OMVS=xx specified in IEASYSxx.

Restriction: The START OMVS,OMVS=xx command is not valid when issued from the command console. OMVS=xx is not valid in parmlib COMMNDxx.

You can use multiple parmlib members to start OMVS. This is shown by the following reply to the IPL message:

```
R 0,CLPA,SYSP=R3,LNK=(R3,R2,L),OMVS=(AA,BB,CC)
```

You can also specify multiple BPXPRMxx parmlib members using the OMVS keyword in IEASYSxx. For example:

```
OMVS=(AA,BB,CC)
```

If the same parameters are specified in more than one member, the first instance of the keyword that is specified in the leftmost member is used. The MOUNT statements are processed in the order they are specified in LEFT to RIGHT sequence.

If you are using SYSPLEX(YES) and mixed releases of z/OS UNIX, you can IPL specifying OMVS=(delta,common) for each unique release, where "delta" identifies the member containing the new keywords for that release, and "common" identifies the common keywords for all releases.

To modify BPXPRMxx parmlib settings without reloading the initial program, you can use the SETOMVS operator command, or you can dynamically change the BPXPRMxx parmlib members that are in effect by using the SET OMVS operator command. For more information, see the section on dynamically changing BPXPRMxx values in *z/OS UNIX System Services Planning*. See *z/OS MVS System Commands* for more information about the SETOMVS and SET OMVS commands.

Syntax rules for BPXPRMxx

When customizing BPXPRMxx, these rules apply:

- Statements that contain limiting keywords (like MAXUIDS, which limits the number of concurrent z/OS UNIX users), should not be duplicated in the same BPXPRMxx member. You can have duplicates of limiting keywords across BPXPRMxx members, but only the last occurrence is used. Resource defining keywords (like MOUNT, which specifies a file system that z/OS UNIX is to logically mount onto the root file system or another file system) are cumulative. Resource defining keywords can be duplicated in the same BPXPRMxx member. Each time you specify a resource defining keyword, its value is added to the previous values.
- If a statement that has a default is omitted, the default is used.
- Use columns 1 through 71 for data; columns 72 through 80 are ignored.
- Enter one or more statements on a line, or use several lines for one statement.
- Use blanks as delimiters. Multiple blanks are interpreted as a single blank. Blanks are allowed between parameters and values; for example, MAXPROCSYS(500) and MAXPROCSYS (500) are allowed and have the same meaning.
- Comments can appear in columns 1-71 and must begin with `/*` and end with `*/`.
- Enter values in uppercase, lowercase, or mixed case. The system converts the input to uppercase, except for values that are enclosed in single quotation marks, which are processed without changing the case.
- These values require single quotation marks and are the only ones that are allowed to be in single quotation marks:
  - STEPLIBLIST
  - USERIDALIASTABLE
  - FILESYSTEM in the ROOT, MOUNT, and ALTROOT statements
  - MOUNTPOINT in the MOUNT and ALTROOT statement
  - PARM in the FILESYSTYPE, ROOT, MOUNT, SUBFILESYSTYPE, and ALTROOT statements
  - RUNOPTS
  - VERSION
  - AUTHPGMLIST
- Enclose values in single quotation marks, using the following rules:
  - Two single quotation marks next to each other on the same line are considered as a single quotation mark. For example, `John''s file` is considered to be `John's file`. One quotation mark in column 71 and another in column 1 of the next line are not considered as a single quotation mark. This input is treated as two strings or an error.
  - Because some values can be up to 1023 characters, a value can require multiple lines. Place one quotation mark at the beginning of the value, stop the value in column 72 of each line, continue the value in column 1 of the next line, and complete the value with one quotation mark. Figure 8 on page 139 shows an example.

```
column                                                              column
1                                                                      71
|                                                                       |
   MOUNT FILESYSTEM('HFS.WORKDS') MOUNTPOINT('/u/john/namedir1/namedir2
/namedir3/namedir4') TYPE(ZFS) MODE(RDWR)
```

Figure 8. Specifying values on multiple lines

Rather than defining parameter limit values in their full decimal or hexadecimal form, you can use a one-character multiplier (denomination values) suffix when specifying them. This value will also be used in displays when the system returns responses to respective D OMVS commands. The multipliers are listed in Table 14

Restrictions:

1. Only those SETOMVS parameters that support this `C` suffix specifically note that support and refer to Table 14..
2. Values that contain a multiplier are limited to 8 digits (nnnnnnnnC) and those values are limited to X'00FF FFFF' (16 777 215 decimal).
3. Values that do not contain a multiplier are limited to X'7FFF FFFF' (2 147 483 647 decimal).

Tip: A complete description of the D OMVS output can be found in message BPXO040I. To look up that message, go to *z/OS MVS System Messages, Vol 3 (ASB-BPX)*.

Table 14. One-character parameter limit multipliers

Denomination value	One-character abbreviation	Bytes
null	n/a	1
Kilo	K	1,024
Mega	M	1,048,576
Giga	G	1,073,741,824
Tera	T	1,099,511,627,776
Peta	P	1,125,899,906,842,624

Syntax of BPXPRMxx

```
{AUTOCVT(ON|ALL|OFF)}
{MAXIOBUFUSER(nnnnn)}
{LOSTMSG(ON|OFF)}
{PWT(SMF|SMFENV|ENV}
{MAXPROCSYS(nnnnn)}
{MAXPROCUSER(nnnnn)}
{MAXUIDS(nnnnn)}
{MAXFILEPROC(nnnnnn)}
{MAXTHREADTASKS(nnnnn)}
{MAXTHREADS(nnnnnn)}  {MAXPIPEUSER(nnnnn)
{MAXPTYS(nnnnn)}
{MAXFILESIZE(nnnnn|NOLIMIT)}
{MAXCORESIZE(nnnnn)}
{MAXASSIZE(nnnnn)}
{MAXCPUTIME(nnnnn)}
{MAXMMAPAREA(nnnnn)}
{MAXSHAREPAGES(nnnnn)}
```

```
{RESOLVER_PROC(nnnnn|DEFAULT|NONE)}
{SHRLIBRGNSIZE(nnnnn)}
{SHRLIBMAXPAGES(nnnnn)}
{PRIORITYGOAL(service_class_name1,...service_class_name40|NONE)}
{IPCMSGNIDS(nnnnn)}
{IPCMSGQBYTES(nnnnn)}
{IPCMSGQMNUM(nnnnn)}
{IPCSEMNIDS(nnnnn)}
{IPCSEMNOPS(nnnnn)}
{IPCSEMNSEMS(nnnnn)}
{IPCSHMMPAGES(nnnnn)}
{IPCSHMNIDS(nnnnn)}
{IPCSHMNSEGS(nnnnn)}
{IPCSHMSPAGES(nnnnn)}
{FORKCOPY(COW|COPY)}
{SUPERUSER(user_name)}
{TTYGROUP(group_name)}
{CTRACE(parmlib_member_name)}
{STEPLIBLIST('/etc/steplib')}
{USERIDALIASTABLE('/etc/tablename')}
{SERV _LPALIB('dsname', 'volser')}
{SERV _LINKLIB('dsname', 'volser')}
{FILESYSTYPE TYPE(type_name)
       ENTRYPOINT(entry_name)
       PARM('parm')}
       ASNAME(proc_name,'start_parms')
{SYSPLEX(YES|NO)}
```

```
{VERSION('nnnn')}
{ROOT  FILESYSTEM('fsname') or DDNAME(ddname)
       TYPE(type_name)
       MODE(access)
       PARM('parameter')
       SETUID|NOSETUID
       SYSNAME(sysname)
       TAG(NOTEXT|TEXT,ccsid)
       AUTOMOVE|NOAUTOMOVE
       MKDIR('pathname')}
{MOUNT FILESYSTEM('fsname') or DDNAME(ddname)
       TYPE(type_name)
       MOUNTPOINT('pathname')
       MODE(access)
       PARM('parameter')
       SETUID|NOSETUID
       SECURITY|NOSECURITY
       SYSNAME(sysname)
       TAG(NOTEXT|TEXT,ccsid)
       AUTOMOVE[(INCLUDE,sysname1,sysname2,...,[sysnameN|*])]
       |[(EXCLUDE,sysname1,sysname2,...,sysnameN)]
       |NOAUTOMOVE|UNMOUNT
       MKDIR('pathname')}
{NETWORK DOMAINNAME(sockets_domain_name)
       DOMAINNUMBER(sockets_domain_number)
       MAXSOCKETS(nnnnn)
       TYPE(type_name)
       INADDRANYPORT(starting_port_number)
       INADDRANYCOUNT(number_of_ports_to_reserve)}
{SUBFILESYSTYPE NAME(transport_name)
       TYPE(type_name)
       ENTRYPOINT(entry_name)
       PARM('parameter')
       DEFAULT}
{STARTUP_PROC(procname)}
{STARTUP_EXEC('dsname(membername)',class)}
{RUNOPTS('string')}
{SYSCALL_COUNTS(YES|NO)}
{MAXQUEUEDSIGS(nnnnnn)}
{LIMMSG(NONE|SYSTEM|ALL)}
{AUTHPGMLIST('/etc/authfile')|NONE}
{SWA(ABOVE|BELOW)}
{ALTROOT FILESYSTEM('fsname')
         PARM('parameter')
         MOUNTPOINT('pathname')}
{NONEMPTYMOUNTPT(NOWARN|WARN|DENY)}
{MAXUSERMOUNTSYS(nnnnn)}
{MAXUSERMOUNTUSER(nnnnn)}
```

Syntax example of BPXPRMxx

```
AUTOCVT(OFF)
MAXPROCSYS(400)
MAXPROCUSER(16)
MAXUIDS(200)
MAXFILEPROC(20)
MAXPIPEUSER(8730)
MAXTHREADTASKS(100)
MAXTHREADS(500)
MAXPTYS(100)
MAXFILESIZE(1000)
/*--- or --- MAXFILESIZE(300M) -----*/
MAXCORESIZE(4194304)
/*--- or --- MAXCORESIZE(1K) -------*/
MAXASSIZE(41943040)
/*--- or --- MAXASSIZE(10M) --------*/
MAXCPUTIME(1000)
MAXMMAPAREA(4096)
/*--- or --- MAXMMAPAREA(16M) ------*/
 MAXSHAREPAGES(32768)
/*--- or --- MAXSHAREPAGES(10K) ----*/
PRIORITYGOAL(CICS4,CICS4,CICS4,CICS3,CICS2,CICS1,TSO2,TSO1,BAT3,BAT2)
IPCMSGNIDS(500)
IPCMSGQBYTES(262144)
IPCMSGQMNUM(100000)
IPCSEMNIDS(500)
IPCSEMNOPS(25)
IPCSEMNSEMS(25)
IPCSHMMPAGES(256)
/*--- or --- IPCSHMMPAGES(55M) -----*/
IPCSHMNIDS(500)
IPCSHMNSEGS(10)
IPCSHMSPAGES(262144)
/*--- or --- IPCSHMSPAGES(300K) ----*/
FORKCOPY(COW)
SUPERUSER(BPXROOT)
TTYGROUP(TTY)
CTRACE(CTCBPX23)

STEBLIBLIST('/etc/steplib')
USERIDALIASTABLE('/etc/tablename')
SYSPLEX(YES)
 VERSION('REL9')
 FILESYSTYPE TYPE(ZFS)
      ENTRYPOINT(IOEFSCM)
      ASNAME(ZFS)
ROOT  FILESYSTEM('OMVS.ROOT')
      TYPE(ZFS)
      MODE(RDWR)
      SYSNAME(SY1)
      TAG(NOTEXT,0)
      AUTOMOVE

MOUNT FILESYSTEM('OMVS.USER.JONES')
      TYPE(ZFS)
      MOUNTPOINT('/u/jones')
      MODE(RDWR)
      SYSNAME(SY1)
      TAG(TEXT,1047)
      AUTOMOVE(INCLUDE,SYS1,SYS2,*)

ALTROOT FILESYSTEM('OMVS.ALTROOT')
        MOUNTPOINT('/sysalt')
        PARM('  ')
```

```
FILESYSTYPE TYPE(INET)
      ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(INET)
STARTUP_PROC(OMVS)
STARTUP_EXEC('OMVS.ROOT(REXX01)',A)
RUNOPTS('')
SYSCALL_COUNTS(YES)
MAXQUEUEDSIGS(1000)
RESOLVER_PROC(DEFAULT)
AUTHPGMLIST('/etc/authfile')
LOSTMSG(ON)
SWA(ABOVE)
NONEMPTYMOUNTPT(DENY)
MAXUSERMOUNTSYS(100)
MAXUSERMOUNTUSER(2)
```

Note:

1. This is an example only; the values presented here are not optimal nor recommended values for your installation.
2. The **bold comments** present alternate specifications of those parameters that support the use of multipliers. Refer to Table 14 on page 139 for information concerning the use of multipliers.

IBM-supplied default for BPXPRMxx

There is no default BPXPRMxx parmlib member. SYS1.SAMPLIB provides a sample parmlib member BPXPRMXX.

Statements and parameters for BPXPRMxx

For guidance information about selecting values for the statements, see the section on customizing z/OS UNIX in *z/OS UNIX System Services Planning*.

ALTROOT FILESYSTEM('fsname') PARM('parameter') MOUNTPOINT('pathname')
Specifies the alternate sysplex root file system that is to automatically replace the current sysplex root file system when the current one becomes unowned.

When the replacement occurs, the alternate sysplex root file system is unmounted and then mounted again as the current sysplex root on all systems in the shared file system configuration. If the replacement is successful, the failed sysplex root is unmounted.

The alternate sysplex root file system is always mounted in the READ mode and designated as AUTOMOVE=YES when it becomes the current sysplex root file system. The previous mount mode and AUTOMOVE settings of the alternate sysplex root and the failed sysplex root are ignored. The mount parameters are preserved when the alternate sysplex root file system becomes the current sysplex root file system.

To manually replace the failed or failing sysplex root file system, use the MODIFY OMVS,NEWROOT operator command with the COND=FORCE option. To validate the syntax of the ALTROOT statement, use the SETOMVS SYNTAXCHECK operator command. See the steps for dynamically replacing the sysplex root file system topic in *z/OS UNIX System Services Planning* for step-by-step instructions of manually replacing the failed or failing sysplex root. For more information about the mentioned system commands, see *z/OS MVS System Commands*.

FILESYSTEM('fsname')
The name of the alternate sysplex root file system.

PARM('parameter')
The parameter to be passed directly to the file system type.

MOUNTPOINT('pathname')
The absolute path name, or a symbolic link that resolves to the path name of the directory onto which the file system is to be mounted. MOUNTPOINT must be specified in the ALTROOT statement. The mount point for the alternate sysplex root file system must reside in the root directory of the current sysplex root file system, otherwise mount of the ALTROOT will fail. The absolute path name can be up to 64 characters.

AUTHPGMLIST('/etc/authfile')|NONE
Specifies the path name of a hierarchical file system (HFS) file that contains the lists of APF-authorized path names and program names. If you do not specify a value for AUTHPGMLIST, or if you specify NONE, invocations of APF-authorized and program controlled programs will not be checked against a list of authorized programs or authorized path names. If you specify a path name for the AUTHPGMLIST parameter, the system checks this list during hfsload, exec and spawn processing. If the target program of an exec or spawn has an authorization code of 1 (AC=1), then that program name must appear in the authorized program list.

Use the SETOMVS or SET OMVS command to dynamically change the value of AUTHPGMLIST. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For more information, see *z/OS UNIX System Services Planning*.

AUTOCVT(ALL|ON|OFF)
Activates and deactivates automatic conversion of I/O data using coded character sets for the program and its associated files. The coded character set identifiers (CCSIDs) are specified by the program or by setting the appropriate environment variables at run time. The system AUTOCVT indicator can be overridden by individual programs at a thread or file descriptor level; AUTOCVT is a controlling global switch only for existing programs that do not explicitly establish their own conversion environment.

Default: OFF

You can use the SETOMVS or SET OMVS commands to change the value of AUTOCVT between ALL, ON, and OFF. Changing this conversion mode does not affect conversion of opened files for which I/O has already started.

AUTOCVT(ALL) enables Unicode Services conversion of data during read and write I/O operations.

AUTOCVT(ON) enables Enhanced ASCII conversion of data during read and write I/O operations. Conversion is limited to the EBCDIC (1047) and ASCII (819) code pages.

Guideline: When AUTOCVT(ALL) or AUTOCVT(ON) is set, every read and write operation for a file must be checked to see if conversion is necessary, which incurs a performance penalty involved, even if no conversion occurs. If possible, keep AUTOCVT(OFF) and have each program enabled for conversion. To do this, set the compile or runtime environment variables that control conversion or use programmable function, as described in *z/OS UNIX System Services Planning*.

Automatic conversion can also be controlled individually by a program with one of the following flags in the thread Thli control block (BPXYTHLI):

```
ThliCvtOn - Activates automatic conversion for this thread.
ThliCvtOff - Deactivates automatic conversion for this thread.
```

Both bits must not be on at the same time.

Automatic conversion is accomplished between programs and files that are tagged with different CCSIDs when a conversion table exists for that CCSID pair in the system. CCSID values are defined in *Character Data Representation Architecture*. For information about supported Unicode Services CCSIDs, see *z/OS Unicode Services User's Guide and Reference*.

AUTOMOVE|NOAUTOMOVE

The AUTOMOVE and NOAUTOMOVE parameters apply only in a sysplex where systems are participating in shared file system. They indicate what happens to the ownership of the file system when a shutdown, PFS termination, dead system takeover, or file system move occurs. AUTOMOVE indicates that ownership of the file system automatically changes to another system that is participating in the shared file system. NOAUTOMOVE indicates that ownership of the file system is not moved if the owning system goes down; as a result, the file system becomes inaccessible.

Note: When you specify NOAUTOMOVE, the file system becomes inaccessible when the owning system goes down. But it still exists in the file system hierarchy. The file system remains unowned until the original owing system re-IPLs.

Use AUTOMOVE for the sysplex root file system and the version file system. For file systems that are associated with a single system, specify UNMOUNT; this includes `/etc`, `/tmp`, `/var`, `/dev`, and other system-specific file systems. For descriptions of the sysplex root, system-specific, and version file systems, see Sharing file systems in a sysplex in *z/OS UNIX System Services Planning*.

To ensure that the root is always available, use the default.

Default: AUTOMOVE

CTRACE(parmlib_member_name)

Specifies the parmlib member that contains the initial tracing options to be used for the z/OS UNIX component. Use this statement to provide tracing while the kernel is starting and to avoid having to issue a TRACE operator command to set tracing options.

Default: CTIBPX00

FILESYSTYPE TYPE(type_name) ENTRYPOINT(entry_name) PARM('parm') ASNAME(proc_name[,'start_parms'])

Specifies the type of file system that is to be started. BPXPRMxx can contain more than one FILESYSTYPE statement.

When SYSPLEX(YES) is specified, each FILESYSTYPE in use within the participating shared file system group must be defined for all systems that are participating in shared file system. The easiest way to accomplish this is by having a single BPXPRMxx member that contains file system information for each system that is participating in shared file system. If you decide to define a BPXPRMxx for each system, the FILESYSTYPE statements must be identical on each system. For more information about shared file system, see *z/OS UNIX System Services Planning*.

Any facilities that are required for a particular FILESYSTYPE must be initiated on all systems that are participating in shared file system. For example, NFS requires TCP/IP, so if you specify an NFS FILESYSTYPE, you must also initialize TCP/IP on NFS initialization.

The SETOMVS RESET command can be used to dynamically specify new FILESYSTYPE statements. To make a permanent change, edit the BPXPRMxx member that is used for IPLs. For more information, see Dynamically adding FILESYSTYPE statements in BPXPRMxx in *z/OS UNIX System Services Planning*.

The parameters are as follows:

ASNAME(proc_name[,'start_parms'])
proc_name specifies the name of a procedure in SYS1.PROCLIB that is to be used to start the address space that is initialized by the physical file system (PFS). Specify ASNAME for any PFS that does not run in the kernel address space. The name that you specify is also used for the name of the address space.

start_parms is an optional quoted string that is to be appended to the *proc_name* 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 system when the address space is started with an internal command. Refer to the START command in *z/OS MVS System Commands* or the ASCRE macro in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*.

By default the address space started with ASNAME is started under JES, but this can be changed by including the additional start_parms **SUB=MSTR**.

ASNAME is an optional parameter. *proc_name* is 1 to 8 characters; the system converts the name to uppercase. If you do not specify ASNAME, or specify *proc_name* as the name of the kernel address space, the PFS is initialized in the kernel address space.

Refer to the documentation for the specific physical file system for valid ASNAME operands.

PARM('parm')
Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotation marks.

Note: For information about specifying the TFS configuration parameters, see the section on parameter key options for the FILESYSTYPE statement in *z/OS UNIX System Services Planning*. For example:

-ea *count*
Allows the TFS file system to automatically grow *count* times.

-em *count*
Allows the TFS file system to manually grow *count* times.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands.

PRM=(aa, bb, ..., zz) is valid for the z/OS File System (zFS).

SYNCDEFAULT(*t*), VIRTUAL(*max*), and FIXED(*min*) are valid only when ENTRYPOINT is GFUAINIT. FSFULL(*threshold*,*increment*) is supported by HFS, TFS, and zFS.

Note: If a syntax error is found in any of these four parameters (SYNCDEFAULT, VIRTUAL, FIXED, or FSFULL), the system issues an error message issued and set all four parameters to the default values.

- **PRM=(***aa, bb, ..., zz***)** is used to define the IOEPRMyy members of the logical parmlib concatenation for specifying zFS configuration parameters, where:

  aa, *bb*, ..., *zz* specify the suffixes of the IOEPRMxx parmlib members in the order in which they are to be concatenated. Up to 32 suffixes may be specified. Suffixes must be in uppercase. Defining an IOEFSPRM configuration file with an IOEZPRM DD card in the zFS procedure will override the PRM specification (that is, cause it to be ignored). See "Specifying the IOEPRMxx concatenation" on page 593 for examples and additional syntax information.

- **SYNCDEFAULT(t)**

  t specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the metadata for a file system is hardened. Setting *t* to 0 indicates that the file system should harden metadata synchronously with syscall requests.

  Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.

  The maximum value that can be specified for *t* is 65534. Values between 65534 and 99999 are rejected.

  A value of 99999 specifies that no sync daemon intervals are specified, and thus, the metadata is not hardened.

  Default: 60 seconds

- **VIRTUAL(*max*)**

  max specifies the maximum amount of virtual storage (in megabytes) that file system data and metadata buffers should use.

  If you do not set a value for *max*, the system assigns a default value that is equal to half the amount of real storage available to the system when z/OS is initialized. See *z/OS UNIX System Services Planning* for more information about the VIRTUAL(max) parameter.

- **FIXED(min)**

  min specifies the amount of virtual storage (in megabytes) that is fixed when z/OS is initialized and remains fixed even if file system activity drops to zero. *min* must be less than or equal to VIRTUAL(max).

  min cannot exceed 50% of real storage available to the system. If the allowed amount of storage is exceeded, an informational message is issued and *min* is set to 50% of real storage. The minimum limit can be changed dynamically by invoking the **confighfs** shell command. See *z/OS UNIX System Services Command Reference* for more information about the **confighfs** shell command.

  Default: 0

- **FSFULL(threshold,increment)**

– *threshold* specifies the percentage of the file system (HFS, TFS, or zFS) capacity at which an operator message is generated. The default is 100%.
– *increment* specifies the percentage of change above the file system capacity at which an operator message is generated. Messages are generated by either an increase or decrease greater than *increment*. The default is 5%.

You can specify *threshold* and *increment* values for the file system. The values can also be set on the MOUNT command for a specific file system. Parameters on the MOUNT command override parmlib values. If no values are specified in either place, no threshold checking is done. If a threshold value is specified but no increment is given, the increment defaults to 5%. The increment value applies both to upgrading the message when the file system continues to fill and to removing the message when more space becomes available due to either deleting files, or to extending the file system. The values are in terms of percent full. The values that are applied to a file system can be changed only when the file system is mounted.

TYPE(type_name)
Specifies the name of the file system type that is to control the file system. TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase. In the FILESYSTYPE statement, specify one of the following types of the file system:

- HFS for a hierarchical file system (HFS).
- zFS for a z/OS File System (zFS).
- NFS for accessing remote files.
- TFS for a temporary file system (TFS).

For planning information, see FILESYSTYPE in *z/OS UNIX System Services Planning*.

FORKCOPY(COW|COPY)
Specifies how user storage is to be copied from the parent process to the child process during a fork() system call.

FORKCOPY(COW)specifies that all fork() calls are processed with the copy-on-write mode if the suppression-on-protection (SOP) hardware feature is available. Before the storage is modified, both the parent and child process refer to the same view of the data. The parent storage is copied to the child only if either the parent or the child modifies the storage. FORKCOPY(COW) causes the system to use the ESQA to manage page sharing.

FORKCOPY(COPY) specifies that fork() immediately copies the parent storage to the child, whether the SOP is available or not. Use this option to avoid any additional ESQA use in support of fork.

Follow these guidelines:

- If the runtime library is in the link pack area, specify FORKCOPY(COPY).
- If the runtime library is not in the link pack area, specify FORKCOPY(COW).

Default: COW

Use the SETOMVS or SET OMVS command to change the value of FORKCOPY dynamically. To make a permanent change, edit the BPXPRMxx member used for IPLs.

IPCMSGNIDS(*nnnnn*)
Specifies the maximum number of unique system-wide message queues.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

Use the SETOMVS or SET OMVS command to change the value of IPCMSGNIDS dynamically. The new minimum is the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *z/OS UNIX System Services Planning*.

IPCMSGQBYTES(*nnnnn*)
Specifies the maximum number of bytes in a single message queue.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647.

Note: The high end of this range is not obtainable due to storage constraints. The actual maximum range varies due to storage allocation and system usage.

Default: 2147483647 (2G)

Use the SETOMVS or SET OMVS command to change the value of IPCMSGQBYTES dynamically.

IPCMSGQMNUM(*nnnnn*)
Specifies the maximum number of system-wide messages for each queue.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647.

Note: The high end of this range is not obtainable due to storage constraints. The actual maximum range varies due to storage allocation and system usage.

Default: 10000

Use the SETOMVS or SET OMVS command to change the value of IPCMSGQMNUM dynamically.

IPCSEMNIDS(*nnnnn*)
Specifies the maximum number of unique system-wide semaphore sets.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

You can change the value of IPCSEMNIDS dynamically using the SETOMVS or SET OMVS command, as described in *z/OS UNIX System Services Planning*.

IPCSEMNOPS(*nnnnn*)
Specifies the maximum number of operations for each semop call.

Value Range: *nnnnn* is a decimal value from 0 to 32767.

Default: 25

You can change the value of IPCSEMNOPS dynamically using the SETOMVS or SET OMVS command.

IPCSEMNSEMS(*nnnnn*)
Specifies the maximum number of semaphores for each semaphore set.

Value Range: *nnnnn* is a decimal value from 0 to 32767.

Default: 1000

You can change the value of IPCSEMNSEMS dynamically using the SETOMVS or SET OMVS command.

IPCSHMMPAGES(*nnnnn*)
Specifies the maximum number of pages for shared memory segments.

Value Range: *nnnnn* is a decimal value from 1 to 4 petabytes (that is, 4 * 1 125 899 906 842 624).

If you obtain memory segments below the 2-gigabyte address range, then a realistic maximum is about 1.5 gigabytes; the actual maximum range varies due to storage allocation and system usage. If you obtain memory segments above the 2-gigabyte address range, the maximum depends on the IEASYS HVSHARE parameter, which specifies the size of the high virtual shared area.

Default: 25600

Use the SETOMVS or SET OMVS command to change the value of IPCSHMMPAGES dynamically.

IPCSHMNIDS(*nnnnn*)
Specifies the maximum number of unique system-wide shared memory segments.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

Use the SETOMVS or SET OMVS command to change the value of IPCSHMNIDS dynamically. The new minimum is the same as the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *z/OS UNIX System Services Planning*.

IPCSHMNSEGS(*nnnnn*)
Specifies the maximum number of attached shared memory segments for each address space.

Value Range: *nnnnn* is a decimal value from 0 to 1000.

Default: 10

You can change the value of IPCSHMNSEGS dynamically using the SETOMVS or SET OMVS command.

IPCSHMSPAGES(*nnnnn*)
Specifies the maximum number of system-wide shared pages that are created by calls to the fork and 31–bit shmat functions.

You can increase, but not decrease, the value, as described in *z/OS UNIX System Services Planning*. Shared memory segments obtained above the 2-gigabyte range in 64-bit programs do not affect this limit.

Value Range: *nnnnn* is a decimal value from 0 to 2621440.

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

Default: 262144

You can change the value of IPCSHMSPAGES dynamically using the SETOMVS or SET OMVS command. The new minimum is the same as the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *z/OS UNIX System Services Planning*.

Because each page of shared storage requires the associated consumption of extended system queue area (ESQA) storage, limiting the shared storage usage provides a way to limit the ESQA usage by z/OS UNIX users. If you use the __IPC_MEGA or __MAP_MEGA options, then the shared pages limits are not affected because MEGA does not affect the system ESQA overhead.

LIMMSG(NONE|SYSTEM|ALL)

Specifies how console messages that indicate when parmlib limits are reaching critical levels are to be displayed.

NONE
Do not display console messages when any of the parmlib limits are reached.

SYSTEM
Display console messages for all processes that reach system limits. In addition, messages are displayed for each process limit of a process if any of the following conditions are met:

- The process limit or limits are defined in the OMVS segment of the owning user ID
- The process limit or limits were changed with a SETOMVS PID=*pid*,*process_limit*

ALL Display console messages for both the system limits and the process limits, regardless of which process reaches a process limit.

Default: NONE

LOSTMSG(ON|OFF)

LOSTMSG(ON) detects lost and duplicate XCF messages in a shared file system configuration. It is ignored if the file system does not have a shared file system configuration; for example, when SYSPLEX(NO) is specified. While LOSTMSG can be specified differently for each member in the sysplex, the same LOSTMSG setting should be specified throughout the sysplex. To disable the detecting of lost and duplicate messages, specify LOSTMSG(OFF).

Tip: Do not use LOSTMSG(ON) when z/OS UNIX sysplex traffic is high, such as when many file systems that are not sysplex-aware are being accessed remotely because performance might be affected.

LOSTMSG(ON) is the default.

MAXASSIZE(*nnnnn*)

Specifies the RLIMIT_AS resource values that will be established as the initial values for new processes. RLIMIT_AS indicates the address space region size. For more information about RLIMIT_AS, refer to the description of setrlimit in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

The soft limit is obtained from MVS; if it is greater than the MAXASSIZE value, the soft limit is set to the hard limit. This value is also used when processes are initiated by a daemon process using an exec() after setuid(). In this case, both the RLIMIT_ AS hard and soft limit values are set to the MAXASSIZE specified value.

When processes are initiated by a daemon process using an exec()after setuid(), this value is used. Therefore, MAXASSIZE will be the region size for all processes created through rlogin or telnet. In this case, both the RLIMIT_AS hard and soft limit values are set to the MAXASSIZE value.

A superuser can override this value by specifying a new region size in the spawn inheritance structure on __spawn(). Or you can change the value of MAXASSIZE dynamically by using the SETOMVS or SET OMVS command. This change only affects the new processes that are created after the change was made.

Note: The IEFUSI user exit can modify the region size of an address space. Users are strongly discouraged from altering the region size of address spaces in the OMVS subsystem category.

Value Range: *nnnnn* is a decimal value from 10485760 (10 megabytes) to 2147483647 (2 gigabytes).

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

Default: 209715200

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXASSIZE. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

For planning information, see MAXASSIZE in *z/OS UNIX System Services Planning*.

MAXCORESIZE(*nnnnn***)**
Specifies the RLIMIT_CORE soft and hard resource values that will be established as the initial values for new processes. RLIMIT_CORE indicates the maximum core dump file size (in bytes) that a process can create. It also specifies the limit when they are initiated by a daemon process using an exec() after a setuid(). For more information about RLIMIT_CORE, see the description of setrlimit() in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647 (2 gigabytes).

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

Default: 4194304 (4 megabytes) Specifying a value of 2147483647 (2 gigabytes) indicates an unlimited core file size.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXCORESIZE. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

MAXCPUTIME(*nnnnn***)**
Specifies the RLIMIT_CPU resource values that will be established as the initial values for new processes. RLIMIT_CPU indicates the CPU time, in seconds, that a process can use. For more information about RLIMIT_CPU, refer to the description of setrlimit() in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

If the soft limit value from MVS is greater than the MAXCPUTIME value, the soft limit is set to the hard limit. This value is also used when processes are initiated by a daemon process using an exec() after setuid(). In this case, both the RLIMIT_CPU hard and soft limit values are set to the MAXCPUTIME value.

A superuser can override this value by specifying a new time limit in the spawn inheritance structure on __spawn().

For processes that are running in or forked from TSO or BATCH, the MAXCPUTIME value has no effect. The TIME limit is inherited from the parent. If a TIME parameter is specified on the JCL for the started task, then that value is used. If not, then the TIME value is taken from the JES default TIME value.

For processes created by the rlogind command or other daemons, MAXCPUTIME is the time limit for the address space.

Specifying a MAXCPUTIME or CPUTIMEMAX of 86400 seconds disables the JWT, SWT, or TWT timeout the same way that JCL TIME=1440 does. See "Parameters for SMFPRMxx" on page 745 for details on the JWT, SWT, and TWT parameters.

Value Range: *nnnnn* is a decimal value from 7 to 2147483647 seconds.

Default: 1000

Use the SETOMVS or SET OMVS command to dynamically increase the value of MAXCPUTIME. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For planning information, see MAXCPUTIME in *z/OS UNIX System Services Planning*.

MAXFILEPROC(*nnnnnn***)**

Specifies the maximum number of descriptors for files, sockets, directories, and any other file system objects that a single process can have concurrently active or allocated. MAXFILEPROC is the same as the OPEN_MAX variable in the POSIX standard.

Value Range: *nnnnnn* is a decimal value from 3 to 524287.

Default: 64000

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXFILEPROC. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

On the ADDUSER command, you can increase the maximum number of open descriptors on a per process basis. The FILEPROCMAX parameter on the ADDUSER command overrides the MAXFILEPROC parameter in the BPXPRMxx profile.

For planning information, see MAXFILEPROC in *z/OS UNIX System Services Planning*.

MAXFILESIZE(*nnnnn***|NOLIMIT)**

Specifies the RLIMIT_FSIZE soft and hard resource values that will be established as the initial values for new processes. RLIMIT_FSIZE indicates the maximum file size (in 4 KB increments) that a process can create. It also specifies the limit when they are initiated by a daemon process using an exec()

after a setuid(). For more information about RLIMIT_FSIZE, see the description of setrlimit() in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647, which indicates an unlimited file size. If MAXFILESIZE is not specified or MAXFILESIZE(NOLIMIT) is specified, there will be no limit to the size of files created, except for the architectural limit of the system.

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

If you specify 0, the process does not create any files. Omitting this statement indicates an unlimited file size.

Default: NOLIMIT

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXFILESIZE. To make a permanent change, edit the BPXPRMxx member that is used in IPLs.

MAXIOBUFUSER(*nnnnn***)**
Specifies that maximum amount of persistent I/O virtual storage that z/OS UNIX obtains on behalf of a user when a process performs I/O in a Unicode Services conversion environment (that is, the AUTOCVT setting is ALL). This storage remains allocated for the life of an open file. The amount that is allocated for each open depends on the CCSID of the file and the size of a read or write requests that are used by the process. The limit does not apply to UID 0 processes.

Value Range: 0 to 2 G, representing the number of megabytes of storage. Thus, 2G represents 2G x 1M = 2P (petabytes) of storage.

Default: 2048 (equivalent to 2G of storage).

MAXMMAPAREA(*nnnnn***)**
Specifies the maximum amount of data space storage space (in pages) that can be allocated for memory mappings of z/OS UNIX files. Storage is not allocated until the memory mapping is active.

Using memory map services causes a large amount of system memory to be consumed. For each page (4 KB) that is memory-mapped, 96 bytes of ESQA are consumed when a file is not shared with any other users. When a file is shared by multiple users, each user after the first causes an additional 32 bytes of ESQA to be consumed for each shared page. Assuming that the default of 40960 pages is taken, and assuming that no sharing is done by mmap() users, a maximum of 3840 KB of ESQA could be consumed. The ESQA storage is consumed when the mmap() function is invoked rather than when the page is accessed by the memory mapping application program.

If you have applications using the __MAP_MEGA option, you can map very large files without the system overhead in ESQA. For more information, see Extended System Queue Area (ESQA) in *z/OS UNIX System Services Planning*.

Value Range: *nnnnn* is a decimal value from 1 to 16777216.

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

Default: 40960

You can change the value of MAXMMAPAREA dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For planning information, see MAXMMAPARE in *z/OS UNIX System Services Planning*.

MAXPIPEUSER(*nnnnn*)
Specifies the maximum number of z/OS UNIX named or unnamed pipes that a real UID can open concurrently. The value only applies to non-UID=0 users. For UID=0 users, the MAXPIPERUSER limit of 8730 is always enforced.

Value Range: *nnnnn* is a decimal value from 256 to 8730.

Default: 8730.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXUIDS. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For planning information, see *z/OS UNIX System Services Planning*.

Syntax example in BPXPRMxx: MAXPIPEUSER(8730)

MAXPROCSYS(*nnnnn*)
Specifies the maximum number of processes that the system allows.

Value Range: *nnnnn* is a decimal value from 5 to 32767.

Default: 900

You can use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXPROCSYS. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

If you are using SETOMVS or SET OMVS to change the value, the new value must be within a certain range, or you will get an error message. The range that you can use has a minimum value of 5; the maximum value is based on the following calculation:

```
MIN(32767,MAX(4096,3*initial value))
```

The initial value is the MAXPROCSYS value that was specified during BPXPRMxx initialization. You cannot use a value less than 5. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (32767), you must change the value in BPXPRMxx and re-IPL. For an example of how to calculate the maximum value in the range, see the section on dynamically changing certain BPXPRMxx parameter values in *z/OS UNIX System Services Planning*. For planning information, also see MAXPROCSYS in *z/OS UNIX System Services Planning*.

MAXPROCUSER(*nnnnn*)
Specifies the maximum number of processes that a single z/OS UNIX user ID can have concurrently active, regardless of how the processes were created. MAXPROCUSER is the same as the CHILD_MAX variable in the POSIX standard.

A value of 25 is required for FIPS 151-2 compliance and a value of 16 is required for POSIX.1 (ISO/IEC 9945-1:1990[E] IEEE Std 1003.1-1990) standard compliance.

The number of processes is tracked by user ID (UID). When a user attempts to create a new process, the limit value for the user (defined by either the user

profile or the default OPTN value) is compared to the value maintained for the user's UID. If the user maximum is larger than the current process count for the UID, the user can create another process. If not, the user is not allowed to create a new process. For example, if user A, with a user-defined limit of 10, tries to create a process and the UID limit is already 12, user A is not allowed to create the new process. Since only 12 processes are currently created, user B, with a user-defined limit of 20, is allowed to create a new process.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the MAXPROCUSER values. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

There are certain cases when a new process can be created with a UID that has already reached the MAXPROCUSER number of processes. Super users (with UID=0) and processes blind dubbed with the default OMVS segment are not limited by MAXPROCUSER.

For planning information, see MAXPROCUSER in *z/OS UNIX System Services Planning*.

Value Range: *nnnnn* is a decimal value from 3 to 32767.

Default: 25

MAXPTYS(*nnnnn***)**
Specifies the maximum number of pseudoterminals (pseudo-TTYs or PTYs) for the system.

Value Range: *nnnnn* is a decimal value from 1 to 10000.

Default: 800

You can use the SETOMVS or SET OMVS command to dynamically increase the value of MAXPTYS. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

If you are using SETOMVS or SET OMVS to change the value, the new value must be within a certain range. If it is outside the range, you will get an error message. To use a value that is outside this range, you must change the MAXPTYS specification in BPXPRMxx and re-IPL. The range's minimum value is 1 and the maximum is based on the following calculation:

```
MIN(10000,MAX(256,2*initial value)
```

The initial value is the MAXPTYS value that was specified during BPXPRMxx initialization. For an example of how to calculate the maximum value in the range, see the section on dynamically changing BPXPRMxx values in *z/OS UNIX System Services Planning*.

For planning information, see MAXPTYS in *z/OS UNIX System Services Planning*.

MAXQUEUEDSIGS(*nnnnnn***)**
Specifies the maximum number of signals that z/OS UNIX allows to be concurrently queued within a single process.

Value Range: *nnnnnn* is a decimal value from 1 to 100000.

Default: 1000

You can change the value of MAXQUEUEDSIGS dynamically by using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

MAXSHAREPAGES(*nnnnn*)

Specifies the maximum amount of shared system storage pages that can be used by z/OS UNIX functions. The purpose of MAXSHAREPAGES is to limit the amount of ESQA storage necessary to maintain the shared pages. See Predicting and limiting ESQA usage in *z/OS UNIX System Services Planning* for the formula to determine the maximum amount of ESQA that is consumed during loading user-shared libraries.

The usage of shared pages is helpful but not critical to the loading of user shared library modules, ptrace, and fork; it serves to increase performance but does not affect functionality. As the amount of shared pages being used reaches certain limits, fewer functions are allowed to continue using them. User shared library loads, ptrace, and fork stop using shared pages when the limit reaches 60% (the only time shared storage is used by the fork service is when FORKCOPY(COW) is specified), mmap stops at 80%, and shmat, the most critical function, uses shared pages until their total capacity has been reached. If while running a 64–bit program, you allocate shared memory segments above the bar by using the shmget() service, the shared page limit is not affected.

Because each page of shared storage requires the associated consumption of extended system queue area (ESQA) storage, limiting the shared storage usage provides a way to limit the ESQA usage by z/OS UNIX users. If you use the __IPC_MEGA or __MAP_MEGA options, then the shared pages limits are not affected because MEGA does not affect the system ESQA overhead.

Note: Evaluate adjusting MAXSHAREPAGES on an active system. Dynamically decreasing the number of pages available to ESQA for active work can cause errors. This is due to the fact that for those jobs, the ESQA limit may now be reached or exceeded. It is possible that shared programs will not be able to be loaded and fork() may not succeed. This situation will exist until the workload adjusts to the new lower limit.

Value Range: *nnnnn* is a decimal value from 0 to 32768000 specifying a number of 4K pages.

You can set a denomination (or multiplier) value when defining this value. The C suffix can have a one-character value as presented in Table 14 on page 139, but must not exceed the parameter-specific upper limit. The denomination value is retained and is used again within a subsequent D OMVS command.

Default: 131072

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the MAXSHAREPAGES value. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

MAXTHREADS(*nnnnnn*)

Specifies the maximum number of pthread_created threads, including running, queued, and exited but undetached, that a single process can have concurrently active. Specifying a value of 0 prevents applications from using pthread_create.

Value Range: *nnnnnn* is a decimal value from 0 to 100000.

Default: 200

You can change the value of MAXTHREADS dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For planning information, see MAXTHREADS in *z/OS UNIX System Services Planning*.

MAXTHREADTASKS(*nnnnn***)**
Specifies the maximum number of MVS tasks that a single process can have concurrently active for pthread_created threads.

Value Range: *nnnnn* is a decimal value from 0 to 32768.

Default: 1000

You can change the value of MAXTHREADTASKS dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

For planning information, see MAXTHREADTASKS in *z/OS UNIX System Services Planning*.

MAXUIDS(*nnnnn***)**
Specifies the maximum number of z/OS UNIX user IDs (UIDs) that can operate concurrently.

Value Range: *nnnnn* is a decimal value from 1 to 32767.

Default: 200

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXUIDS. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For planning information, see the section on MAXUIDS in *z/OS UNIX System Services Planning*.

MAXUSERMOUNTSYS(*nnnn***)**
Specifies the maximum number of nonprivileged user mounts for the system or for the shared file system configuration environment. The MAXUSERMOUNTSYS limit applies only to nonprivileged users and does not affect privileged mounts.

Value Range: *nnnnn* is a decimal value from 0 to 35000 that indicates the maximum number of nonprivileged user mounts allowed in the system. For those using a shared file system configuration, this value is the maximum number of nonprivileged user mounts allowed in the shared file system environment. The most recent specification prevails for all of the systems that are participating in a shared file configuration.

Default: 0.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXUSERMOUNTSYS. To make a permanent change, edit the BPXPRMxx member that is used for IPLs. If MAXUSERMOUNTSYS is not specified, default values are used. The default value indicates that user mounts are not allowed in the system.

For planning information, see MAXUSERMOUNTSYS in *z/OS UNIX System Services Planning*.

```
MOUNT FILESYSTEM('fsname') DDNAME(ddname) TYPE(type_name)
MOUNTPOINT('pathname') MODE(access) PARM('parameter')
TAG(NOTEXT|TEXT,ccsid) SETUID|NOSETUID SECURITY|NOSECURITY
AUTOMOVE[(INCLUDE,sysname1,sysname2,...,sysnamen|*)]
[(EXCLUDE,sysname1,sysname2,...,sysnamen)]
```

|NOAUTOMOVE|UNMOUNT SYSNAME(sysname) MKDIR('pathname')

Specifies a file system that z/OS UNIX is to logically mount onto the root file system or another file system.

Mount statements are processed in the sequence in which they appear. If they are cascading, the system mounts the first file system first. Make sure that a mount point exists before the file system is mounted. If you mount a file system over an existing directory that contains files, you will cover up the existing files.

If a MOUNT statement uses a DDNAME parameter to identify the HFS data set, allocate that HFS data set in the OMVS cataloged procedure. See the section on customizing the OMVS cataloged procedure to run the kernel initialization program in *z/OS UNIX System Services Planning*.

The MOUNT statement is optional; the BPXPRMxx member can contain one or more MOUNT statements. The MOUNT parameters are:

AUTOMOVE[(INCLUDE|EXCLUDE,sysname1,sysname2,...,{sysnameN| *})]|NOAUTOMOVE|UNMOUNT

The AUTOMOVE, NOAUTOMOVE, and UNMOUNT parameters apply only in a sysplex where systems are participating in shared file system. The parameters indicate what happens if the system that owns a file system goes down.

AUTOMOVE indicates that ownership of the file system automatically changes to another system that is participating in shared file system. You can specify AUTOMOVE on its own to allow the system to randomly select a new owner for the file system. You can direct the system how to choose a new owner for the file system by using the indicators INCLUDE (I) or EXCLUDE (E).

Specify INCLUDE with a system list to provide a prioritized list of systems to which the file system can be moved if the owning system goes down. For example, AUTOMOVE(INCLUDE,SYS1,SYS4,SYS9) tells the system that the file system can be moved to SYS1, SYS4, or SYS9, in that order, or AUTOMOVE(INCLUDE,SYS1,SYS4,*) tells the system that the file system can be moved to SYS1 or SYS4, in that order, then to any other available system. This selection is not totally random; rather, MVS attempts to move the file system to a new server system in which the file system is actively in use.

Restriction: If specified, the asterisk must be listed last or listed as the only system name in the INCLUDE system list.

Specify EXCLUDE with a system list to provide a list of systems to which the file system cannot be moved. For example, AUTOMOVE(EXCLUDE,SYS3,SYS5,SYS7) tells the system that the file system can be moved to any system except SYS3, SYS5, and SYS7. If the file system cannot be moved as you directed in the system list, the file system is unmounted when the owning system goes down.

NOAUTOMOVE indicates that ownership of the file system is not moved in some situations; as a result, the file system becomes inaccessible.

Note: When specifying NOAUTOMOVE, although the file system becomes inaccessible when the owning system unexpectedly goes down, it still exists in the file system hierarchy. The file system remains unowned until the original owning system re-IPLs. Changing the AUTOMOVE value in BPXPRMxx of an unowned file system before re-IPLing will not change the

AUTOMOVE value of this file system since it is already mounted. To change the AUTOMOVE value, the file system must be unmounted before the IPL.

UNMOUNT indicates that the file system should be unmounted in certain situations.

See *z/OS UNIX System Services Planning* for more information about the behavior of the AUTOMOVE options.

Note:

1. Use AUTOMOVE for the version file system and the sysplex root file system.
2. For file systems that are associated with a single system, specify UNMOUNT. The file systems include `/etc`,`/tmp`, `/var`, `/dev`, and the system-specific file system. For descriptions of the sysplex root, system-specific, and version file systems, see the section on sharing file systems in a sysplex in *z/OS UNIX System Services Planning*.
3. For sysplex-unaware file systems that are mostly exported by the DFS or SMB server to their remote clients, consider specifying NOAUTOMOVE on the MOUNT statement. By doing so, the file systems will not change ownership if the system is suddenly recycled and they are available for automatic re-export by DFS or SMB.

   Consider specifying NOAUTOMOVE because a file system can only be exported by the DFS or SMB server at the system that owns the file system. After a file system is exported by DFS, it cannot be moved until it has been unexported by DFS. The same holds true of file systems that are exported by SMB. When recovering from system outages, you need to weigh sysplex availability against availability to the DFS or SMB clients. When an owning system recycles and a file system that is exported by DFS or SMB is taken over by one of the other systems, DFS or SMB cannot automatically re-export that file system. When an owning system is recycled and an exported file system is taken over by one of the other systems, that file system is not automatically reexported. The file system must be moved from its current owner back to the original system, the one that has just been recycled, and then exported again.

Default: AUTOMOVE

DDNAME(ddname)
The ddname on the JCL DD statement that defines the file system. To use the DDNAME parameter, a DD statement for the HFS data set containing the mountable file system should be placed in the OMVS cataloged procedure. Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 8 characters; the system converts the ddname to uppercase.

Restriction: zFS does not support DDNAME; the FILESYSTEM keyword must be used.

FILESYSTEM('fsname')
The name of the file system. The name must be unique in the system.

In the FILESYSTYPE statement, you can mount the following file systems:

- HFS for a hierarchical file system (HFS). See the section on managing hierarchical file system data sets in *z/OS DFSMSdfp Advanced Services*.

- zFS for a z/OS File System (zFS). See the mount section in *z/OS Distributed File Service zFS Administration*.
- NFS for accessing remote files. See the section on mount processing parameters in *z/OS Network File System Guide and Reference*.
- TFS for a temporary file system (TFS). See the section on mounting the TFS in *z/OS UNIX System Services Planning*.

Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 44 characters; the characters can be in uppercase, lowercase, or both. The name must be enclosed in single quotation marks. An HFS data set name must conform to the rules of MVS data set names.

MKDIR('pathname')
Specifies the name of a directory that the system will create dynamically after the file system has been successfully mounted. This allows the system to create mount points that subsequent MOUNT statements can reference. MKDIR is an optional keyword.

You can specify more than one MKDIR keyword on each ROOT or MOUNT statement. The directories are created in the order they are listed. Currently, the maximum number of MKDIR statement that is allowed for each ROOT or MOUNT statement is 50.

You must specify a relative path name; the path name cannot start with a slash (/). Enclose the path name in single quotation marks.

The path name is relative to the file system mount point specified on the MOUNTPOINT keyword.

The path name can contain intermediate directories, but these intermediate directories must exist in the file system hierarchy. If these intermediate directories do not exist, you can use additional MKDIR keywords to create the necessary intermediate directories.

The directory to be created must be in a file system that was mounted as RDWR. The permission bits for the created directory will be 755. The UID and GID is inherited from this directory's parent. These attributes are overlaid when this directory is used as a mount point.

Do not use the MKDIR keyword for file systems that mount asynchronously, such as the NFS file system. When a file system is mounted asynchronously, message BPXF025I is issued to the system log. Similarly, do not use MKDIR with the SYSNAME keyword, when SYSNAME identifies a remote system to perform the mount. The results are unpredictable.

A failure to create a directory does not cause a failure of the mount. A message is written to the system log when a problem occurs during the creation of the directory. If the directory exists, no message is written.

MODE(access)
Specifies access to the mounted file system by all users:

- READ: Users can only read the file system that is being mounted.
- RDWR: Users can read and write in the file system that is being mounted.

Default: RDWR

MOUNTPOINT('pathname')
Specifies the absolute path name, or a symbolic link that resolves to the path name of the directory onto which the file system is to be mounted.

Mount point restrictions are as follows:

- The mount point must be a directory.
- Any files in the directory are not accessible while the file system is mounted.
- Only one mount can be active at any time for a mount point.
- A file system can be mounted at only one directory at any time.

MOUNTPOINT is required and must specify an absolute path name. The path name is up to 1023 characters long; the characters can be in uppercase, lowercase, or both.

PARM('parameter')
Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 500 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotation marks.

Note: For information about specifying the TFS configuration parameters, see the section on parameter key options for the mount statement and mount command in *z/OS UNIX System Services Planning*. For example:

-ea *count*
Allows the TFS file system to automatically grow count times.

-em *count*
Allows the TFS file system to manually grow count times.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

SYNC(*t*), NOWRITEPROTECT, NOSPARSE, and SYNCRESERVE are valid only when ENTRYPOINT is GFUAINIT. FSFULL(*threshold*,*increment*) is supported by HFS, TFS, and z/OS File System (zFS).

Note: If a syntax error is found in any of these parameters (SYNC(t), NOWRITEPROTECT, NOSPARSE, FSFULL, and SYNCRESERVE), an error message is issued and all five parameters are set to the default values.

- **SYNC(t)**
  - *t* specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the metadata for a file system is hardened. Setting *t* to 0 indicates that the file system should harden metadata synchronously with syscall requests.
  - Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.
  - The maximum value that can be specified for *t* is 65535. Values between 65535 and 99999 are rejected.
  - A value of 99999 specifies that no sync daemon intervals are specified, and thus, the metadata is not hardened.
  - **Default:** 60 seconds
- **NOWRITEPROTECT**
  - This keyword overrides the WRITEPROTECT function. When NOWRITEPROTECT is specified, the file system is not protected from

being read/write mounted by multiple systems simultaneously. Read/write mounting by multiple systems corrupts the file system.

Extreme care should be taken when specifying this keyword. It should only be used when there is no possibility of the file system that is being mounted by multiple systems.

Use of the NOWRITEPROTECT keyword avoids an additional file system read operation that is required at Sync time to support the WRITEPROTECT function.

– **Default:** WRITEPROTECT

- **NOSPARSE(DUMP | LOGREC)**
  – When NOSPARSE is specified on the MOUNT statement, HFS will not allow any files in that file system to be sparse. A file becomes sparse when all of the data cannot be written. For example, suppose we are only able to write the first 10,000 bytes of a file, and then the system has to lseek out to offset 50,000 and resume writing from there. The file is considered sparse because bytes 10,000-50,000 were never written to the file. If the user attempts to read bytes 10,000 to 50,000, binary 0's will be returned as the value. NOSPARSE handles this by causing a dump to be taken or LOGREC record to be created when either of the following situations occur:
    - The file system attempts to read metadata from disk for a file and detects that the subject file is sparse, or
    - An application attempts to write to a page beyond the end of the file, causing the file to become sparse.
  – DUMP causes a dump to be created. Only one dump is created for each of the possible reason codes while a file system is mounted. DUMP is the default if you specify NOSPARSE without the DUMP or LOGREC keywords.
  – LOGREC causes a LOGREC record to be written. A dump is not created.
  – **Default:** DUMP
- **FSFULL(threshold,increment)**
  – *threshold* specifies the percentage of the file system (HFS, TFS, or zFS) capacity at which an operator message is generated. The default is 100%.
  – *increment* specifies the percentage of change above the HFS or zFS capacity at which an operator message is generated. Messages are generated by either an increase or decrease greater than *increment* . The default is 5%.

  You can specify *threshold* and *increment* values for all file systems. The values can also be set on the MOUNT command for a specific file system. Parameters on the MOUNT command override parmlib values. If no values are specified in either place, no threshold checking is done. If a threshold value is specified but no increment is given, the increment defaults to 5%. The increment value applies both to upgrading the message when the file system continues to fill and to removing the message when more space becomes available due to either deleting files, or to extending the file system. The values are in terms of percent full. The values that are applied to a file system can be changed only when the file system is mounted.
- **SYNCRESERVE(nn)**

- This keyword controls the number of pages to be reserved for sync processing of the file system metadata.*nn* represents the percentage of the file system space that is to be reserved for the sync shadow write mechanism. *nn* is a decimal number between 1 and 50. There is no reason to reserve more than 50% of the file system space, because the reserved space must always be less than the actual index size and the index size plus the reserved space cannot be greater than the file system space.
- When this parameter is specified on the MOUNT statement, it overrides the internal reserved page estimation algorithm. Only use this parameter if the internal algorithm is not providing the desired results.

SYSNAME(sysname)

For systems that are participating in a shared file system, SYSNAME specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. This system must be IPLed with SYSPLEX(YES).

Default: The name of the system, if IPLed with SYSPLEX(YES), that the mount is processed on.

Note:

1. Use the defaults for SYSNAME and AUTOMOVE to ensure that the root is always available.
2. Only specify a SYSNAME() value if you want only the specified system to be the file system owner.
3. During z/OS UNIX initialization processing the MOUNT statement is ignored if SYSNAME() specifies another system.
4. For SET OMVS and SETOMVS processing, the MOUNT statement is processed and the MOUNT is function-shipped to the system specified by SYSNAME(). If SYSNAME() is used with a value that resolves to another system, do not include any subsequent parmlib MOUNT statements that specify a MOUNTPOINT() with a path name that includes a directory in this file system.

TAG (NOTEXT|TEXT,*ccsid*)

Specifies whether implicit file tags are assigned to untagged files in the mounted file system. File tagging controls whether a file's data can be converted during file reading and writing. "Implicit" in this case means that the tag is not permanently stored with the file. Instead, the tag is associated with the file during reading and writing, or when stat() type functions are issued. Either TEXT, or NOTEXT, and *ccsid* must be specified when TAG is specified.

NOTEXT specifies that none of the files in the file system are automatically converted during file reading and writing.

TEXT specifies that each untagged file is implicitly marked as containing pure text data that can be converted.

ccsid names the coded character set identifier to be implicitly set for the untagged file. *ccsid* is specified as a decimal value from 0 to 65536. However, when TEXT is specified, the values of 0 and 65536 are illegal because those values imply no conversion. Other than this, the value is not checked as being valid and the corresponding code page is not checked as being installed.

For example:

- TAG(TEXT,819) identifies text files that contain ASCII (ISO-8859-1) data.
- TAG(TEXT,1047) identifies text files that contain EBCDIC ((ISO-1047) data.
- TAG(NOTEXT,65536) tags files as containing binary or unknown data.
- TAG(NOTEXT,0) is the equivalent of not specifying the TAG parameter.
- TAG(NOTEXT,273) tags file with the German code set (ISO-273) but is not eligible for automatic conversion.

Default: NOTEXT

SECURITY|NOSECURITY
SECURITY specifies that security checks should be performed. NOSECURITY specifies that security checks should not be performed.

Default: SECURITY

SETUID|NOSETUID
SETUID specifies that the setuid() and setgid() mode bit on an executable file will be supported.

NOSETUID specifies that the setuid() and setgid() mode bit on an executable file will not be supported. The UID or GID will not be changed when the program is executed and the APF and Program Control extended attributes are not honored. The entire HFS is uncontrolled.

Default: SETUID

TYPE(type_name)
Specifies the name of a file system type that is identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement. TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

For additional information, see the MOUNT section in *z/OS UNIX System Services Planning*.

NETWORK DOMAINNAME(sockets_domain_name) DOMAINNUMBER(sockets_domain_number) MAXSOCKETS(number) TYPE(type_name) INADDRANYPORT(starting_port_number) INADDRANYCOUNT(number_of_ports_to_reserve)
Specifies that a socket physical file system domain should be readied for use. The TYPE in this statement matches the TYPE on the previous FILESYSTYPE statement.

Use the SETOMVS RESET command to dynamically change the MAXSOCKET value or add a new NETWORK. To make a permanent change, edit the BPXPRMxx member that is used for IPLs. For more information, see the section on dynamically adding FILESYSTYPE statements in BPXPRMxx in *z/OS UNIX System Services Planning*.

Provide a NETWORK statement for each socket file system domain to be initialized.

- For AF_UNIX file systems, always include a FILESYSTYPE statement that specifies ENTRYPOINT(BPXTUINT) and a NETWORK statement with a matching TYPE, usually TYPE(UDS), on both.
- For TCP/IP sockets, always include a FILESYSTYPE statement that specifies ENTRYPOINT(EZBPFINI) and a NETWORK statement with a matching TYPE, usually TYPE(INET), on both.

- To activate an Internet Protocol Version 6 (IPv6) socket on a system, you must configure both the AF_INET domain and the AF_INET6 domain. You cannot code a NETWORK statement for domain name AF_INET6 without coding a NETWORK statement for domain name AF_INET.
- For CINET sockets, include a FILESYSTYPE statement with ENTRYPOINT (BPXTCINT) and a NETWORK statement with a matching TYPE, usually TYPE(CINET), that specifies INADDRANYPORT and INADDRANYCOUNT. See the section on specifying INADDRANYPORT and INADDRANYCOUNT in *z/OS UNIX System Services Planning* for more information.

DOMAINNAME(sockets_domain_name)
The 1 to 16 character name by which this socket file system domain is to be known.

DOMAINNUMBER(sockets_domain_number)
A number that matches the value that is defined for this domain name. The currently supported values for this field are:
1 AF_UNIX
2 AF_INET
19 AF_INET6

Table 15 shows some supported domain names, domain numbers, and their associated entry point names. See the documentation for the physical file system you are using to get the correct entry point name.

Table 15. Supported domains

Domain name	Domain number	Entry point
AF_UNIX	1	BPXTUINT
AF_INET	2	EZBPFINI, BPXTCINT
AF_INET6	19	EZBPFINI, BPXTCINT

INADDRANYCOUNT(number_of_ports_to_reserve)
Specifies the number of ports that the system reserves, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET.

Value Range: *number_of_ports_to_reserve* is a decimal value from 1 to 8000.

Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYCOUNT is 1000. Otherwise, no ports are reserved (0).

INADDRANYPORT(starting_port_number)
Specifies the starting port number for the range of port numbers that the system reserves for use with PORT 0, INADDR_ANY binds. This value is only needed for CINET.

Value Range: *starting_port_number* is a decimal value from 1024 to 65534. Ports 1 — 1023 are well-known ports that cannot be reserved for use with PORT 0, INADDR_ANY binds.

Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYPORT is 63000. Otherwise, no ports are reserved (0).

Note: If you do not want to support INADDRANY with CINET, you should specify INADDRANYPORT(*xx*), where *xx* is a valid value, without specifying INADDRANYCOUNT.

Note: When activating IPv6 on a system, the INADDRANYPORT is shared across domains. The INADDRANYPORT value is taken from the NETWORK statement for the AF_INET domain. Any INADDRANYPORT value specified for the AF_INET6 domain is ignored.

MAXSOCKETS(*nnnnn*)
Specifies the maximum number of sockets supported by this file system for this address family. You can specify a value from 0 to 16777215. This is an optional parameter. The maximum value that this field can have is defined by each domain. If a value larger than the maximum is specified, an informational message is issued and the value used is the maximum.

Note: Ensure that this number is large enough for socket connections for all applications using your z/OS UNIX environment. This upper limit is set when the NETWORK statement is processed during IPL. It can only be changed if the NETWORK statement is changed using the SETOMVS RESET command.

When the Common Event Adapter (CEA) is active, each CEA client requires two socket connections for communication with the CEA server.

For AF_UNIX, MAXSOCKETS will be ignored if it is specified. The maximum number of AF_UNIX sockets is 10000.

When activating IPv6 on a system, you can specify separate MAXSOCKETS values for domains AF_INET and AF_INET6. If you do not specify a MAXSOCKETS value for the AF_INET6 domain, the default will be the MAXSOCKETS value specified or defaulted to for the AF_INET domain.

Table 16 shows the maximum and default values for MAXSOCKETS by domain.

Table 16. Maximum and default values for MAXSOCKETS by domain

Domain name	MAXSOCKETS maximum	MAXSOCKETS default
AF_UNIX	10000	10000
AF_INET	16777215	64000
AF_INET6	16777215	Same as the current value for AF_INET

TYPE(type_name)
Specifies the name of a file system type that is identified in a FILESYSTYPE statement. The TYPE(type_name) must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement. TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

NONEMPTYMOUNTPT(NOWARN|WARN|DENY)
Specifies how the system is to mount any file system on a mount point when it is a non-empty directory.

NOWARN causes the system to mount any file system on mount point without any warning message when the mount point is a non-empty directory. The contents of that directory are hidden for the duration of the mount.

WARN causes the system to mount any file system on mount point with a warning message when the mount point is a non-empty directory. The contents of that directory are hidden during the mount.

DENY causes the system not to mount any file system when the mount point is a non-empty directory.

Default: NOWARN

If NONEMPTYMOUNTPT is not specified, default values are used.

For planning information, see NONEMTPYMOUNTPT in *z/OS UNIX System Services Planning*.

PRIORITYGOAL(service_class_name1,...service_class_name40)
Specifies a list of 1 to 40 service class names of 8 characters or less separated by commas, which are used in association with the setpriority, nice and chpriority callable services when the system is running in goal mode. These functions allow a program to alter the priority of one or more processes.

Generally, it is recommended that you not set PRIORITYGOAL unless the nice(), setpriority() or chpriority() values is enabled.

If the list has less than 40 entries, the system propagates the last service class specified into the remaining unspecified entries in the table. For example:

```
PRIORITYGOAL(CICS4,CICS4,CICS4,CICS3,CICS2,CICS1,TSO2,TSO1,BAT3,BAT2)
```

If you do not specify this statement, arrays are not created for it. All service classes that are specified on the PRIORITYGOAL statement must also be specified in your workload manager service policy.

PRIORITYGOAL(NONE) means that there are no values. If you do not specify PRIORITYGOAL, that means that there are no values.

If you do not want to allow users to increase the priority but still want to enable the **nice()** and **setpriority()** functions, define a range of service classes with priority increments on a base that is normal for the users. Using these functions lets the user order the priority of processes, but will not let a user improve performance over that of other users.

Value Range: *service_class_name* is a 1 to 8 character value.

Default: None

You can dynamically change the values of PRIORITYGOAL by using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

PWT(SMF|SMFENV|ENV)
Allows installations to time out z/OS UNIX processes that are waiting on terminal activity. When specified, the timeout value applies to all z/OS UNIX processes that are waiting on terminal activity. To override that value for a specific process, the _BPXK_TIMEOUT environment value can be set for an individual process. The PWT value can be changed with the SETOMVS PWT operator command. Unless a BPXK_TIMEOUT value is set, timeouts do not occur.

Specify SMF to honor the SMPFPRMxx JWT|TWT|SWT values for the timeout value. The _BPXK_TIMEOUT environment value is ignored. An individual process cannot override the system settings.

Specify SMFENV to honor the SMFPRMxx JWT|TWT|SWT value and to allow the _BPXK_TIMEOUT settings to override the setting.

Specify ENV to allow the _BPXK_TIMEOUT variable be set. Only processes setting the _BPXK_TIMEOUT value are timed out.

Default: ENV

RESOLVER_PROC(*procname*|DEFAULT|NONE)

Specifies how the resolver address space is processed during z/OS UNIX initialization. The resolver is used by TCP/IP applications for name-to-address or address-to-name resolution. In order to create a resolver address space, a system must be configured with an AF_INET or AF_INET6 domain.

procname is the name of the address space for the resolver and the procedure member name in the appropriate proclib. *procname* is one to eight characters long. The procedure must reside in a data set that is specified by the MSTJCLxx parmlib member's IEFPDSI DD card specification.

DEFAULT causes an address space named RESOLVER to start, using the system default procedure of IEESYSAS. The address space is started with SUB=MSTR so that it runs under the MASTER address space instead of the JES address space.

NONE specifies that no address space is to be started. If you are using z/OS Communications Server IP, the resolver must be started before TCP/IP can be started. TCP/IP does not initialize until the resolver address space is started.

RUNOPTS('string')

Specifies the _CEE_RUNOPTS environment variable that is used when z/OS UNIX initialization invokes `/etc/init` or `/usr/sbin/init`. z/OS UNIX passes the _CEE_RUNOPTS value and all programs that are invoked from `/etc/rc` to the shell.

If you want to change the value of RUNOPTS, you will must edit the BPXPRMxx member and then re-IPL. You cannot use the SET OMVS or SETOMVS command to change the value. After the value is specified in BPXPRMxx, you can use one of the following methods to change this string:

- The system is re-IPLed with a new BPXPRMxx RUNOPTS string.
- The user or installation sets _CEE_RUNOPTS in `/etc/rc` or `/etc/init.config`.
- A program or shell script sets _CEE_RUNOPTS.

If you do not specify a value for RUNOPTS, the RUNOPTS string or _CEE_RUNOPTS environment variable is not provided.

The TSO/E OMVS command uses the specified options as the Language Environment runtime options, by default.

The setting of RUNOPTS has no effect on BPXBATCH jobs.

Specifying the RUNOPTS parameter causes the kernel to set the _CEE_RUNOPTS environment variable when starting `/etc/init`, or when the TSO/E OMVS command is entered. This environment variable is normally propagated to subsequent processes (such as `/etc/init` to `/bin/sh` to `/etc/rc` to `/bin/inetd` to `/bin/rlogind` to `/bin/sh` for shell users).

To do this, you must make sure that any other steps in the flow (such as export statements in `/etc/rc`) do not overwrite the value of _CEE_RUNOPTS. If more runtime options are needed, they should be concatenated to the old value of _CEE_RUNOPTS.

Value Range: From 1 to 250 characters.

Default: No RUNOPTS string or _CEE_RUNOPTS environment variable is provided.

Restrictions:

- The string must be enclosed in parentheses and quotation marks (").

- An empty string (' ') is not valid.
- Although all characters are allowed, nulls, slashes (/), unbalanced SO/SI, and unbalanced parentheses and quotation marks cause unpredictable problems in areas such as the TSO/E OMVS command.

For more information about specifying RUNOPTS strings, see Customizing the BPXPRMxx parmlib member in *z/OS UNIX System Services Planning*.

ROOT FILESYSTEM('fsname') DDNAME(ddname) TYPE(type_name) MODE(access) PARM('parameter') SETUID|NOSETUID AUTOMOVE|NOAUTOMOVE SYSNAME(sysname) TAG(NOTEXT|TEXT,*ccsid*) MKDIR('pathname')

Specifies a file system that z/OS UNIX is to logically mount as the root file system. It is optional. If it is not specified, a TFS file system is mounted as the root.

To change the value of the ROOT statement without having to re-IPL, use the TSO/E MOUNT and UNMOUNT commands.

The root file system can be unmounted using the TSO/E UNMOUNT command or ISHELL. Ensure that you specify the IMMEDIATE option.

The ROOT statement does not support the SYSNAME() keyword.

The parameters are as follows:

DDNAME(ddname)

The ddname on the JCL DD statement that defines the root file system. To use the DDNAME parameter, a DD statement for the HFS data set containing the root file system should be placed in the z/OS UNIX cataloged procedure.

Either FILESYSTEM or DDNAME is required; do not specify both. The ddname is 1 to 8 characters; the system converts the ddname to uppercase.

Restriction: zFS does not support DDNAME; the FILESYSTEM keyword must be used.

FILESYSTEM('fsname')

The name of the root file system. The name must be unique in the system.

You can specify the following types of the file system:

- HFS for a hierarchical file system (HFS).
- zFS for a z/OS File System (zFS).
- NFS for accessing remote files.
- TFS for a temporary file system (TFS).

Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 44 characters; the characters can be in uppercase, lowercase, or both. The name must be enclosed in single quotation marks. An HFS data set name must conform to the rules of MVS data set names.

MKDIR('pathname')

Specifies the name of a directory that the system will create dynamically after the file system is successfully mounted. This allows the system to create mount points that can be referenced by subsequent MOUNT statements. MKDIR is an optional keyword.

You can specify more than one MKDIR keyword on each ROOT statement. The directories are created in the order they are listed.

You must specify a relative path name; the path name cannot start with a slash (/). Enclose the path name in single quotation marks.

The path name is relative to the file system mount point specified on the MOUNTPOINT keyword.

The path name can contain intermediate directories, but these intermediate directories must exist in the file system hierarchy. If these intermediate directories do not exist, you can use additional MKDIR keywords to create the necessary intermediate directories.

The directory to be created must reside in a file system that was mounted as RDWR. The permission bits for the created directory is 755. The UID and GID is inherited from this directory's parent. These attributes will be overlaid when this directory is used as a mount point.

Do not use the MKDIR keyword for file systems that mount asynchronously, such as the NFS file system. When a file system is mounted asynchronously, message BPXF025I is issued to the system log. Also, do not use MKDIR with the SYSNAME keyword, when SYSNAME identifies a remote system to perform the mount. The results are unpredictable.

A failure to create a directory does not cause a failure of the mount. A message is written to the system log when a problem occurs when the directory is created. If the directory exists, no message is written.

Restriction: MKDIR works only on systems that are at z/OS Version 1 Release 5 level or later. If you are using MKDIR in a sysplex that shares a common BPXPRMxx member, make sure that all systems are at least at the z/OS V1R5 level.

MODE(access)
Specifies access to the root file system by all users:

- READ: Users can only read the root file system.
- RDWR: Users can read and write in the root file system.

Default: RDWR

PARM('parameter')
Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 500 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotation marks.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

SYNC(*t*), NOWRITEPROTECT, NOSPARSE, and SYNCRESERVE are valid only when ENTRYPOINT is GFUAINIT. FSFULL(*threshold*,*increment*) is supported by HFS, TFS, and zFS.

Note: If a syntax error is found in any of these parameters (SYNC(*t*, NOWRITEPROTECT, NOSPARSE, FSFULL, and SYNCRESERVE), an error message is issued and all five parameters are set to the default values.

- **SYNC**(*t*)
  - *t* specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the metadata for a file system is hardened. Setting *t* to 0 indicates that the file system should harden metadata synchronously with syscall requests.

  - Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.
  - The maximum value that can be specified for *t* is 65535. Values between 65535 and 99999 are rejected.
  - A value of 99999 specifies that no sync daemon intervals are specified, and thus, the metadata is not hardened.
  - **Default:** 60 seconds

- **NOWRITEPROTECT**
  - This keyword overrides the WRITEPROTECT function. When NOWRITEPROTECT is specified, the file system is not protected from being read/write mounted by multiple systems simultaneously. Read/write mounting by multiple systems corrupts the file system.

    Extreme care should be taken when specifying this keyword. Use it only when there is no possibility of the file system being mounted by multiple systems.

    Use of the NOWRITEPROTECT keyword avoids an additional file system read operation that is required at Sync time to support the WRITEPROTECT function.
  - **Default:** WRITEPROTECT
- **NOSPARSE**(**DUMP** | **LOGREC**)
  - When NOSPARSE is specified on the MOUNT statement, files are not allow to be sparse. A file becomes sparse when all of the data cannot be written. For example, suppose we are only able to write the first 10,000 bytes of a file, and then the system has to lseek out to offset 50,000 and resume writing from there. The file is considered sparse because bytes 10,000-50,000 were never written to the file. If the user attempts to read bytes 10,000 to 50,000, binary 0's are returned as the value. NOSPARSE handles this by causing the file system to create a dump or a LOGREC record when either of the following situations occur:
    - The file system attempts to read metadata from disk for a file and detects that the subject file is sparse.
    - An application attempts to write to a page beyond the end of the file, causing the file to become sparse.
  - DUMP causes the file system to create a dump. Only one dump is created for each of the possible reason codes while a file system is mounted. DUMP is the default if you specify NOSPARSE without the DUMP or LOGREC keywords.
  - LOGREC will cause the file system to write a LOGREC record instead of creating a dump.
  - **Default:** DUMP
- **FSFULL**(*threshold*,*increment*)
  - *threshold* specifies the percentage of the file system (HFS, TFS, or zFS) capacity at which an operator message is generated. The default is 100%.
  - *increment* specifies the percentage of change above the file system capacity at which an operator message is generated. Messages are generated by either an increase or decrease greater than *increment*. The default is 5%.

You can specify *threshold* and *increment* values for all file systems. The values can also be set on the MOUNT command for a specific file system. Parameters on the MOUNT command override parmlib values. If no values are specified in either place, no threshold checking is done. If a threshold value is specified but no increment is given, the increment defaults to 5%. The increment value applies both to upgrading the message when the file system continues to fill and to removing the message when more space becomes available due to either deleting files, or to extending the file system. The values are in terms of percent full. The values that are applied to a file system can be changed only when the file system is mounted.

- **SYNCRESERVE**(*nn*)
  - This keyword controls the number of pages to be reserved for sync processing of the file system metadata.*nn* represents the percentage of the file system space which is to be reserved for the sync shadow write mechanism. *nn* is a decimal number between 1 and 50. There is no reason to ever reserve more than 50% of the file system space, because the reserved space must always be less than the actual index size and the index size plus the reserved space cannot be greater than the file system space.
  - When this parameter is specified on the MOUNT statement, it overrides the internal reserved page estimation algorithm. Only use it if the internal algorithm does not provide the expected results.

SERV_LINKLIB('dsname', 'volser')

Specifies the target service library where the UNIX System Services modules that are normally loaded from SYS1.LINKLIB into the private area of the OMVS address space are located.

Value Range: *dsname* is a 1-to-44 character value that represents a valid MVS load library data set name. The alphabetic characters in the load library name must be uppercase. *volser* is a 1-to-6 character value that represents a valid volume serial number for the volume that contains the specified MVS load library. The alphabetic characters in the volume serial number must be uppercase.

You can change the value of SERV_LINKLIB dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

SERV_LPALIB('dsname', 'volser')

Specifies the target service library where the UNIX System Services modules that are normally built into LPA are located.

Value Range: *dsname* is a 1-to-44 character value that represents a valid MVS load library data set name. The alphabetic characters in the load library name must be uppercase. *volser* is a 1-to-6 character value that represents a valid volume serial number for the volume that contains the specified MVS load library. The alphabetic characters in the volume serial number must be uppercase.

You can change the value of SERV_LPALIB dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

SETUID|NOSETUID

SETUID specifies that the setuid() and setgid() mode bit on an executable file will be supported.

NOSETUID specifies that the setuid() and setgid() mode bit on an executable file will not be supported. The UID or GID will not be changed when the program is executed and the APF and program control extended attributes are not honored. The entire file system is uncontrolled.

Default: SETUID

SHRLIBMAXPAGES(*nnnnn***)**
If you specify the SHRLIBMAXPAGES parameter, it is accepted but will not have any impact on the system. The value that you specify will never be reached, because user-shared library objects are no longer supported. This parameter is intended to control the maximum number of pages that can be allocated in the system to contain user shared library modules. This value, used with MAXSHAREPAGES, can be used to control the amount of ESQA consumed by user shared library modules. Refer to *z/OS UNIX System Services Planning* for more details.

Value Range: *nnnnn* is a decimal value between 1 and 16777216 specifying a number of 4K pages.

Default: 4096

SHRLIBRGNSIZE(*nnnnn***)**
Specifies the maximum size of the shared library region for address spaces that load system shared library modules. For these address spaces, the size that is specified is allocated from high private storage and is used for the loading of system shared library modules. This storage is not allocated in an address space until it loads a system shared library module. This parameter applies to modules loaded from system shared libraries, which allocate storage on megabyte boundaries. Therefore, this storage does not count against the MAXSHAREPAGES limit, and does not consume ESQA.

Value Range: *nnnnn* is a decimal value between 16777216 (16 megabytes) and 1610612736 (1.5 gigabytes).

Default: 67108864

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the SHRLIBRGNSIZE value. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

Because the shared library region is allocated in megabytes, the value for SHRLIBRGNSIZE must be evenly divisible by 1048576. Otherwise, the D OMVS,L command might show values in the Current Usage and Highwater Usage columns that are larger than the value in the System Limit column.

STARTUP_EXEC
STARTUP_EXEC names a REXX exec that does application environment initialization for z/OS UNIX. This statement is optional; if it is specified, the BPXOINIT process will not run **/etc/init**. The startup exec is typically used by an installation that does not have an HFS, but is using a TFS for a file system. It can be used to populate the TFS with any directories and files that are needed. It is specified as:

```
STARTUP_EXEC('Dsname(Memname)',SysoutClass)
```

where:

- Dsname is a 1-to-44-character valid data set name.
- Memname is a 1-to-8-character valid REXX exec member.

- SysoutClass is 1 character and is alphanumeric and specifies the sysout class that the REXX exec will run under. Specifying SysoutClass is optional.

If you want to change the value of STARTUP_EXEC, you will have to edit the BPXPRMxx member and then reIPL. You cannot use the SET OMVS or SETOMVS command to change the value.

Default: There is no default value for STARTUP_EXEC.

STARTUP_PROC

This statement specifies a 1-to-8-character name of a started JCL procedure that initializes the kernel. The name specified in this statement must exist on the system before IPL or errors will occur.

Using a started procedure other than OMVS is strongly discouraged. If you want to change the value of STARTUP_PROC, you must edit the BPXPRMxx member and then re-IPL. You cannot use the SET OMVS or SETOMVS command to change the value.

If you decide to use a started procedure other than OMVS:

- The replacement started procedure must also be a single job step procedure that invokes the BPXINIT program (EXEC PGM=BPXINIT). If it invokes any other program, the OMVS initialization will fail.
- Change the procedure name in the RACF started procedures table or the definitions in the STARTED Class. See Preparing for RACF in *z/OS UNIX System Services Planning*.

Note: Renaming OMVS to another value might affect the setup of other products such as TCP/IP.

Default: STARTUP_PROC(OMVS).

SUBFILESYSTYPE NAME(transport_name) TYPE(type_name) ENTRYPOINT(entry_name) PARM('parameter') DEFAULT

Specifies an AF_INET or AF_INET6 physical file system that is to run underneath the CINET socket file system. The TYPE() value is usually CINET and matches the TYPE operand on a previous FILESYSTYPE and NETWORK statement. In the case of TCP/IP, the NAME() value is the procname. The system attaches the EZBPFINI load module during initialization, and this file system should be used as the default INET physical file system.

The SUBFILESYSTYPE statement is associated with its corresponding FILESYSTYPE and NETWORK statements by matching the value that is specified in the TYPE operand.

The value that is specified on all of the TYPE operands must match, but can be any 1- to 8-character value. The value that is specified on the NAME parameter on the SUBFILESYSTYPE statement is the name to be used by the physical file system when it is initialized. The first character of the NAME parameter must be non-numeric.

For SecureWay Communications Server, the SUBFILESYSTYPE statement must match the TCPIPJOBNAME of that stack. For more information about defining file systems, see the section on customizing the BPXPRMxx member in *z/OS UNIX System Services Planning*.

New SUBFILESYSTYPE statements can be added dynamically. However, you cannot dynamically change (or delete) a value. For more information, see the section on dynamically adding FILESYSTYPE statements in *z/OS UNIX System Services Planning*.

The parameters are as follows:

DEFAULT
Identifies this file system as the default CINET file system. DEFAULT is an optional parameter. If it is not specified, the file system that is specified in the first SUBFILESYSTYPE statement found in the parmlib member is designated as the default. See the section on setting up for CINET AF_INET sockets in *z/OS UNIX System Services Planning* for more information about the use of the DEFAULT parameter.

ENTRYPOINT(entry_name)
Specifies the name of the load module that contains the entry point into the file system type. ENTRYPOINT is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

NAME(transport_name)
Specifies the name that identifies this file system to the CINET physical file system.

NAME is a required parameter. The name is 1 to 8 characters with the first character non-numeric; the system converts the name to uppercase. The value specified by the NAME parameter on the SUBFILESYSTYPE statement is the name that the physical file system uses to identify itself when it is initialized. For example, for TCP/IP, this is the starting procedure name.

PARM('parameter')
Provides a parameter to be passed to the transport driver. The parameter format and content are specified by the file system that is receiving the data.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. If the characters are not all in uppercase, the parameter must be enclosed in single quotation marks.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

TYPE(type_name)
Specifies the name of the CINET file system type that is identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same name that was used for the TYPE(type_name) parameter on the FILESYSTYPE statement for the CINET physical file system. TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

For more information, see SUBFILESYSTYPE in *z/OS UNIX System Services Planning*.

SUPERUSER(user_name)
Superuser name, which must conform to the restrictions for the z/OS user ID. The user name must also be defined to RACF (or another security product) and must have a z/OS UNIX user ID (UID) of 0. For example, in RACF, specify OMVS(UID(0)) on the ADDUSER command.

When a daemon issues a setuid() to set a UID to 0 and the user ID is not known, setuid() uses the user ID from the SUPERUSER statement.

Never permit the user ID BPXROOT to the BPX.DAEMON profile (described in the section on setting up the BPX.* FACILITY class profiles in *z/OS UNIX System Services Planning*). This warning applies even if you use a name other than BPXROOT.

Value Range: *user_name* is a 1 to 8 character value.

Default: BPXROOT

Use the SETOMVS or SET OMVS command to dynamically change the value of SUPERUSER. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

SWA(ABOVE|BELOW)
Specifies whether SWA control blocks should be allocated above or below the 16-megabyte line. The SWA parameter only affects the OMVS address space and its SWA blocks. It has no impact on the colony address spaces.

ABOVE
All SWA control blocks are to be allocated above the 16-megabyte line.

BELOW
All SWA control blocks are to be allocated below the 16-megabyte line.

Default: BELOW

SYSCALL_COUNTS(YES | NO)
Specifies that syscall counts are to be accumulated in internal kernel data areas so that the RMF data gatherer can record the information.

If you specify YES, the path length for the most frequently used z/OS UNIX system calls is increased by more than 150 instructions. This setting will also cause the reporting of CPU time for z/OS UNIX to be more accurate. This is reflected in the output from the BPX1TIM, BPX1GPS, BPX1GTH, and BPX1RMG services and from BPXESMF.

If you specify NO (or default to NO) the counts for syscall and CPU time are not accumulated.

If you switch between YES and NO for actively running processes, the accumulated data is inaccurate.

The SMF30OST field of the SMF type 30 record can be seen as a negative number when the CPU time used to compute the time value is less than the previous CPU time and in some cases can even appear as zero (for example, when SYSCALL_COUNTS is set to NO after having been YES for an earlier part of the run.) In addition, if SYSCALL_COUNTS=YES, the SMF30OST value is calculated from OCVTSYSTIME (accumulated syscall time for all syscalls issued in the system) rather than from OASBSYSCALLCOUNT (syscalls for the current process).

Default: NO

Use the SETOMVS or SET OMVS command to dynamically change the value of SYSCALL_COUNT. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

TAG (NOTEXT|TEXT,*ccsid*)
Specifies whether implicit file tags are assigned to untagged files in the mounted file system. File tagging controls whether a file's data can be

converted during file reading and writing. “Implicit” in this case means that the tag is not permanently stored with the file. Instead, the tag is associated with the file during reading and writing, or when stat() type functions are issued. Either TEXT, or NOTEXT, and *ccsid* must be specified when TAG is specified.

NOTEXT specifies that none of the files in the file system are automatically converted during file reading and writing.

TEXT specifies that each untagged file is implicitly marked as containing pure text data that can be converted.

ccsid names the coded character set identifier to be implicitly set for the untagged file. *ccsid* is specified as a decimal value from 0 to 65536. However, when TEXT is specified, the values of 0 and 65536 are illegal because those values imply no conversion. Other than this, the value is not checked as being valid and the corresponding code page is not checked as being installed.

For example:

- TAG(TEXT,819) identifies text files that contain ASCII (ISO-8859-1) data.
- TAG(TEXT,1047) identifies text files that contain EBCDIC ((ISO-1047) data.
- TAG(NOTEXT,65536) tags files as containing binary or unknown data.
- TAG(NOTEXT,0) is the equivalent of not specifying the TAG parameter.
- TAG(NOTEXT,273) tags file with the German code set (ISO-273) but is not eligible for automatic conversion.

Default: NOTEXT

SYSPLEX(YES|NO)
For z/OS UNIX, the SYSPLEX statement specifies whether a system is to join the SYSBPX XCF group to share resources across the sysplex. If SYSPLEX(YES) is specified, the system participates in shared file system. If SYSPLEX(NO) is specified, the system does not participate in shared file system. If the SYSPLEX statement is not provided, the default is SYSPLEX(NO). Also, to participate in shared file system, the systems must be at the R9 level or later.

For more information about shared file systems, see Sharing file systems in a sysplex in *z/OS UNIX System Services Planning*. It contains information about parameters specific to shared file system: SYSPLEX, VERSION, AUTOMOVE, NOAUTOMOVE, and SYSNAME.

Note: You cannot adjust the SYSPLEX field dynamically. There is no SETOMVS, SET OMVS, or SETOMVS RESET=(xx) capability. To change the value of SYSPLEX, you must re-IPL the system.

Default: NO

TYPE(type_name)
Specifies the name of a file system type that is identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

TTYGROUP(group_name)
Specifies the z/OS group name given to slave pseudoterminals (PTYs) and

OCS remote terminals (RTYs). This group name must defined to the security product and have a unique group ID (GID). No users should be connected to this group.

The group_name is used by certain setgid() programs, such as talk and write, when writing to another user's PTY or RTY.

Value Range: *group_name* is a 1 to 8 character value.

Default: TTY

You can change the value of TTYGROUP dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

USERIDALIASTABLE('/etc/tablename')
Specifies the path name of a z/OS UNIX file. This file is intended to contain a list of z/OS user IDs and group names with their corresponding alias names. The alias names can contain any characters in the portable file name character set.

You can change USERIDALIASTABLE dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

Once a user is logged into the system, changing the user ID or group name alias table does not change the alias name immediately. If a change needs to be activated sooner, you can use the SETOMVS or SET OMVS command to change the table more quickly.

For planning information, see USERIDALIASTABLE in *z/OS UNIX System Services Planning*.

VERSION('*nnnn*')
The VERSION statement applies only to systems that are exploiting shared file systems. VERSION allows multiple releases and service levels of the binaries to coexist and participate in shared file system. A directory with the value *nnnn* specified on VERSION is dynamically created at system initialization under the sysplex root that is used as a mount point for the version file system. This directory, however, is only dynamically created if the sysplex root HFS is mounted read/write.

Note: *nnnn* is a case-sensitive character string no greater than 8 characters in length. It indicates a specific instance of the version file system. The most appropriate values for *nnnn* are the name of the target zone, &SYSR1, or another qualifier meaningful to the system programmer. For example, if the system is at V2R9, you can specify REL9 for VERSION.
When SYSPLEX(YES) is specified, you must also specify the VERSION parameter.

The VERSION value is substituted in the content of symbolic links that contain $VERSION. For scenarios that describe the use of the version file system, see Sharing file systems in a sysplex in *z/OS UNIX System Services Planning*.

When testing or changing to a new maintenance level, dynamically change the VERSION value by using the SETOMVS command.

```
SETOMVS VERSION='string'
```

You can also change the settings of this parameter by using SET OMVS=(xx) and SETOMVS RESET=(xx) parmlib specifications.

Note: Do not change VERSION dynamically if you have any users that are logged on or running applications because replacing the system files for these users might be disruptive.

Chapter 14. CEAPRMxx (common event adapter parameters)

Use the CEAPRMxx parmlib member to customize common event adapter (CEA) values. CEA provides the ability to deliver z/OS events to C-language clients, such as the z/OS CIM server. CEA also provides instrumentation services used by the IBMzOS_Incident CIM provider. The CEA address space is started automatically during z/OS initialization.

Syntax rules for CEAPRMxx

Follow the rules in "General syntax rules for the creation of members" on page 9. The following rules also apply to the creation of CEAPRMxx parmlib members:

- If a statement that has a default is omitted, the default is used.
- Enter data only in columns 1 through 71. The system ignores data in columns 72 through 80.
- You can enter one or more statements on a line, or use several lines for one statement.
- Use blanks as delimiters. The system interprets multiple blanks as a single blank. You can use blanks between parameters and values. For example, both of the following parameter specifications are equally valid:

  ```
  SNAPSHOT  (Y)
  SNAPSHOT(Y)
  ```

- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- Enter values in uppercase, lowercase, or mixed case. The system converts input to uppercase, except for values enclosed in single quotation marks, which are processed without altering the case.
- Enclose values in single quotation marks, using the following rules
  - Two single quotation marks next to each other on the same line are processed as a single quotation mark. For example, the system interprets `John''s file` as `John's file`.
  - If you enter a single quotation mark in column 71, and another in column 1 of the next line, the system does not treat these two quotation marks as one single quotation mark, as above. Instead, the system interprets this input as quotation marks from two separate strings or as an error.
- When you specify multiple CEAPRMxx parmlib members using concatenation, the system will apply the values in the order in which they are specified. That means that when there are duplicate parameters specified, the last value specified is the one that the system applies.
- When CEA is restarted, it will attempt to determine and use parmlib options that were in effect when it terminated. If it fails, the CEA IPL parmlib concatenation will be used.

Note: When you specify CEA=(xx) in IEASYSxx, do not specify a suffix of "NO".

Syntax format for CEAPRMxx

```
SNAPSHOT ({Y | N})
HLQLONG(CEA)
HLQ(CEA)
BRANCH(branchoffice),COUNTRYCODE(countrycode)
DUMPCAPTURETIME(
  SLIP(OPERLOG(hh:mm:ss) LOGREC(hh:mm:ss) LOGRECSUMMARY(hh:mm:ss))
  DUMP(OPERLOG(hh:mm:ss) LOGREC(hh:mm:ss) LOGRECSUMMARY(hh:mm:ss))
  ABEND(OPERLOG(hh:mm:ss) LOGREC(hh:mm:ss) LOGRECSUMMARY(hh:mm:ss)
)
STORAGE({STORCLAS(stgclass)|VOLSER(volume[,volume[,volume[...]]])})
TSOASMGR(RECONSESSIONS(n) RECONTIME(hh:mm:ss))
```

IBM-supplied defaults for CEAPRMxx

The default CEAPRM00 parmlib member provided in SYS1.PARMLIB contains:

```
SNAPSHOT(N)
```

This value will have to be set to Y when the CIM provider for IBMzOS_Incident is set up.

Statements/parameters for CEAPRMxx

SNAPSHOT({Y | N})

Specifies whether or not you want CEA to preserve snapshots of log information about incidents.

- If Y, every time an SVC Dump is taken, CEA will capture the specified amount of operlog, logrec detail, and logrec summary information for review by the IBM Support Center, if necessary. N is the default. Y is the recommended setting when using the CIM IBMzOS_Incident support. The default amounts are 30 minutes of Operlog, one hour of Logrec detail and a 4-hour Logrec summary report. (The System Logger needs to be set up before OPERLOG or LOGREC snapshots can be taken.)
- If N, the log data will not be captured and will not be available for review by the IBM Support Center.

HLQLONG(*CEA*)

Specifies the high-level qualifier for all log snapshot data sets that are created when SNAPSHOT(Y) is specified. These data sets are of snapshot log streams, prepared data sets used for service, and other associated diagnostic information.

You can adjust this to a different 1- to 8-character qualifier of your own choice. The first character cannot be a number. This value will default to CEA.

If both the HLQLONG and HLQ statements are specified, processing will be done according to the following rules:

- All new diagnostic information will be constructed with the HLQLONG high-level qualifier.
- When searching for specific diagnostic information, the search will first be performed with the HLQLONG value as the high-level qualifier. If no search results are found, a second search will be performed using the HLQ value as the high-level qualifier. If both searches yield no results, the information does not exist.

Note: You are not required to migrate to the longer high-level qualifier. To continue to use your existing configuration, do not specify the HLQLONG statement.

HLQ(*CEA*)

Specifies the high-level qualifier for all log snapshot data sets that are created when SNAPSHOT(Y) is specified. These data sets are of snapshot log streams, prepared data sets used for service, and other associated diagnostic information.

You can adjust this to a different 1- to 4-character qualifier of your own choice. The first character cannot be a number. This value will default to CEA.

If both the HLQLONG and HLQ statements are specified, processing will be done according to the following rules:

- All new diagnostic information will be constructed with the HLQLONG high-level qualifier.
- When searching for specific diagnostic information, the search will first be performed with the HLQLONG value as the high-level qualifier. If no search results are found, a second search will be performed using the HLQ value as the high-level qualifier. If both searches yield no results, the information does not exist.

BRANCH(*branchoffice*),COUNTRYCODE(*countrycode*)

Specifies the branch and country code for use in sending data to the IBM Support Center.

- BRANCH (*branchoffice*) specifies the branch office, typically a three character value.
- COUNTRYCODE (*countrycode*) specifies the country code, typically a three character value.

There are no defaults for these values.

DUMPCAPTURETIME

Specifies the amount of diagnostic data to be captured for different types of dumps. This statement consists of 1 to 3 other statements (SLIP, DUMP or ABEND).

SLIP

Specifies the amount of diagnostic data to be captured with SLIP dumps, measured in time (hours, minutes). The default amounts are 30 minutes of Operlog, one hour of Logrec detail and a 4-hour Logrec summary report.

This statement is an operand of DUMPCAPTURETIME.

DUMP

Specifies the amount of diagnostic data to be captured with DUMP command dumps, measured in time (hours, minutes). The default amounts are 30 minutes of Operlog, one hour of Logrec detail and a 4-hour Logrec summary report.

This statement is an operand of DUMPCAPTURETIME.

ABEND

Specifies the amount of diagnostic data to be captured with system ABEND dumps, measured in time (hours, minutes). The default amounts are 30 minutes of Operlog, one hour of Logrec detail and a 4-hour Logrec summary report.

This statement is an operand of DUMPCAPTURETIME.

The following statements are operands of the SLIP, DUMP and ABEND statements of DUMPCAPTURETIME.

OPERLOG
Specifies the amount of OPERLOG data to be captured with the dump type. The maximum value of OPERLOG is 24 hours.

This statement is an operand of SLIP, DUMP and ABEND.

LOGREC
Specifies the amount of LOGREC data to be captured with the dump type. The maximum value of LOGREC is 24 hours.

This statement is an operand of SLIP, DUMP and ABEND.

LOGRECSUMMARY
Specifies the amount of LOGRECSUMMARY data to be captured with the dump type. The maximum value of LOGRECSUMMARY is 4 hours.

This statement is an operand of SLIP, DUMP and ABEND.

The amount of data captured depends on how the LOGREC information is configured on your system. If the LOGREC information is written to a logstream, the value is rounded to the nearest 1-hour interval. If the LOGREC information is written to a logger data set, the value closely approximates the capture time for the request and is not rounded.

STORAGE(STORCLAS(*stgclass*))
Specifies the SMS class you request the system to use for allocation of incident data sets. *stgclass* is the one- to eight-character name of the SMS storage class that you request to add.

STORAGE(VOLSER(*volume*[,*volume*[,*volume*[...]]]))
Specifies the non-SMS-managed direct access volumes to use for automatic allocation of various incident related data sets. Allocation assigns space from the first volume in the list until that volume is full, and then uses the next volume.

volume is the 1-6 character volume serial identifier of the direct access volume that you want to add to the system's list of resources for automatic allocation. You can specify one to eight direct access volume serial identifiers. Separate the volume serial identifiers with commas.

Note: The storage specified is used to indicate where various data sets are to be allocated during processing. Storage is used for logrec snapshot reports, prepared data sets for the dumps, the logrec reports, the operlogs, and table information.

TSOASMGR
Specifies the ability for TSO/E address spaces to be considered reconnectable for a user; that is, the user is able to reconnect to a TSO/E user session from which the user has requested to log off but that the system has not yet ended (dormant). The next time a request is made to create an address space for that user (through the CEA TSO address space manager services), the address space will be reconnected. The user security credentials and log on information for a reconnectable address space must match those of the original user in order for the system to perform the reconnect.

The statement consists of 1 to 2 statements (RECONSESSIONS and RECONTIME.)

Note: There is an inherent imprecision between the reconnect time interval specified by RECONTIME or the number of sessions requested specified by RECONSESSIONS and the behavior of a running system. The values are enforced at logical times on a system. Logical times occur when creating a session, ending a session, or reaching a system monitor interval of ten (10) minutes.

For example, the TSO/E sessions might have expired before the monitoring interval occurs but will not be removed until the monitoring interval has elapsed. It is possible for sessions that have expired their reconnect time interval to still appear on the system. They are in an expired state, and will not be used for reconnection. These sessions will be removed from the system. However, there may be a delay in this removal, and this delay can be as much as ten (10) minutes after the RECONTIME interval has expired. This delay is due to processing the cleanup of the sessions.

RECONSESSIONS (*n*)
Specifies the number of sessions that the system considers dormant (that is, the user has requested to log off but the session is not ended) and available for that user to reconnect. *n* can be a value in the range 0 - 3. A value of 0 disables this function and is the default.

A non-zero value for *n* allows the specified number of sessions to be considered ended and available to be reconnected for that user. The maximum value is three (3). This value can be changed by the following system command:

```
F CEA,CEA=xx
```

Note that it takes some time for the new values to be integrated into the system. The in-flight sessions will remain until the next time CEA enforces the values. (See **Note** above.)

This statement is an operand of TSOASMGR.

RECONTIME (*hh:mm:ss*)
RECONTIME Specifies the amount of time that a session will remain dorman (the user has requested to log off but the session is not ended) in the format *hh:mm:ss*, where *hh* is hours, *mm* is minutes, and *ss* is seconds. A value of 00:00:00 disables this function and is the default.

A value of 00:00:01 or greater specifies the amount of time an address space remains a candidate for reconnection before it is ended. The maximum value is 23:59:59. This value can be changed by the following system command:

```
F CEA,CEA=xx
```

Note that it takes some time for the new values to be integrated into the system. The in-flight sessions will remain until the next time CEA enforces the values. (See **Note** above.)

This statement is an operand of TSOASMGR.

Examples of CEAPRMxx parmlib member

Example 1:

```
SNAPSHOT(Y)
HLQ(CEA)
BRANCH(379) COUNTRYCODE(000)
DUMPCAPTURETIME(
```

```
    SLIP(OPERLOG(01:00:00) LOGREC(01:00:00) LOGRECSUMMARY(01:00:00))
    DUMP(OPERLOG(01:00:00) LOGREC(01:00:00) LOGRECSUMMARY(01:00:00))
    ABEND(OPERLOG(01:00:00) LOGREC(01:00:00) LOGRECSUMMARY(01:00:00))
)
STORAGE(STORCLAS(STCLAS01))
```

Example 2:

```
SNAPSHOT(Y)
HLQ(CEA)
BRANCH(180) COUNTRYCODE(000)
DUMPCAPTURETIME(
    SLIP(OPERLOG(01:00:00) LOGREC(01:00:00) LOGRECSUMMARY(01:00:00))
    DUMP(OPERLOG(00:10:00) LOGREC(00:10:00) LOGRECSUMMARY(00:10:00))
    ABEND(OPERLOG(02:00:00) LOGREC(02:00:00) LOGRECSUMMARY(02:00:00))
)
STORAGE(VOLSER(VOL101,VOL102,VOL103))
```

Chapter 15. CEEPRMxx (runtime option parameters)

Use the CEEPRMxx parmlib member to provide default Language Environment run-time options for a system. The member is identified during IPL by a CEE=(xx,yy,...) statement, either in the IEASYSxx data set or in the IPL parameter.

You can change the member after IPL with the use of the SET CEE=xx command. You can use the SETCEE command to change individual options. You can display the current options settings with the D CEE command.

The options set in the CEEPRMxx parmlib member override the IBM-supplied default runtime options. When multiple members are concatenated, the individual Language Environment runtime options are merged with the last parmlib member taking precedence. During Language Environment initialization, runtime options are merged from all sources in a specific order of precedence. For more information about the order of precedence, see *z/OS Language Environment Programming Guide* and *z/OS Language Environment Programming Guide for 64-bit Virtual Addressing Mode*.

The Run-Time Options report at the conclusion of a program identifies options merged from the parmlib, those set on the console by SETCEE, and all other sources.

For more information about the runtime options that are valid in the CEEPRMxx parmlib member, see *z/OS Language Environment Customization*.

Parameter in IEASYSxx (or supplied by the operator)

```
CEE=(xx,yy,...,L)
```

The two character identifier (xx) is appended to CEE to identify the CEEPRMxx member of SYS1.PARMLIB. If you specify the L option in the syntax of the CEEPRMxx member, or in reply to the 'SPECIFY SYSTEM PARAMETERS' message, the system writes all statements read from the CEEPRMxx member to the operator's console.

Syntax rules for CEEPRMxx

The following rules apply to the creation of CEEPRMxx:

- More than one option may be specified on a line and an option may be continued on multiple lines.
- Use blanks or commas as delimiters. Multiple blanks are interpreted as a single blank. Blanks are allowed between parameters and values. Commas are allowed between options. Commas are required between suboptions and before an OVR or NONOVR attribute.

  Restriction: Blanks are not allowed within the required =((delimiter or ((delimiter for runtime options that specify an OVR or NONOVR attribute.

- Enter values in uppercase, lowercase, or mixed case. The system converts the input to uppercase, except for values enclosed in single quotation marks, which are processed without changing the case.
- The options for each group are saved independently so that CEEDOPT can specify NATLANG(ENU) while CEECOPT can specify NATLANG(JPN).
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". Nested comments are not allowed.

Syntax format of CEEPRMxx

```
CEECOPT(rtopt[=]((subopts),{OVR|NONOVR})[,rtopt[=]((subopts),{OVR|NONOVR})]...)
CEEDOPT(rtopt[=]((subopts),{OVR|NONOVR})[,rtopt[=]((subopts),{OVR|NONOVR})]...)
CELQDOPT(rtopt[=]((subopts),{OVR|NONOVR})[,rtopt[=]((subopts),{OVR|NONOVR})]...)
CEEROPT(ALL|COMPAT)
CELQROPT(ALL|NONE)
```

To specify runtime options without the OVR or NONOVR attribute, use the following syntax:

```
CEECOPT(rtopt(subopts)[,rtopt(subopts)]...)
CEEDOPT(rtopt(subopts)[,rtopt(subopts)]...)
CELQDOPT(rtopt(subopts)[,rtopt(subopts)]...))
```

You can override runtime options that are specified with this format.

IBM-supplied default for CEEPRMxx

None.

Statements/parameters for CEEPRMxx

To specify system level default runtime options, use the following group options:

CEECOPT
Specifies runtime options for CICS environments.

CEEDOPT
Specifies runtime options for non-CICS environments (except AMODE 64 environments).

CELQDOPT
Specifies runtime options for AMODE 64 environments.

To specify individual runtime options within one of the valid group options:

rtopt Specifies a runtime option.

subopts
Specifies the valid suboptions for a particular runtime option.

OVR Indicates that the option can be overridden.

NONOVR
Indicates that the option cannot be overridden.

Note: There are runtime options that use the *rtopt*|NO*rtopt* format without suboptions to turn the option on or off, for example, DEBUG|NODEBUG.

To use the override attributes with this type of option, it must be specified in the corresponding *rtopt*=((ON | OFF),OVR | NONOVR) format. For example, to turn off the DEBUG option, specify DEBUG=((OFF),OVR).

Once a runtime option is specified as non-overrideable with the NONOVR attribute in a CEEPRMxx parmlib member or a SETCEE command, it cannot later be overridden within the same parmlib member or with a SETCEE command.

To remove a non-overrideable setting from a runtime option, use the SETCEE CLEAR operator command or another parmlib member.

For more information about individual runtime options, see *z/OS Language Environment Customization*.

To control Language Environment processing, use the following keywords:

CEEROPT
Indicates whether region-specific runtime options should be used in a non-CICS or non-LRR environment. The value can be either ALL or COMPAT.

ALL
Attempts a load and use of CEEROPT in all AMODE 31 environments.

COMPAT
Attempts a load and use of CEEROPT only in CICS or LRR environments.

CELQROPT
Indicates whether region-specific runtime options should be used in AMODE 64. The value can be either ALL or NONE.

ALL
Attempts a load and use of CELQROPT in all AMODE 64 environments.

NONE
Does not attempt a load or use of CELQROPT in AMODE 64 environments.

Example of CEEPRMxx parmlib member

Figure 9 on page 190 shows the IBM-supplied version of the CEEPRM00 member, with all valid options within comments. You must remove the comment characters from the options that you want to use.

```
/*********************************************************************/
/* CEEPRM00 - Sample Language Environment parmlib member for         */
/*            runtime options.                                       */
/*                                                                   */
/* LICENSED MATERIALS - PROPERTY OF IBM                              */
/*                                                                   */
/* 5650-ZOS                                                          */
/*                                                                   */
/* COPYRIGHT IBM CORP. 2005, 2012                                    */
/*                                                                   */
/* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                      */
/* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP                   */
/* SCHEDULE CONTRACT WITH IBM CORP.                                  */
/*                                                                   */
/* STATUS = HLE7790                                                  */
/*                                                                   */
/*                                                                   */
/* This sample parmlib member contains the IBM-supplied default      */
/* runtime options that are valid at the system level. The defaults  */
/* can be overridden using the options groups CEEDOPT, CEECOPT, and  */
/* CELQDOPT.                                                         */
/*                                                                   */
/* This sample also contains the default values for the CEEROPT and  */
/* CELQROPT keywords.                                                */
/*                                                                   */
/*                                                                   */
/* Syntax for options:                                               */
/*                                                                   */
/*    group_name( option_1, option_2,                                */
/*                option_3, option_4 )                               */
/*                                                                   */
/*    Where:                                                         */
/*       group_name is CEEDOPT, CEECOPT or CELQDOPT.                 */
/*       option_x is any option valid at the system level.           */
/*                                                                   */
/*                                                                   */
/* Syntax for keywords:                                              */
/*                                                                   */
/*    CEEROPT( value )  -   Where value is ALL or COMPAT             */
/*    CELQROPT( value ) -   Where value is ALL or NONE               */
/*                                                                   */
/*                                                                   */
/* All valid options and keywords are listed but commented out.      */
/* To include an option you must edit this file (or a copy) and      */
/* remove the comment delimiters around the options to be used.      */
/* It is not necessary to uncomment all options.                     */
/*                                                                   */
/*                                                                   */
/* Notes:                                                            */
/*                                                                   */
/* * Comments and blank lines are allowed for readability.           */
/*                                                                   */
/* * Individual options must be separated by a comma or a blank      */
/*                                                                   */
/* * There can be more than one option on a line.                    */
/*                                                                   */
/* * Mixed case is allowed.                                          */
/*                                                                   */
/* * Individual options can be specified as overrideable (OVR) or    */
/*   nonoverrideable (NONOVR).                                       */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
```

Figure 9. IBM-supplied version of the CEEPRM00 member (Part 1 of 4)

```
/*******************************************************************/
/* 31 bit non-CICS option group                                    */
/*******************************************************************/
/*CEEDOPT(                                                         */
/*        ABPERC=((NONE),OVR),                                     */
/*        ABTERMENC=((ABEND),OVR),                                 */
/*        AIXBLD=((OFF),OVR),                                      */
/*        ALL31=((ON),OVR),                                        */
/*        ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR),                    */
/*        BELOWHEAP=((8K,4K,FREE),OVR),                            */
/*        CBLOPTS=((ON),OVR),                                      */
/*        CBLPSHPOP=((ON),OVR),                                    */
/*        CBLQDA=((OFF),OVR),                                      */
/*        CEEDUMP=((60,SYSOUT=*,FREE=END,SPIN=UNALLOC),OVR),       */
/*        CHECK=((ON),OVR),                                        */
/*        COUNTRY=((US),OVR),                                      */
/*        DEBUG=((OFF),OVR),                                       */
/*        DEPTHCONDLMT=((10),OVR),                                 */
/*        DYNDUMP=((*USERID,NODYNAMIC,TDUMP),OVR),                 */
/*        ENVAR=(('',),OVR),                                       */
/*        ERRCOUNT=((0),OVR),                                      */
/*        ERRUNIT=((6),OVR),                                       */
/*        FILEHIST=((ON),OVR),                                     */
/*        FILETAG=((NOAUTOCVT,NOAUTOTAG),OVR),                     */
/*        HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR),                */
/*        HEAPCHK=((OFF,1,0,0,0,1024,0,1024,0),OVR),               */
/*        HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,   */
/*        10,0,10,0,10,0,10,0,10,0,10,0,10),OVR),                  */
/*        INFOMSGFILTER=((OFF,,,,),OVR),                           */
/*        INQPCOPN=((ON),OVR),                                     */
/*        INTERRUPT=((OFF),OVR),                                   */
/*        LIBSTACK=((4K,4K,FREE),OVR),                             */
/*        MSGFILE=((SYSOUT,FBA,121,0,NOENQ),OVR),                  */
/*        MSGQ=((15),OVR),                                         */
/*        NATLANG=((ENU),OVR),                                     */
/*        NOAUTOTASK=(OVR),                                        */
/*        NOTEST=((ALL,*,PROMPT,INSPPREF),OVR),                    */
/*        NOUSRHDLR=((),OVR),                                      */
/*        OCSTATUS=((ON),OVR),                                     */
/*        PC=((OFF),OVR),                                          */
/*        PLITASKCOUNT=((20),OVR),                                 */
/*        POSIX=((OFF),OVR),                                       */
/*        PROFILE=((OFF,''),OVR),                                  */
/*        PRTUNIT=((6),OVR),                                       */
/*        PUNUNIT=((7),OVR),                                       */
/*        RDRUNIT=((5),OVR),                                       */
/*        RECPAD=((OFF),OVR),                                      */
/*        RPTOPTS=((OFF),OVR),                                     */
/*        RPTSTG=((OFF),OVR),                                      */
/*        RTEREUS=((OFF),OVR),                                     */
/*        SIMVRD=((OFF),OVR),                                      */
/*        STACK=((128K,128K,ANYWHERE,KEEP,512K,128K),OVR),         */
/*        STORAGE=((NONE,NONE,NONE,0K),OVR),                       */
/*        TERMTHDACT=((TRACE,,96),OVR),                            */
/*        THREADHEAP=((4K,4K,ANYWHERE,KEEP),OVR),                  */
/*        THREADSTACK=((OFF,4K,4K,ANYWHERE,KEEP,128K,128K),OVR),   */
/*        TRACE=((OFF,4K,DUMP,LE=0),OVR),                          */
/*        TRAP=((ON,SPIE),OVR),                                    */
/*        UPSI=((00000000),OVR),                                   */
/*        VCTRSAVE=((OFF),OVR),                                    */
/*        XUFLOW=((AUTO),OVR)                                      */
/*       )                                                         */
```

Figure 10. IBM-supplied version of the CEEPRM00 member (Part 2 of 4)

```
/*********************************************************************/
/* 31 bit CICS option group                                          */
/* The following options are ignored in CICS:                        */
/*  - ABPERC                - OCSTATUS                               */
/*  - AIXBLD                - PC                                     */
/*  - AUTOTASK              - PLITASKCOUNT                           */
/*  - CBLOPTS               - POSIX                                  */
/*  - CBLQDA                - PRTUNIT                                */
/*  - DYNDUMP               - PUNUNIT                                */
/*  - ERRUNIT               - RDRUNIT                                */
/*  - FILEHIST              - RECPAD                                 */
/*  - FILETAG               - RTEREUS                                */
/*  - INQPCOPN              - SIMVRD                                 */
/*  - INTERRUPT             - THREADHEAP                             */
/*  - MSGFILE               - THREADSTACK                            */
/*  - MSGQ                  - VCTRSAVE                               */
/*                                                                   */
/*********************************************************************/
/*CEECOPT(                                                           */
/*        ABTERMENC=((ABEND),OVR),                                   */
/*        ALL31=((ON),OVR),                                          */
/*        ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR),                     */
/*        BELOWHEAP=((4K,4080,FREE),OVR),                            */
/*        CBLPSHPOP=((ON),OVR),                                      */
/*        CEEDUMP=((60,SYSOUT=*,FREE=END,SPIN=UNALLOC),OVR),         */
/*        CHECK=((ON),OVR),                                          */
/*        COUNTRY=((US),OVR),                                        */
/*        DEBUG=((OFF),OVR),                                         */
/*        DEPTHCONDLMT=((10),OVR),                                   */
/*        ENVAR=(('',OVR),                                           */
/*        ERRCOUNT=((0),OVR),                                        */
/*        HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR),                */
/*        HEAPCHK=((OFF,1,0,0,0,1024,0,1024,0),OVR),                 */
/*        HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,     */
/*        10,0,10,0,10,0,10,0,10,0,10,0,10),OVR),                    */
/*        INFOMSGFILTER=((OFF,,,,),OVR),                             */
/*        LIBSTACK=((32,4080,FREE),OVR),                             */
/*        NATLANG=((ENU),OVR),                                       */
/*        NOTEST=((ALL,*,PROMPT,INSPPREF),OVR),                      */
/*        NOUSRHDLR=((),OVR),                                        */
/*        PROFILE=((OFF,''),OVR),                                    */
/*        RPTOPTS=((OFF),OVR),                                       */
/*        RPTSTG=((OFF),OVR),                                        */
/*        STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR),               */
/*        STORAGE=((NONE,NONE,NONE,0K),OVR),                         */
/*        TERMTHDACT=((TRACE,CESE,96),OVR),                          */
/*        TRACE=((OFF,4K,DUMP,LE=0),OVR),                            */
/*        TRAP=((ON,SPIE),OVR),                                      */
/*        UPSI=((00000000),OVR),                                     */
/*        XUFLOW=((AUTO),OVR)                                        */
/*        )                                                          */
```

Figure 11. IBM-supplied version of the CEEPRM00 member (Part 3 of 4)

```
/*********************************************************************/
/* 64 bit options group                                              */
/*********************************************************************/
/*CELQDOPT(                                                          */
/*         CEEDUMP=((60,SYSOUT=*,FREE=END,SPIN=UNALLOC),OVR),        */
/*         DYNDUMP=((*USERID,NODYNAMIC,TDUMP),OVR),                  */
/*         ENVAR=(('')),OVR),                                        */
/*         FILETAG=((NOAUTOCVT,NOAUTOTAG),OVR),                      */
/*         HEAPCHK=((OFF,1,0,0,0,1024,0,1024,0),OVR),                */
/*         HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,         */
/*         2048,10,0,10,0,10,0,10,0,10,0,10,0,10),OVR),              */
/*         HEAPPOOLS64=((OFF,8,4000,32,2000,128,700,256,350,         */
/*         1024,100,2048,50,3072,50,4096,50,8192,25,16384,10,        */
/*         32768,5,65536,5),OVR),                                    */
/*         HEAP64=((1M,1M,KEEP,32K,32K,KEEP,4K,4K,FREE),OVR),        */
/*         INFOMSGFILTER=((OFF,,,,),OVR),                            */
/*         IOHEAP64=((1M,1M,FREE,12K,8K,FREE,4K,4K,FREE),OVR),       */
/*         LIBHEAP64=((1M,1M,FREE,16K,8K,FREE,8K,4K,FREE),OVR),      */
/*         NATLANG=((ENU),OVR),                                      */
/*         NOTEST=((ALL,*,PROMPT,INSPPREF),OVR),                     */
/*         POSIX=((OFF),OVR),                                        */
/*         PROFILE=((OFF,''),OVR),                                   */
/*         RPTOPTS=((OFF),OVR),                                      */
/*         RPTSTG=((OFF),OVR),                                       */
/*         STACK64=((1M,1M,128M),OVR),                               */
/*         STORAGE=((NONE,NONE,NONE,),OVR),                          */
/*         THREADSTACK64=((OFF,1M,1M,128M),OVR),                     */
/*         TERMTHDACT=((TRACE,,96),OVR),                             */
/*         TRACE=((OFF,,DUMP,LE=0),OVR),                             */
/*         TRAP=((ON,SPIE),OVR)                                      */
/*        )                                                          */

/*********************************************************************/
/* Keywords                                                          */
/*********************************************************************/
/*CEEROPT( COMPAT )                                                  */
/*CELQROPT( NONE )                                                   */
```

Figure 12. IBM-supplied version of the CEEPRM00 member (Part 4 of 4)

Chapter 16. CLOCKxx (time of day parameters)

CLOCKxx performs the following functions:

- Prompts the operator to initialize the time of day (TOD) clock during system initialization.
- Specifies the difference between the local time and Coordinated Universal Time (UTC).
- Controls the utilization of the IBM Sysplex Timer (9037), which is an external time reference (ETR). Having all systems in your complex attached and synchronized to a Sysplex Timer ensures accurate sequencing and serialization of events.
- Provides the means of specifying that the Server time Protocol (STP) architecture is to be used in the sysplex. STP defines the method by which multiple servers maintain time synchronization.
- Allows an installation to set an acceptable time deviation for the TOD clock from an external time source. When the specified ACCURACY bounds are exceeded, an error message is issued so the problem can be corrected. See 'Setting the TOD clock accuracy monitor service ' in *z/OS MVS Planning: Operations* for additional details.

The CLOCKxx member for a system that is a member of a multisystem sysplex must contain a specification of ETRMODE YES, STPMODE YES, or both. The system then uses the Sysplex Timer or STP to synchronize itself with the other members of the sysplex. The system uses a synchronized time stamp to provide appropriate sequencing and serialization of events within the sysplex.

Note: If all MVS images in the sysplex will run in LPARs or under VM on a single physical processor, you can specify SIMETRID instead of ETRMODE YES or STPMODE YES.

For more information about CLOCKxx and the Sysplex Timer, see *z/OS MVS Setting Up a Sysplex*.

Parameter in IEASYSxx (or supplied by the operator)

```
CLOCK= {aa          }
       {(aa,bb,...L)}
```

The two character identifier (aa, bb, and so forth) is appended to CLOCK to identify the CLOCKxx member of SYS1.PARMLIB. If you specify the L option in the syntax of the CLOCKxx member, or in reply to the 'SPECIFY SYSTEM PARAMETERS' message, the system writes all statements read from the CLOCKxx member to the operator's console.

Syntax rules for CLOCKxx

The following rules apply to the creation of CLOCKxx:

1. Use columns 1 through 71. Do not use columns 72-80 for data; these columns are ignored.

2. Comments may appear in columns 1-80 and must begin with "/*" and end with "*/"
3. At least one blank has to follow the statement types.
4. The following combinations of parameters are not valid and will be rejected:
   - ETRMODE NO and ETRZONE YES
   - STPMODE NO and STPZONE YES
5. In STP mode, TIMEDELTA is the replacement value to be used instead of ETRDELTA, although the system will continue to accept ETRDELTA.

Syntax format of CLOCKxx

```
OPERATOR  {PROMPT  }
          {NOPROMPT}

TIMEZONE d.hh.mm.ss

ACCURACY mmmmm

ETRMODE  {YES}
         {NO }

ETRDELTA  nn

ETRZONE  {YES}
         {NO }

SIMETRID nn

STPMODE  {YES}
         {NO }

TIMEDELTA  nn

STPZONE  {YES}
         {NO }
```

IBM-supplied default for CLOCKxx

The IBM-supplied default parmlib member of SYS1.PARMLIB is CLOCK00, which contains the following:

```
OPERATOR  NOPROMPT
TIMEZONE  W.00.00.00
ETRMODE   YES
ETRZONE   YES
STPMODE   YES
STPZONE   YES
ETRDELTA  10
ACCURACY  0
```

Statements/parameters for CLOCKxx

OPERATOR {PROMPT|NOPROMPT}

Specifies whether the operator is to be prompted to set the TOD clock during system initialization.

PROMPT

Specifies that the system is to prompt the operator during TOD initialization.

NOPROMPT
Specifies that the system is not to prompt the operator during TOD initialization unless the clock is not set.

Note:

1. If ETRMODE YES or STPMODE YES is specified, the system ignores the OPERATOR parameter.
2. OPERATOR PROMPT and SIMETRID are mutually exclusive keywords. Specify either OPERATOR PROMPT or SIMETRID, but not both. If both are specified, the system rejects the CLOCKxx member during system initialization, and issues a message to prompt the operator for one of the following:
   - A valid CLOCKxx member or
   - EOB (by pressing the enter button on the console).

   Otherwise, the operator must reIPL the system.
3. Systems running as Sysplex Test Datesource LPARs must use SIMETRID and therefore must specify OPERATOR NOPROMPT.

Default: NOPROMPT

TIMEZONE d.hh.mm.ss
Specifies the difference between the local time and the Coordinated Universal Time (UTC). If ETRMODE YES and ETRZONE YES are specified (and an operational Sysplex Timer is available), the system ignores the TIMEZONE parameter.

d Specifies the direction from UTC.

Value Range: E for east of UTC or W for west of UTC.

Default: W

hh.mm.ss
Specifies the number of hours (*hh*) minutes (*mm*) and seconds (*ss*) that the local time differs from the UTC.

Value Range: The value for hh must be between 00 and 15. The value for *mm* and *ss* must be between 00 and 59. mm.ss values are optional.

In addition, the combined hh.mm.ss value must be within the range (00:00:00 - 15:00:00). This means that a value like 15.59.59 is not valid because it is outside the range, even though the *hh* portion is between 00 and 15 and the *mm* and *ss* portions are between 00 and 59. If the *mm* portion or *ss* portion or both are omitted, a default value of 00 is applied to the omitted portion, and appears in message IEA598I at IPL time.

For example, if CLOCKxx contains TIMEZONE W.15, then at IPL time, message IEA598I would indicate:

```
IEA598I TIME ZONE = W.15.00.00
```

Default: 00.00.00

Note: Some applications have a requirement to run with local time equal to UTC. When Server Time Protocol (STP) is implemented, this requirement can be satisfied by setting an LPAR time offset equal to the local time of day offset and a time zone of zero in the CLOCKxx parmlib member.

ACCURACY mmmmm
Specifies an acceptable time deviation for the TOD clock from an external time

source in milliseconds, with valid values between 0 and 60000 milliseconds (60 seconds). The default value is 0 and indicates that the time accuracy check function is not enabled. If the difference between the TOD clock and the external time source time exceeds the ACCURACY value, message IEA032E is issued.

ETRMODE {YES|NO}
Specifies whether MVS is using a Sysplex Timer. You must code ETRMODE YES if this system is a member of a multisystem sysplex. To set ETRMODE to YES, the sysplex must be attached to an operational Sysplex Timer.

Note: If all MVS images in the sysplex will run in LPARs or under VM on a single physical processor, you can specify SIMETRID instead of ETRMODE YES.

YES
Specifies that MVS is to use the Sysplex Timer, if available. If you specify ETRMODE YES and an operational Sysplex Timer is not available, the operator will be prompted to set the TOD clock during system initialization.

NO Specifies that MVS is not to use the Sysplex Timer.

Note:

1. If PR/SM* is active, the TOD clocks initially are synchronized to the Sysplex Timer.
2. Specifying **NO** overrides both the external time reference (ETR) and the simulated sysplex timer (SIMETR).
3. **Configurations that support ETRMODE NO are XCF-Local mode and Monoplex mode.**

Default: YES

ETRDELTA nn
Indicates the greatest difference, after IPL, between the system's TOD and the Sysplex Timer TOD by which the system will adjust its TOD, when necessary, to match the Sysplex Timer TOD.

If the difference between the system's TOD and the Sysplex Timer's TOD exceeds the ETRDELTA, the result is:

1. If the system is part of a multisystem sysplex, the system is terminated with wait state 0A2.
2. If the system is not part of a multisystem sysplex, processing continues, but the Sysplex Timer is not used for the remainder of that IPL.

If you select a value of 0 seconds for ETRDELTA, no deviation between the processor TOD clock and the 9037 Sysplex Timer clock can be corrected by z/OS, so no TOD adjustment is possible. When a synch check is recognized in the ETRDELTA 0 case, one of the two above actions will result. IBM suggests that the default of 10 seconds be selected.

Value Range: 0 to 99 seconds

Default: 10 seconds

ETRZONE {YES|NO}
Specifies whether the system is to get the time zone constant from the Sysplex Timer. The time zone constant specifies the difference between the local time and the Coordinated Universal Time (UTC).

YES
Specifies that the system is to use the Sysplex Timer to set the time zone constant. If you specify ETRZONE YES and an operational Sysplex Timer is not available, the system uses the time zone constant specified on the TIMEZONE parameter.

NO Specifies that the system is to use the time zone constant specified on the TIMEZONE parameter.

Default: YES when ETRMODE is set to YES. NO when ETRMODE is set to NO.

SIMETRID nn
Specifies the simulated Sysplex Timer identifier. SIMETRID allows MVS images running on the same central electronics complex (CEC), in native mode in LPARs, or as z/VM® guests to participate in a multi-system sysplex when no real sysplex timer is available. In these environments, the MVS TOD clocks are synchronized by PR/SM or the z/VM host. If a real sysplex timer is available when MVS is not being run as a VM guest, IBM suggests that you use it instead of SIMETRID.

Do not use SIMETRID on MVS images running on different CECs. Instead, use a real sysplex timer and specify ETRMODE YES. When using a real sysplex timer, your installation operational requirements will determine whether you specify YES or NO for ETRZONE. Do not specify SIMETRID if you plan to IPL MVS native (for example, not in an LPAR and not under VM), even if you do not plan to have more than one MVS image in the sysplex. SIMETRID is only available when MVS runs in an LPAR, or under z/VM.

Each image participating in the same sysplex and using simulated ETR for time synchronization must specify the same Sysplex Timer identifier (nn).

Once the logical TOD clock has been set in a partition by an IPL, the LPAR must be reactivated to reset the time again.

Note: OPERATOR PROMPT and SIMETRID are mutually exclusive keywords. Specify either OPERATOR PROMPT or SIMETRID, but not both. If both are specified, the system rejects the CLOCKxx member during system initialization, and issues a message to prompt the operator for one of the following:

- A valid CLOCKxx member or
- EOB (by pressing the enter button on the console).

Otherwise, the operator must reIPL the system.

Value Range: A two-digit hexadecimal number (X'00-1F').

Default: None. This value is optional. If specified, ETRZONE, ETRMODE, STPZONE and STPMODE are ignored and the timezone constant specified by TIMEZONE is used.

STPMODE {YES|NO}
Specifies whether MVS is using STP timing mode. If you specify STPMODE YES and STP is not available, the operator will be prompted to set the TOD clock during system initialization.

YES
Specifies that MVS is to use STP timing mode if STP is active.

NO Specifies that MVS is not to use STP timing mode. Either ETR or local MVS

timing mode will be used depending on the value of the ETRMODE keyword and whether the ETR is usable or not.

Default: YES

Note:

1. If the z/OS system image is running on a server that is in ETR timing mode, and both STPMODE and ETRMODE have been specified as YES, z/OS uses ETRMODE YES.
2. If the z/OS system image is running on a server that is in STP timing mode, and both STPMODE and ETRMODE have been specified as YES, z/OS uses STPMODE YES.
3. If the z/OS system image is running on a server that is in LOCAL timing mode, STPMODE NO should be explicitly coded to avoid the issuing of messages IEA888A and IEA381I during system initialization.

   If either ETRMODE YES or STPMODE YES (default) is specified, z/OS issues IEA888A to prompt the operator to set the TOD clock during system initialization. This occurs regardless of whether OPERATOR PROMPT or NOPROMT has been specified.

   In addition, if ETRMODE YES is specified, the following message is issued:

   ```
   IEA261I NO ETR PORTS ARE USABLE. CPC CONTINUES TO RUN IN LOCAL MODE.
   ```

   If ETRMODE NO and STPMODE YES (default) is specified, the following message is issued:

   ```
   IEA381I THE STP FACILITY IS NOT USABLE. SYSTEM CONTINUES IN LOCAL
   TIMING MODE.
   ```

TIMEDELTA nn

Indicates the greatest difference, after IPL, between the system's TOD and the STP Stratum 1 Server's time value, by which the system will adjust its TOD, when necessary, to match the Stratum 1 Server's TOD.

If the difference between the system's TOD and the Stratum 1 Server's TOD exceeds the TIMEDELTA, the result is:

1. If the system is part of a multisystem sysplex, the system is terminated with wait state 0A2.
2. If the system is not part of a multisystem sysplex, processing continues, but STP is not used for the remainder of that IPL.

If you select a value of 0 seconds for TIMEDELTA, no deviation between the processor TOD clock and the Stratum 1 Server's TOD can be corrected by z/OS, so no TOD adjustment is possible. When a synch check is recognized in the TIMEDELTA 0 case, one of the two above actions will result. IBM suggests that the default of 10 seconds be selected.

Value Range: 0 to 99 seconds

Default: 10 seconds

STPZONE {YES|NO}

Specifies whether the system is to get the time zone constant from STP. The time zone constant specifies the difference between the local time and the Coordinated Universal Time (UTC).

YES

Specifies that the system is to use STP to set the time zone constant. If you specify STPZONE YES and STP is not enabled, the system uses the time zone constant specified on the TIMEZONE parameter.

NO Specifies that the system is to use the time zone constant specified on the TIMEZONE parameter.

Default: YES when STPMODE is set to YES. NO when STPMODE is set to NO.

Chapter 17. CNGRPxx (specify console groups)

Use the CNGRPxx parmlib member to define console groups. You can specify HMCS, MCS, SMCS, and extended MCS consoles as members of these groups.

HMCS, MCS, and SMCS consoles are defined to your system through the CONSOLxx Parmlib member, which is described in Chapter 23, "CONSOLxx (console configuration definition)," on page 231. For information about defining extended MCS consoles to your system, see *z/OS MVS Planning: Operations*.

You can use console groups in the following situations:

- To specify the order in which consoles are to receive synchronous messages. You specify this group on the DEFAULT statement in the CONSOLxx parmlib member.
- To identify the consoles that must be inactive for the system to place the system console into problem determination state. You specify this group in the AUTOACT keyword on the CONSOLE statement for the system console.

Console groups in a SYSPLEX

When a system joins a sysplex, the system inherits any console group definitions that are currently defined in the sysplex; its own console group definitions in the INIT statement on CONSOLxx are ignored. If there are no console groups defined when a system joins the sysplex, the joining system's definitions will be in effect for the entire sysplex. After the system is up, any system in the sysplex can issue the SET CNGRP command to add or change the console group definitions. The change lasts for the duration of the IPL.

Selecting a CNGRPxx member

You can select a CNGRPxx member in the following ways:

- Specify the CNGRP keyword on the INIT statement of the CONSOLxx Parmlib member. For more information, see Chapter 23, "CONSOLxx (console configuration definition)," on page 231.
- Issue the SET CNGRP command either through the COMMNDxx Parmlib member or after initialization. You can have up to 38 CNGRPxx members active at a time. For more information about the SET CNGRP command, see *z/OS MVS System Commands*.

If you define the same console group in separate CNGRPxx members, and those members are activated, the system will use the first specification of the console group.

Syntax rules for CNGRPxx

The following syntax rules apply to CNGRPxx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- A comma must be used to separate multiple keyword values within a list.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

- No more than 226 lines per CNGRPxx member, including comments, will be processed by the SET CNGRP command.

Syntax format of CNGRPxx

```
GROUP     NAME(group name)
          MEMBERS(console name[,console name,...])
```

IBM-supplied default for CNGRPxx

None.

Statement/parameters for CNGRPxx

GROUP

Identifies the beginning of a console group definition. Each group definition consists of one NAME keyword and one MEMBERS keyword. You can specify more than one GROUP statement in one CNGRPxx parmlib member.

NAME(group name)

Specifies the name of the console group. Use this name in the CONSOLxx parmlib member for one of the following reasons:

- On the DEFAULT statement to specify the order in which consoles are to receive synchronous messages.
- On the CONSOLE statement in the AUTOACT keyword to indicate the automatic activate group for the system console.

Value Range: 1 to 8 alphanumeric or special (#, @, or $) characters.

Default: None

MEMBERS(console name [,console name,...])

Specifies the ordered list of console names belonging to the specified group. Depending on whether a console is an HMCS, MCS, SMCS or an extended MCS console, define its name through either:

- The CONSOLE statement of the CONSOLxx parmlib member for HMCS, MCS, and SMCS consoles
- Security Server or TSO/E for extended MCS consoles.

Value Range: *console name* is from 2 to 8 characters. The first character of *console name* must begin with the letters A through Z or with a #, $, or @; the remaining characters can be A through Z, 0 through 9, or #, $, or @.

The following reserved console name has a special meaning when used as part of a console group:

Console name	Meaning
SYSCON	When used in a group specified on the SYNCHDEST keyword on the DEFAULT statement in CONSOLxx, this routes a synchronous message to the system console.

Chapter 18. CNLcccxx (time and date format for translated messages)

Use the CNLcccxx member of parmlib to specify how translated messages are to be displayed at your installation. CNLcccxx, which is called the message configuration member, allows you to specify the time and date format for translated messages, using the MONTH, DAY, DATE, and TIME statements, as follows:

- Use the required MONTH statement to specify the month names that are to appear in your translated messages. MONTH1 specifies the month name for January, MONTH2 for February, and so on.
- Use the required DAY statement to specify the names of the days of the week that are to appear in your translated messages. DAY1 specifies the name for Sunday, DAY2 for Monday, and so on.
- Use the required DEFAULTS statement to define the default date and time formats for a language.
- Use the optional DATE statement to specify the format for the date. Examples of different date formats are:

  ```
  1/14/2000
  14-1-2000
  14 January 2000
  ```

- Use the optional TIME statement to specify the format for the time. Examples of different time formats include:

  ```
  11:46:12 PM
  46 MINUTES PAST 11
  23:46:12
  ```

You need one message configuration member for each language your installation supports. Each message configuration member is named CNLcccxx, where ccc is the appropriate language code (see Table 30 on page 647 for a table of valid language codes) and xx identifies the member.

The message configuration member for a specific language is specified on the CONFIG keyword of the LANGUAGE statement in the MMSLSTxx parmlib member. For more information, see Chapter 70, "MMSLSTxx (MVS message service list)," on page 645.

IBM provides you with message configuration members for English and Japanese in SYS1.PARMLIB (named CNLENU00 and CNLJPN00, respectively).

Restrictions for CNLcccxx

Observe the following restrictions when using CNLcccxx:

- Only one CNLcccxx member can be active at one time.
- The system uses the IBM-supplied CNLENU00 member for message processing. You can add definitions to the CNLENU00 member, but do not delete the IBM-supplied definitions. Doing so can cause misformatted text to appear in system messages.

Parameter in IEASYSxx (or supplied by the operator)

None.

Selecting a CNLcccxx member

The SET MMS command allows you to modify the MMS parameters that are currently in effect for your system, including the currently active CNLcccxx member. By selecting a MMSLSTxx member that specifies a different CNLcccxx member, you cause the system to refresh the current CNLcccxx settings. The new MMS and CNLcccxx settings take effect immediately (that is, without requiring a reIPL of the system). For more information about the SET MMS command, see *z/OS MVS System Commands*.

Syntax rules for CNLcccxx

The following syntax rules apply to the creation of CNLcccxx:

- Use columns 1 through 71 for data. Columns 72 - 80 are ignored.
- Specify at least one delimiter (space or comma) between a statement and a keyword. Delimiters are not required between keywords.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". Comments can span more than one line.
- The beginning and ending double byte character set (DBCS) delimiters are called *shift-out* and *shift-in* characters. In examples, the convention <d1d2> is used to represent DBCS strings enclosed in their shift-out and shift-in characters, where d1 and d2 each represent a DBCS character, < represents X'0E'(shift-in), and > represents X'0F'(shift-out).
- The following statements are **required**:

  ```
  MONTH
  DAY
  DEFAULTS
  ```

- The following statements are **optional**:

  ```
  DATE
  TIME
  ```

  When optional DATE and TIME statements are not specified, the system uses the default date and time from the DEFAULTS statement.

Syntax format of CNLcccxx

```
MONTH      MONTH1(monthname)
           MONTH2(monthname)
           MONTH3(monthname)
           MONTH4(monthname)
           MONTH5(monthname)
           MONTH6(monthname)
           MONTH7(monthname)
           MONTH8(monthname)
           MONTH9(monthname)
           MONTH10(monthname)
           MONTH11(monthname)
           MONTH12(monthname)

DAY        DAY1(dayname)
           DAY2(dayname)
           DAY3(dayname)
           DAY4(dayname)
           DAY5(dayname)
           DAY6(dayname)
           DAY7(dayname)

DATE       ID(DATEnnnnnn)
           FORMAT(datestring)

TIME       ID(TIMEnnnnnn)
           FORMAT(timestring)

DEFAULTS   DEFAULTDATE(datestring)
           DEFAULTTIME(timestring)
```

Syntax example of CNLcccxx

Figure 13 shows an example message configuration member for Italian.

```
MONTH MONTH1(GENNAIO) MONTH2(FEBRAIO) MONTH3(MARZO) MONTH4(APRILE)
      MONTH5(MAGGIO) MONTH6(GIUGNO) MONTH7(LUGLIO) MONTH8(AGOSTO)
      MONTH9(SETTEMBRE) MONTH10(OTTOBRE) MONTH11(NOVEMBRE)
      MONTH12(DICEMBRE)
DAY DAY1(DOMENICA) DAY2(LUNEDI) DAY3(MARTEDI) DAY4(MERCOLEDI)
    DAY5(GIOVEDI) DAY6(VENERDI) DAY7(SABATO)
DATE ID(DATE1) FORMAT(&mm/&dd/&yy)
DATE ID(DATESHORT) FORMAT(&dd-&mm-&yy)
DATE ID(DATELONG) FORMAT('&dd &mt &yr')
TIME ID(TIME1) FORMAT(&hh:&mm:&ss)
TIME ID(TIME2) FORMAT(&hh-&mm)
TIME ID(TIMESHORT) FORMAT(&hh-&mm-&ss&12)
TIME ID(TIMELONG) FORMAT('&mm MINUTES PAST &hh')
DEFAULTS DEFAULTDATE(&mm/&dd/&yy) DEFAULTTIME(&hh:&mm:&ss)
```

Figure 13. Example: CNLcccxx message configuration member for Italian

Statements/parameters for CNLcccxx

MONTH

Specifies the string to be substituted in translated messages for the corresponding month of the year. For example,

```
MONTH MONTH1(JANVIER)
```

for the French language specifies that the string “JANVIER” will be printed in messages translated to French where the U.S. English message contains the month name “January”.

MONTH1(monthname) - MONTH12(monthname)
Specifies the string to be substituted in translated messages for the months of January through December.

DAY
Specifies the string to be substituted in translated messages for the corresponding day of the week. For example,

```
DAY DAY1(SONNTAG)
```

for the German language specifies that the string "SONNTAG" will be printed in messages translated to German where the U.S. English message contains the day name "Sunday".

DAY1(dayname) - DAY7(dayname)
Specifies the string to be substituted in translated messages for the days of Sunday through Saturday.

DATE
Specifies the format for displaying dates in message texts.

ID(DATExxxxxx)
Specifies the format identifier where *xxxxxx* is 1 to 6 alphanumeric characters. The format identifiers are case sensitive and each identifier must be unique.

FORMAT(datestring)
Specifies the format of the date in the translated message text. The following keywords can appear in the string:

&dd Include the numerical day of the month in the text (01-31)

&dz Include the numerical day of the month in the text but suppress leading zeros (1-31)

&dj Include the numerical day of the year in the text (001-366)

&mm Include the numerical month of the year in the text (01-12)

&mz Include the numerical month of the year in the text but suppress leading zeros (1-12)

&mt Include the name of the month in the text as specified on the MONTH statement

&yy Include the last two digits of the year in the text

&yr Include all four digits of the year in the text

You can use any characters to delimit the keywords.

For example, given the date April 12, 2001, the date formats and their results follow:

```
DATE ID(DATE1) FORMAT(&mm/&dd/&yr)      (4/12/2001)
DATE ID(DATESHORT) FORMAT(&dd-&mm-&yr)  (12-4-2001)
DATE ID(DATELONG) FORMAT('&dd &mt &yr') (12 April 2001)
```

TIME
Specifies the format for displaying time in message texts.

ID(TIMExxxxxx)
Specifies the format identifier where *xxxxxx* is 1 to 6 alphanumeric characters. The format identifiers are case sensitive and each identifier must be unique.

FORMAT(timestring)
Specifies the format of the time in the translated message text. The following keywords can appear in the string:

&hh Include the numerical hour (in the range 00-11) in the text

&hz Include the numerical hour (in the range 00-11) in the text but suppress leading zeros

&h4 Include the numerical hour (in the range 00-23) in the text

&h0 Include the numerical hour (in the range 00-23) in the text but suppress leading zeros

&mm Include the position of the minutes in the text

&ss Include the position of the seconds in the text

&dn Include the number of decimal places, where n is a number from 1 to 6

&ap Specifies an am/pm indicator

You can use any characters to delimit the keywords.

For example, given the time 11:46:12.1234 P.M., the date formats and their results follow:

```
TIME ID(TIME1) FORMAT(&h4:&mm:&ss)               23:46:12
TIME ID(TIME2) FORMAT(&h0-&mm)                   23-46
TIME ID(TIME3) FORMAT(&hh-&mm-&ss.&d3)           11-46-12.123
TIME ID(TIMESHORT) FORMAT('&hh:&mm:&ss &ap')     11:46:12 PM
TIME ID(TIMELONG) FORMAT('&mm MINUTES PAST &hh')  46 MINUTES PAST 11
```

DEFAULTS
Defines the default date and time formats for the language.

DEFAULTDATE(datestring)
Specifies the date format to use when no DATE statements are specified. To code *datestring*, follow the format shown for the FORMAT keyword of the DATE statement.

DEFAULTTIME(timestring)
Specifies the time format to use when no TIME statements are specified. To code *timestring*, follow the format shown for the FORMAT keyword of the TIME statement.

Chapter 19. COFDLFxx (hiperbatch parameters)

The COFDLFxx parmlib member provides the name of the DLF installation exit and sets limits on the amount of storage that the system can use for Hiperbatch. To activate DLF, the CLASS statement describing the group of shared objects must be present in the active COFDLFxx parmlib member (the member named on the START command for DLF). Only one COFDLFxx member can be active at a time. . On the CLASS statement, you must supply the maximum amount of storage that your installation can use for DLF objects (MAXEXPB).

Although only one COFDLFxx member can be active at a time, you can create several members with different limit values for the storage definition parameters, MAXEXPB and PCTRETB, then use the MODIFY system command to change the active parmlib member.

You can monitor the effectiveness of the values you specify for MAXEXPB and PCTRETB through the MODIFY DLF command with the STATUS operand. If the resulting display shows that the percentage of storage in use is consistently low, you might want to switch to a COFDLFxx parmlib member that has lower limits set for MAXEXPB and PCTRETB.

IBM supplies a default DLF parmlib member (COFDLF00) that contains a default CLASS statement. You will probably want to tailor this CLASS statement to meet your installation's needs.

You cannot replace a DLF installation exit while DLF is active. The exit requested that when DLF is started remains in effect for the duration of the DLF address space. To replace the exit, you must stop DLF, replace the exit (or change the parmlib CONEXIT parameter to specify a different exit), and then start DLF again.

Parameter in IEASYSxx (or issued by the operator)

None.

Syntax rules for COFDLFxx

The following syntax rules apply to COFDLFxx:

- The content of the member is one CLASS statement.
- The CLASS statement begins with the statement identifier “CLASS” and ends with the end of file (EOF). No explicit continuation syntax is required.
- Commas and blanks in any combination constitute a delimiter.
- Delimiters or comments can precede the statement identifier.
- Delimiters are not required between parameters with values; the right parenthesis after the specified parameter value is sufficient.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". Comments are allowed between parameters.
- If the class statement contains duplicates of the following parameters, the system uses the first valid occurrence and issues a message that a duplicate parameter was specified:
  - MAXEXPB
  - PCTRETB

– CONEXIT

Syntax format of COFDLFxx

```
CLASS     MAXEXPB(megabytes)
          [PCTRETB(percent)]
          [CONEXIT(exit-name)]
```

Starting DLF

Issue the following command to start DLF:

```
START DLF,SUB=MSTR,NN=xx
```

The two alphanumeric characters (xx) are added to COFDLF to form the name of the COFDLFxx member. If you do not code NN=xx, the system defaults to COFDLF00. Note that DLF will not start unless SUB=MSTR is specified on the START command. SUB=MSTR means that DLF can continue to run across a JES restart.

Statements/parameters for COFDLFxx

CLASS

The CLASS statement indicates the start of the parameters that define the DLF objects.

MAXEXPB(nnnn)

MAXEXPB is a required parameter that allows you to specify, in megabytes, the maximum amount of storage that the system is to use for Hiperbatch. The value for *n* must be a decimal number from 0 through 9999. You can specify up to 4 digits, including leading zeros. The maximum value you can specify is the maximum amount of storage that is available on your system.

Value Range: 0 - 9999 megabytes

PCTRETB(nnn)

PCTRETB is an optional parameter that allows you to specify the percentage of the storage amount (defined by MAXEXPB) that DLF is to use for retained DLF objects. Retained DLF objects are used when the output of one job step is passed to another job or job step. The value for *n* must be a decimal number from 0 through 100. You can specify up to 3 digits, including leading zeros. If the value for *n* is either not in the range or is not specified, the system uses 0.

Value Range: 0 - 100

Default: 0

CONEXIT(routine-name)

CONEXIT specifies the name of the installation exit routine for DLF. This routine must be reentrant and reside in an authorized library in the LNKLST concatenation. For details, see *z/OS MVS Installation Exits*.

Chapter 20. COFVLFxx (virtual lookaside facility parameters)

The virtual lookaside facility (VLF) enables an authorized program to store named objects in virtual storage managed by VLF and to retrieve these objects by name on behalf of users in multiple address spaces. VLF is designed primarily to improve performance by retrieving frequently used objects from virtual storage rather than performing repetitive I/O operations from DASD.

Certain IBM products or components such as LLA, TSO/E, CAS, and RACF use VLF as an alternate way to access data. Since VLF uses virtual storage for its data space there are performance considerations each installation must weigh when planning for the resources required by VLF.

For a description of VLF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

A VLF class is a group of related objects made available to users of an application or component. To get the most benefit from using VLF, consider its use for objects that are:

- Used frequently
- Changed infrequently
- Used by multiple end users concurrently.

To activate a class of VLF objects, VLF requires that a CLASS statement describing that group of objects be present in the active COFVLFxx parmlib member (the member named on the START command used for VLF).

For example, library lookaside (LLA) uses the class of VLF objects named CSVLLA. If the CLASS statement for CSVLLA is not included in the active COFVLFxx parmlib member, LLA cannot use VLF, and many of the performance and operational benefits of LLA will not be available.

IBM supplies a default VLF parmlib member (COFVLF00) that contains CLASS statements for the VLF classes used by IBM-supplied products. You might need to tailor some of these CLASS statements to meet your installation's needs. In addition, your installation can write applications that use VLF, and you must include the CLASS statements for those applications.

There are three items VLF requires for each VLF class used. They are:

1. The name of the class, specified on the required NAME parameter.

   IBM supplies the names of the classes it uses. These names start with the letters A-I.
2. The maximum amount of virtual storage that your installation wants VLF to use for the objects in the class.

   Unless you supply this value on the optional MAXVIRT parameter of the CLASS statement, VLF will use a default value. Generally, the information about the IBM product that uses the VLF class provides some guidance about how to determine an appropriate value for MAXVIRT.

For any given class, the goal is usually to provide an amount of virtual storage large enough to hold a working set of objects—those objects that are used frequently enough to justify keeping them in virtual storage to avoid DASD retrieval.

When you specify the MAXVIRT value, ensure that it is large enough to hold most or all of the frequently-used objects in a VLF class. An excessively small value tends to cause thrashing of the data in that VLF class, while an excessively large MAXVIRT value tends to increase the consumption of auxiliary storage because infrequently-used data is paged out, rather than discarded. Specifying MAXVIRT allows you to limit the maximum amount of auxiliary storage that could be used to back the VLF virtual storage that is holding the objects in the class.

The MAXVIRT value does not represent the *exact* amount of virtual storage that can be used to store objects. A small percentage of the storage (about 10 percent) is used for control information, and, in most cases, VLF begins to "trim" (discard) least recently used objects when the amount of virtual storage used for the class approaches 90 percent of the MAXVIRT value. Therefore, allow some excess.

3. A list of the major names that represent the eligible sources of data for objects in the VLF class.

   How you specify the major names depends on whether the VLF class is a PDS class or a non-PDS class.

   For a PDS class, each major name identifies a unique partitioned data set and consists of a PDS name concatenated to the volume serial number. For a PDS class, use the EDSN and VOL parameters on the CLASS statement to define the major names.

   For a non-PDS class, the major name does not correspond to a partitioned data set. To specify the eligible major names for the class, use the EMAJ parameter on the CLASS statement. For an IBM-supplied class, use the product information to determine if anything other than the name(s) specified in the IBM-supplied default COFVLFxx member are eligible.

In addition to the class and major name, VLF also needs a minor name to identify a unique data object, but the minor names do not come from the COFVLFxx parmlib member.

In using the VLF naming structure, consider the TSO/E use of VLF to manage objects in the IKJEXEC class.

- Each TSO user can have a SYSPROC DDNAME with a different concatenation. When TSO/E identifies the user to VLF, it specifies DDNAME(SYSPROC). VLF then determines the major names (volume serial number and PDS name combinations) that make up the individual major name search order for that user. VLF returns a unique user token (UTOKEN) for that user.
- When that user requests a specific object (such as a CLIST named COPY) from the IKJEXEC class, the user token implicitly specifies the major name search order. When VLF returns the COPY object to the user, the object represents the first object with that minor name to be found in the major name search order for that user.

COFVLFxx parmlib members can be concatenated. When they are concatenated, a class definition can span parmlib members.

Collecting VLF statistics

SMF record type 41 record, subtype 3, allows you to capture SMF data related to the usage of VLF. If you request subtype 3, the system writes this record every 15 minutes. For more information about the type of data SMF provides, see *z/OS MVS System Management Facilities (SMF)*.

Parameter in IEASYSxx (or issued by the operator)

None.

Syntax rules for COFVLFxx

The following syntax rules apply to COFVLFxx:

- The content of the member is one or more CLASS statements.
- A CLASS statement begins with the statement identifier "CLASS" and ends with the end of file (EOF) or when another class statement identifier is found. No explicit continuation syntax is required.
- Commas and blanks in any combination constitute a delimiter.
- Delimiters or comments can precede the statement identifier.
- Delimiters are not required between parameters with values; the right parenthesis after the specified parameter value is sufficient.
- Comments begin with /* and end with */. Comments are allowed between parameters.
- If a class statement contains duplicates of the following parameters, the system uses the first valid occurrence and issues a message that a duplicate parameter was specified:
  - NAME
  - MAXVIRT(nnn)
  - multiple VOL parameters for one EDSN parameter.
- COFVLFxx parmlib members can be concatenated. When they are concatenated, a class definition can span parmlib members.

Syntax format of COFVLFxx

```
CLASS     NAME(classname)
          {EDSN(dsn1) [VOL(vol)] EDSN(dsn2)...}
          {EMAJ(majname1) EMAJ(majname2)...}
          [MAXVIRT(nnn)]
          [ALERTAGE(alert_age)]
```

Starting VLF

Issue the following command to start VLF:

```
START VLF,SUB=MSTR,NN=xx
```

The two alphanumeric characters (xx) are added to COFVLF to form the name of the COFVLFxx member. If you do not code NN=xx, the system defaults to COFVLF00.

Note that VLF will not start unless SUB=MSTR is specified on the START command. SUB=MSTR means that VLF can continue to run across a JES restart. It is recommended that you arrange for the VLF start command to be issued automatically during the IPL process. If VLF is stopped and restarted during the life of an MVS IPL, any request by an application to retrieve or create a VLF object with a user token obtained prior to the restart causes VLF to pass a return code of X'10' back to the application and continue processing. The application should detect the return code and obtain a new token. For more information about using VLF macros, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

COFVLFxx parmlib members can be concatenated. When they are concatenated, a class definition can span parmlib members.

Statements/parameters for COFVLFxx

CLASS

Each group of objects that VLF processes must have a CLASS statement defining it. The CLASS statement indicates that the following parameters define that particular group of objects to VLF.

NAME(*classname*)

NAME(*classname*) specifies the name of the VLF class. The *classname* may be one to seven alphanumeric characters including @, #, and $. IBM-supplied VLF class names begin with the letters A through I, for example, NAME(CSVLLA). See the COFVLFxx member of SYS1.SAMPLIB for the IBM-supplied VLF class names. Installation-supplied class names should begin with the letters J-Z or @, #, or $.

NAME(*classname*) is required on the CLASS statement.

EDSN(*dsn*) [VOL(*vol*)]

For a PDS class, EDSN(*dsn*) identifies a partitioned data set name whose members are eligible to be the source for VLF objects in the class. The *dsn* can be 1 to 44 alphanumeric characters, including @, #, $, and periods (.).

You do not need to specify the volume if the cataloged data set is the desired one. If the data set is not cataloged, or if you have multiple data sets with the same name on different volumes, you must specify VOL(*vol*). Without the volume serial number, an uncataloged data set is not included in the eligible data set name list. The system issues an informational message to the operator identifying any data sets where the system cannot find the catalog entry to extract the volume.

The *vol* can be any combination of alphanumeric characters, including @, #, and $, or a dash (-).

Multiple occurrences of the same data set name with different volumes is acceptable. However, if duplicate entries of the same data set name and the same volume occur, the system issues an informational message and ignores the redundant information.

Do not use the EDSN parameter and the EMAJ parameter on the same CLASS statement.

EMAJ(*majname*)

EMAJ identifies an eligible major name (*majname*) for a non-PDS class, a class that does not have major names and minor names related to partitioned data sets and their members. The *majname* can be 1 to 64 alphanumeric characters except comma (,), blank, or right parenthesis ( )); for example EMAJ(LLA).

Do not use the EMAJ parameter and the EDSN parameter on the same CLASS statement.

MAXVIRT(*nnnnnn*)

MAXVIRT(*nnnnnn*) is an optional parameter that allows you to specify, in 4K blocks, the maximum amount of virtual storage that VLF can use to store objects for the class.

The value for *n* must be a decimal number from 256 through 524288. You can specify up to six digits, including leading zeros. The maximum value is the maximum data space size. If the system cannot obtain the amount of storage specified on MAXVIRT, it obtains as much as possible.

VLF trims least frequently used objects when the amount of storage used for a class approaches 90 percent of its MAXVIRT value. For example, if the MAXVIRT value of a class is 1024, VLF begins trimming when the objects of the class exceed 921 4K blocks of storage. By way of trimming, VLF keeps the most frequently used objects in storage. Small amounts of trimming should be considered normal. A continuous large amount of trimming may indicate that VLF effectiveness is constrained. In this case it may be advisable to address the trimming. To lessen the chance of trimming, specify a large enough value for MAXVIRT. Specifying a MAXVIRT value that is too large can, in some cases, cause high CPU utilization in VLF. The IGGCAS class for CATALOG is one class where caution should be paid to avoid over specifying MAXVIRT.

Note: Subtype 3 of SMF record type 41 can be used to examine any trimming that may be occurring.

If the value for *n* is not in the valid range, the system uses 16 megabytes (4096 4K blocks.)

Default Value: MAXVIRT(4096)

ALERTAGE(*alert_age*)

ALERTAGE specifies the age, in seconds, for objects in the specified class. The IBM Health Checker for z/OS check IBMVLF, VLF_MAXVIRT uses the ALERTAGE to determine whether objects are being trimmed too rapidly to meet the installation's usage goals for VLF. You can specify an ALERTAGE in the range from 0 (which indicates that the check will find no exceptions for this class) to 99999999.

Default Value: ALERTAGE(60)Note that the higher the ALERTAGE value you specify, the more likely it is that the VLF_MAXVIRT check will issue an exception message. See the check description for VLF_MAXVIRT in *IBM Health Checker for z/OS User's Guide* for more information.

Chapter 21. COMMNDxx (commands automatically issued at initialization)

COMMNDxx is an optional, installation-created list of automatic commands the system internally issues as part of system initialization. COMMNDxx is useful for automatic entry of commands that are frequently issued at system initialization.

You cannot use this member to issue JES commands, because JES is not started when the system issues the COMMNDxx commands.

Note: Some commands should not be issued frequently in large numbers. For example, issuing a large number of VARY device commands could cause your system to go into a 07E wait state. While a 07E wait state may not be seen, IPL performance may be affected when a large number of VARY device commands need to be processed from a COMMNDxx member. See *z/OS MVS System Commands* for possible restrictions on other commands.

System trace is activated during the IPL. You can deactivate system trace or change trace options by using the TRACE operator command or by selecting a COMMNDxx parmlib member that contains the options you want. (For information about the TRACE command, see *z/OS MVS System Commands*.)

Note: Do not use COMMNDxx to enter SLIP commands. Instead, use the IEASLPxx parmlib member, with the SET SLIP=xx command specified in COMMNDxx to identify the IEASLPxx member to be used.

Some console-oriented commands affect the operation of a console. You can issue a console-oriented command directly from a console or you can place the command in a COMMNDxx member. If you place a console-oriented command (other than CONTROL M) in COMMNDxx and you do not specify a routing location operand, the system does not execute the command and might not generate an error message. The CONTROL M command does not support routing location operands, but it is valid in COMMNDxx.

When you specify the routing location operands, do so only as described in *z/OS MVS System Commands*.

If you do not include CMD=xx in the system parameter list (IEASYSxx), or if the operator does not specify CMD=xx, the system searches for the COMMND00 member by default. The COMMND00 member, if it exists, is read if CMD=xx is not included in the system parameter list (IEASYSxx) or is not specified by the operator. If the system cannot locate either the specified COMMNDxx member or COMMND00 during initialization, processing continues without automatic commands.

Parameter in IEASYSxx (or issued by the operator)

```
CMD= {aa }
{(aa,bb...)}
```

The two-character identifier (aa,bb,etc.) is appended to COMMND to identify the COMMNDxx member(s) of parmlib. Multiple members can be specified.

Note:

1. Commands issued from COMMNDxx do not show on the console. Therefore, the results of these commands may appear on the console without the operator's seeing the command.
2. Commands are issued in the order that they appear in COMMNDxx, but they are executed as follows:
   - Immediate commands, such as DISPLAY T, are executed sequentially as they are issued from COMMNDxx.
   - Execution of task-creating commands, such as DISPLAY A, is deferred until system initialization is complete. Then, factors such as multitasking, multiprocessing, and competition for resources influence the order in which these commands are executed. Thus, COMMNDxx should *not* be used to issue task-creating commands that must be executed in a specific order, because the execution order of these commands can vary.
3. A command placed in COMMNDxx must look exactly as it does if entered from the console. For example, to place the command SE 'TSO IS UP', CN=CONS1 in COMMNDxx, specify the following:

   ```
   COM='SE 'TSO IS UP',CN=CONS1'
   ```

Support for system symbols

System commands in COMMNDxx can specify system symbols. System symbols can represent any type of variable text in system commands, with the exception of command prefixes and names.

Figure 14 is an example of a COMMNDxx member that includes system symbols. See Figure 5 on page 49 for a sample IEASYMxx member that defines the system symbols in this COMMNDxx member.

```
COM='S VTAM,,,(LIST=&NODE;)'
COM='S NETVIEW,SUB=MSTR,CAT=&CATALOG;,SYS=&SYSNAME;'
COM='S APPC,SUB=MSTR,APPC=02'
COM='S ICR&SYSNAME;.SYSLOG'
COM='SET DAE=01'
COM='SET DAE=&DAE;'
COM='SETXCF START,POLICY,TYPE=CFRM,POLNAME=&POLNAME;'
```

Figure 14. Example: COMMNDxx parmlib member

Be aware that the system does not process system symbols in COMMNDxx during parmlib processing. Instead, the system processes the system symbols in the same way that it processes system symbols in commands that are entered on a console. When a command flows through two or more systems in a sysplex, the target system processes the system symbols in the command text, with a few exceptions. See the section on sharing system commands in *z/OS MVS Planning: Operations* for details about how the system processes system symbols in commands that flow through two or more systems.

Syntax rules for COMMNDxx

The following rules apply to the creation of COMMNDxx:

- Enter only one command per line. To do so, specify the COM=keyword, followed by the command enclosed in single quotation marks.
- Do not specify continuation on any line.
- Lines that begin with an asterisk in column 1 are comments.
- Do not use COMMNDxx to enter MONITOR commands. Instead, use the SETCON MONITOR command to enable or disable Monitor messages.

IBM-supplied default for COMMNDxx

None.

Statements/parameters for COMMNDxx

COM='command'

The specified command will be issued by the system during initialization.

Default Value: No commands will be issued.

Chapter 22. CONFIGxx (standard configuration list)

CONFIGxx is a list of control statements that an installation can use to define a standard configuration of system elements. The system elements include the processors, storage, channel paths, devices, PCIE function identifiers (PFIDs), and volumes. You can use the configuration defined in CONFIGxx in two ways:

1. To compare the differences between the current configuration and the standard configuration as defined in a CONFIGxx member. When the operator issues the DISPLAY command with the M=CONFIG(xx) option, the system displays any differences.
2. To reconfigure some of the system elements. The operator can issue the CONFIG command with the MEMBER option.

Comparing the current and standard configurations

In response to the DISPLAY M=CONFIG(xx) command, the system compares the contents of the CONFIGxx member to the existing configuration. It then displays the differences to the operator. The operator can then resolve these differences by using the CONFIG command.

Matching configurations

If the existing configuration matches the one specified in CONFIGxx, the following message (IEE097I) is sent to the target console:

```
NO DEVIATION FROM REQUESTED CONFIGURATION
```

Nonmatching configurations

If the existing configuration does not match the one specified in CONFIGxx, the following message (IEE097I) is sent to the target console:

```
syselm    DESIRED    ACTUAL
aaa       bbb        ccc
```

- `syselm` is the system element (CHP, CPU, DEVICE, ESTOR, STORAGE, SWITCH, VOLUME, PFID, or PAV).
- `aaa` is the address of the system element.
- `bbb` is the status specified in CONFIGxx.
- `ccc` is the existing status.

Examples

If CPU 0 is actually offline, but is specified as online in CONFIGxx, the message is:

```
CPU       DESIRED     ACTUAL
00        ONLINE      OFFLINE
```

If CPU 0 is actually online and a standard (general purpose) processor, but is specified as offline and a zIIP processor in CONFIGxx, the message is:

```
CPU       DESIRED     ACTUAL
00        OFFLINE     ONLINE
          zIIP        STANDARD
```

Error in CONFIGxx statement

If a statement in CONFIGxx is incorrect, the following message (IEE097I) is sent to the target console:

```
INVALID aaa SPECIFIED BEGINNING xxx
```

- aaa is either
  - REQUEST TYPE if the statement is not recognized or
  - OPERAND if there is an error in specifying an operand.
- xxx is the first sixteen characters of the invalid CONFIGxx statement.

Reconfiguring system elements

The CONFIG command with the MEMBER option enables the operator to reconfigure the system according to the options in the specified CONFIGxx member. This reconfiguration is effective until the operator issues a different CONFIGxx MEMBER command or until the operator IPLs the system. With this command, the operator can reconfigure or verify the configuration of available channel paths, processors, storage elements, PCIE function identifiers (PFIDs), and switches. These are defined by the CHP, CPU, STOR(E=id), ESTOR(E=id), PFID, and SWITCH statements in CONFIGxx. Other statements found in CONFIGxx, such as DEV, VOL, ESTOR(ddddM-ddddM), and PAV, cannot be used for reconfiguration; they can be used only for verification.

The SWITCH record allows verification or reconfiguration of ports on a specified switch device in regard to the dynamic channel path management function. The DCM parameter verifies the enabled status or enables the specified port or port range. The NODCM parameter verifies the disabled status or disables the specified port or port range.

See *z/OS MVS System Commands* for more information about the DISPLAY and CONFIG commands.

Note: If you specify ONLINE and OFFLINE in the same CONFIGxx member, the ONLINE options are processed before the OFFLINE options. The system processes statements that include the ONLINE option in the following order:

1. STOR
2. ESTOR
3. CPU
4. CHP
5. PFID

The system processes statements that include the OFFLINE option in the following order:

1. PFID
2. CHP
3. CPU
4. ESTOR
5. STOR

Parameter in IEASYSxx:

None.

Syntax rules for CONFIGxx

Use the general syntax rules listed in the introduction to this chapter with the following exceptions:

- Continuation statements are not permitted.
- Comment statements are permitted and are indicated by an asterisk in column one.

IBM-supplied default for CONFIGxx

None.

Statements and parameters for CONFIGxx

CHP

Specifies the configuration of the channel paths. The syntax is as follows:

```
CHP    {xx      }[,ONLINE           ][,STATIC ]
       {(xx-xx) }[                  ][,MANAGED]
       {(ALL,id)}[{,OFFLINE}[,UNCOND]]
       {(list)  }[          [,FORCE]]
```

- *xx* is one or two hexadecimal digits that identify the channel path identifiers and indicate the different combinations of channel paths.
- `ALL,id` specifies all channel paths on the side that is identified by the side identifier (id), which can be either 0 or 1. Use CHP ALL,id only when your system is able to be partitioned.
- `list` can be any combination of the elements (except ALL,id) separated by a comma. The list must be enclosed in parentheses.
- `STATIC` indicates that the channel path was statically defined to control units in the channel subsystem. (Used for verification only.)
- `MANAGED` indicates that the channel path was defined to be managed among control units in the channel subsystem. (Used for verification only.)
- `OFFLINE,FORCE`

  CAUTION:
  FORCE is a very powerful option. See the CONFIG command in *z/OS MVS System Commands***.**

Example: This example specifies that channel paths 0, 3, 4, 5, 10, 12, 13, 14, and 15 are to be either verified or reconfigured as online, depending on the command issued.

```
CHP (0,3-5,10,12-15)
```

For information about generating CONFIGxx statements through the z/OS Hardware Configuration Definition (HCD), see *z/OS HCD User's Guide*.

CORE

Specifies the configuration for a core. The syntax is as follows:

```
{CORE } {(x,x[,x]...)} [,ONLINE  [,OFFLINE] [,{STANDARD | ZAAP | ZIIP | ANY}] ]
```

- *x* is a hexadecimal value that specifies the address of the core.

- STANDARD indicates that the core specified to be brought online or offline is expected to be a standard (general purpose) core.
- ZAAP indicates that the core specified to be brought online or offline is expected to be a System z® Application Assist Processor (zAAP) core.
- ZIIP indicates that the core specified to be brought online or offline is expected to be a System z Integrated Information Processor (zIIP) core.
- ANY indicates that the core specified to be brought online or offline can be any type of core.

Note:

1. ANY is the default if no core type is specified.
2. If the processor that is being configured does not match the core type that was specified, the command is rejected, and the system issues message IEE241I with 'PROCESSOR TYPE MISMATCH'.
3. The CORE parameter is rejected when LOADxx PROCVIEW CPU is in effect. See the LOADxx PROCVIEW CPU parameter in "Statements/parameters for LOADxx" on page 625.

CPU or CPUAD

Specifies the configuration for the processors. The statement can be specified as CPU or CPUAD. The syntax is as follows:

```
{CPU  }{(x)         } [,ONLINE  [,{STANDARD | ZAAP | ZIIP | ANY}] ]
{CPUAD}{(x,x[,x]...)} [,OFFLINE [,{STANDARD | ZAAP | ZIIP | ANY}] ]
```

- *x* is a hexadecimal value that specifies the address of the processor.
- STANDARD indicates that the processor specified to be brought online or offline is expected to be a standard (general purpose) processor.
- ZAAP indicates that the processor specified to be brought online or offline is expected to be a System z Application Assist Processor (zAAP).
- ZIIP indicates that the processor specified to be brought online or offline is expected to be a System z Integrated Information Processor (zIIP).
- ANY indicates that the processor specified to be brought online or offline can be any type of processor.

Note:

1. ANY is the default if no processor type is specified.
2. If the processor being configured does not match the processor type that is specified, the command is rejected, and the system issues message IEE241I with PROCESSOR TYPE MISMATCH.
3. The CPU/CPUAD parameter is rejected when LOADxx PROCVIEW CORE is in effect. For more information, see the LOADxx PROCVIEW CPU parameter in "Statements/parameters for LOADxx" on page 625.
4. With LOADxx PROCVIEW CORE,CPU_OK, CPU is accepted and treated as an alias for CORE. For more information, see the LOADxx PROCVIEW CPU parameter in "Statements/parameters for LOADxx" on page 625.

Example: Depending on the command that is issued, this example specifies that the processor addressed by 2 is to be either verified as online or reconfigured online.

```
CPU (2),ONLINE
```

DEV or DEVICE

Specifies the configuration for devices. The syntax is as follows:

```
{DEV   } {dev    } [,xx    ] [,ONLINE ]
{DEVICE} {dev-dev} [,(list)] [,OFFLINE]
         {(list) } [,*     ]
```

- dev specifies the device number and reflects the different combinations of devices. Each *dev* is 1 to 4 hexadecimal digits, optionally preceded by a slash (/).
- list can be any combination of the elements that are separated by a comma. The list must be enclosed in parentheses.
- xx specifies the channel paths to be used to access the devices.
- * is used to verify that at least one channel path is online and can access the devices, regardless of which channel path it is.

The system displays a deviation message (IEE097I) for any of the following conditions:

- If channel paths are specified and all of the specified channel paths to the devices are not online or offline.
- If * is specified and there is no channel path online to access the devices.
- If channel paths are not specified and * is not specified and all of the channel paths to the devices are not online or offline.

Examples:

- To cause the system to verify that the devices with device numbers 334 and 33A can be accessed through channel path 12, code:

  ```
  DEV (334,33A),12,ONLINE
  ```

- To cause the system to verify that the devices with device numbers 340 through 34F can be accessed by all of their associated channel paths, code:

  ```
  DEV 340-34F,ONLINE
  ```

- To cause the system to verify that devices 100 and 2000 through 21F0 can each be accessed by channel paths 01, 11, and 21, code:

  ```
  DEV (100,2000-21F0),(01,11,21)
  ```

  Example: This example assumes the default of ONLINE.

- To cause the system to verify that devices A451, A453, and A455 can be accessed by at least one channel path, code:

  ```
  DEV (A451,A453,A455),*,ONLINE
  ```

  The * indicates that it is unimportant which channel paths are used to access the devices.

For information about generating CONFIGxx statements through the z/OS Hardware Configuration Definition (HCD), see *z/OS HCD User's Guide*

ESTOR

Specifies the configuration for elements of expanded storage. The syntax is as follows:

```
ESTOR  {(E=id)       } [,ONLINE ]
       {(ddddM-ddddM)} [,OFFLINE]
```

- *id* is the identifier of a storage element. The identifier can be one to four hexadecimal digits from X'0000' to X'FFFF'.
- *dddd* is one to four decimal digits that must be a multiple of 4, followed by an M (megabytes), and cannot exceed a value of 4095. These values are the starting and ending addresses of the section of storage to be verified. You can specify only one range. The starting and ending address must not be the same.

Note:

1. A CONFIGxx parmlib member that contains the ESTOR(ddddM-ddddM) can be used only as the target of a DISPLAY M=CONFIG command. It cannot be used to reconfigure storage.
2. ESTOR is not supported in the z/Architecture environment. Its specification is tolerated by CONFIGxx and ignored during IPL.

Example: This example indicates that a section of expanded storage, whose identifier is X'03' is to be either verified or reconfigured as online.

```
ESTOR(E=03),ONLINE
```

Example: This example indicates that a section of expanded storage (location 0 through location 67,108,864) is to be verified as online.

```
ESTOR (0M-64M),ONLINE
```

PFID

Specifies the configuration of the PCIE function identifiers (PFIDs). The syntax is as follows:

```
PFID   {(xx)   }[,ONLINE|ON           ]
       {(aa-bb)}[,{OFFLINE|OFF}[,FORCE]]
       {(list) }
```

- xx is a single PFID. The PFID can have a hexadecimal value from 0 to X'FF'.
- aa-bb is a range of PFIDs. The starting and ending PFIDs can have values from 0 to X'FF'.
- list is one or more single PFIDs, ranges of PFIDs, or a combination of single PFIDs and ranges of PFIDs.
- ONLINE or ON brings the specified PFID or PFIDs online.
- OFFLINE or OFF takes the specified PFID or PFIDs offline.
- OFFLINE,FORCE

  CAUTION:
  FORCE is a very powerful option. See the CONFIG command in *z/OS MVS System Commands***.**

For information about generating CONFIGxx statements through the z/OS Hardware Configuration Definition (HCD), see *z/OS HCD User's Guide*.

STOR or STORAGE

Specifies the configuration for sections of storage. Multiple ranges can be specified. The statement can be specified as STOR or STORAGE. The syntax is as follows:

```
{STOR   } {ddddddK-ddddddK} [,ONLINE ]
{STORAGE} {xxxxxxxxxxxxxxx-xxxxxxxxxxxxxxx} [,OFFLINE]
          {ddddX-ddddX    }
          {(E=id          }
          {(list          }
```

- *ddddddd* is one to seven decimal digits, followed by a K, which are the starting and ending addresses of the section, and cannot exceed a value of 4194303. Each address represents a multiple of 1024 bytes. If necessary, the system rounds the low address down to the next lower 4 K boundary. (This rounding is done to begin and end a section of storage on a 4 K boundary.) The system does not reconfigure a section of storage (in response to a CONFIG command) when the section is specified in this manner.
- *xxxxxxxxxxxxxxx* is one to sixteen hexadecimal digits that address the first and last bytes of the section. If necessary, the system rounds the low address down to the next lower 4 K boundary. (This rounding is done to begin and end a section of storage on a 4 K boundary.) The system does not reconfigure a section of storage (in response to a CONFIG command) when the section is specified in this manner.

  Note: You can insert underscores in hexadecimal specifications to make them easier to read. Underscores are ignored during processing, and do not count towards the number of digits that are used for the address.
- *ddddd* is one to five decimal digits, followed by a multiplier, which are starting and ending addresses of the section, and cannot exceed a value of 16383. The valid multipliers are:
  M Megabytes (2**20 bytes)
  G Gigabytes (2**30 bytes)
  T Terabytes (2**40 bytes)
  P Petabytes (2**50 bytes)

  The starting and ending addresses must not be the same.

  Note: You can also specify the addresses in explicit hexadecimal notation (X'xxxxx') followed by a multiplier.
- `E=id` specifies a storage element that is identified by the storage element identifier (ID), which is one hexadecimal digit.
- `list` can be any combination of the elements (except E=id) separated by a comma. The list must be enclosed in parentheses.

Example: This example indicates that a section of storage (location 8,388,608 through location 33,554,431) is to be verified as online.

```
STOR 8192K-32768K,ONLINE
```

SWITCH

Specifies the configuration of the switch ports for Dynamic Channel Path Management. If no SWITCH statement is specified for a switch port, then no verification occurs. The specification is routed to all systems in the logical partition cluster to ensure that all systems run with the same configuration of managed channel paths. The syntax is as follows:

```
SWITCH    {(ssss[,pp[-pp]])},DCM|,NODCM
          {ALL             }
```

- *ssss* is the device number of the switch device to be verified.
- pp[-pp] is the port number or port number list.
- ALL specifies all switch devices.
- DCM specifies that Dynamic Channel Path Management is allowed for the subject port.
- NODCM specifies that Dynamic Channel Path Management is not allowed for the subject port.

For information about generating CONFIGxx statements through the z/OS Hardware Configuration Definition (HCD), see *z/OS HCD User's Guide*.

VOL or VOLUME

Specifies the configuration for DASD volumes. The statement can be specified as VOL or VOLUME. The device number on which a volume should be mounted can optionally be specified. The syntax is as follows:

```
{VOL   } {v    }
{VOLUME} {v=dev}
{list  }
```

- *v* is the volume name.
- dev is the device number and is 1 to 4 hexadecimal digits, optionally preceded by a slash (/). Any digit can be replaced with an X to indicate that the value of that digit is not important. The X and the hexadecimal numbers A, B, C, D, E, and F must be in uppercase.
- list can be any combination of the elements separated by a comma.

Example: This example specifies that PAGE1 and P00045 are to be verified as mounted. Mount P00045 on a device with a device number in the range 305 through 3F5.

```
VOL PAGE1,P00045=3X5
```

PAV

Specifying the Parallel Access Volumes (PAV) statement causes the display of a list of all unbound PAV-alias devices. No parameters are specified with the PAV statement. The syntax is as follows:

```
PAV
```

Chapter 23. CONSOLxx (console configuration definition)

CONSOLxx is an installation-created member of parmlib in which you can define a console configuration to meet the particular needs of your installation.

In CONSOLxx, you define multiple console support (MCS), SNA multiple console support (SMCS) consoles, and hardware management console multiple console support (HMCS) consoles. You define the specific characteristics of MCS, SMCS, HMCS, subsystem consoles and the system console. You might need to have PF keys on one console issue one group of system commands while on another console you need the PF keys set to issue a different set of commands. You might need to have particular messages routed to one console, and you may want those messages deleted in a certain way. You might need a cluster of consoles to serve different functions for a subsystem. One console in the cluster may display, in a particular format, only the messages associated with a particular group of routing codes. Through CONSOLxx, you can specify the configuration and have the system automatically initialize the consoles.

Note: In a sysplex environment, the limit of consoles you can define varies depending on the operation mode of the console support. In shared mode, up to 99 MCS, SMCS, HMCS, and subsystem consoles can be defined per sysplex. In distributed mode, up to 250 MCS, SMCS, HMCS, and subsystem consoles can be defined per z/OS image. For more information about operation modes, see *z/OS MVS Planning: Operations*.

Through CONSOLxx, you can specify initialization values for all console configurations. For example, you can define the characteristics of the hardcopy message set and set routing codes for messages that do not have routing information. You can specify the parmlib member that contains tracing options for the operations services (OPS) component.

You also can specify the operator logon requirements for the consoles at your installation (except the system console). Setting operator logon requirements depends on the security policy in effect at your installation. For information about how operator logon requirements relate to securing access to system commands, and examples of defining consoles through CONSOLxx, see *z/OS MVS Planning: Operations*.

You can use some operator commands to dynamically change the console attributes. In distributed mode, the changes made by such commands are only effective when the console is active. The CONSOLxx values will be used the next time the console is activated. In both shared and distributed mode, for the most part, operator commands cannot change the attributes of an inactive console. See "Related commands" on page 233 for more information.

SET CON= can be used to dynamically add a console definition and to dynamically change console settings and attributes.

Note: MCS consoles are locally attached to the system through control devices that do not support Systems Network Architecture (SNA) protocols. SMCS consoles use z/OS Communications Server to communicate with the system and may be remotely attached to the system. SMCS consoles are only available for use when

the z/OS Communications Server is active. HMCS consoles are on the Hardware Management Console Integrated 3270 Console window.

Using CONSOLxx in a sysplex

Take into account the following considerations when you are using CONSOLxx in a sysplex:

- When two or more systems in a sysplex require a CONSOLxx member, you can do *one* of the following:
  - Code a separate CONSOLxx member for each system in the sysplex (the least efficient method).
  - Code a single CONSOLxx member for all systems in your sysplex. Specify parameters with *sysplex scope* to be used by all systems in the sysplex. Consider using system symbols to represent unique values in the member.
- CONSOLxx parameters with *sysplex scope* are valid only for the first system that enters a sysplex. Because these parameters are ignored by systems that later join a sysplex, you do not need to set them up to specify unique values for different systems in a multisystem environment. For a complete list of parameters in CONSOLxx that have sysplex scope, refer to *z/OS MVS Planning: Operations*.

  If different systems require unique values on parameters that do not have sysplex scope, you can use system symbols to represent those unique values in a shared CONSOLxx member. When each system processes CONSOLxx, it replaces the system symbols with the values it has defined to them. See Chapter 2, “Sharing parmlib definitions,” on page 33.
- A named console can be defined on multiple systems, but can only be active on one system at one time.
- The system uses default values for the CONSOLxx statements INIT, HARDCOPY, and DEFAULT if you do not code them. If the default values for these statements are acceptable to your installation, do not code them for the systems in your multisystem environment.
- In distributed mode, the console attributes specified in CONSOLxx are used whenever the console is activated. If the VARY command changes the attributes of an active console, those changes remain until the console is deactivated. The CONSOLxx values will then be used the next time the console is activated. In both shared and distributed mode, for the most part, the VARY command can not change the attributes of an inactive console.
- For a complete list of the CONSOLxx values that can be changed by operator command, the sysplex or system scope of the values, and whether or not these values are restored to CONSOLxx parmlib member settings or IBM defaults by a single system IPL or by reinitialization of the sysplex, see the topic "Summary of CONSOLxx and Commands to Change Values" in *z/OS MVS Planning: Operations*.

Related members of parmlib

CONSOLxx provides a way to centralize the definitions of the console configuration for your installation. Within CONSOLxx, you specify:

- MPFLSTxx parmlib members for message processing control.
- The MMSLSTxx parmlib member to display translated U.S. English messages into another language that your installation has provided.
- The PFKTABxx parmlib member to define any PFK tables you require.
- The CNGRPxx parmlib member to define SYNCHDEST and AUTOACT groups.

- The MSGFLDxx parmlib member to define your installation policy for controlling message flooding situations.

Specifying MPFLSTxx members, the MMSLSTxx member, and the PFKTABxx member within the CONSOLxx member makes it easier to maintain these related members of parmlib rather than setting them after IPL with the SET operator command. See Chapter 71, "MPFLSTxx (message processing facility list)," on page 649, Chapter 70, "MMSLSTxx (MVS message service list)," on page 645, and Chapter 75, "PFKTABxx (program function key table definition)," on page 693. See Chapter 17, "CNGRPxx (specify console groups)," on page 203 for information about specifying CNGRPxx.

CONSOLxx is closely related to the Add Device panel of hardware configuration definition (HCD). The device number you specify in CONSOLxx must match the device number on the panel. See *z/OS HCD User's Guide*.

See also "Parameter in IEASYSxx (or supplied by the operator)" on page 236

Related commands

Through the CONTROL, SET, and VARY commands, you can change some of the characteristics specified in CONSOLxx.

- In shared mode, the changes made through these commands can last the life of the sysplex.
- In distributed mode, the changes can only persist while the console is active. When the console is activated, the attributes of the console are always obtained from CONSOLxx for MCS, SMCS, HMCS, or subsystem consoles.

Note that if the console is inactive, the request to change the console attribute will be rejected (with a few exceptions). See the specific command description in *z/OS MVS System Commands* for more details.

For example, when system S1 is IPLed into a sysplex, the console CONS1 is defined in the CONSOLxx member with AUTH(INFO). The operator enters a VARY CN(CONS1),AUTH=SYS to give the console a higher authority. System S1 is then removed from the sysplex. When S1 is IPLed and the CONSOLxx member is processed, the definition of CONS1 results in message IEA196I, which indicates that the AUTH parameter is ignored. CONS1 will continue to have SYS authority.

For more information, see *z/OS MVS System Commands*.

CONSOLxx contains four optional statement types:

- CONSOLE, which is optional if processor supports the system console. If not, a CONSOLE statement is required for the first system in a sysplex.
- INIT
- HARDCOPY
- DEFAULT.

CONSOLE statement

The CONSOLE statement lets you define each console by device number on the CONSOLE statement.

Within a sysplex, different MCS consoles on different systems might have the same device number. Each console must be defined by a unique name that identifies it to

the sysplex. The console name you define on the CONSOLE statement is used to identify a given console in system commands (MGCRE), write-to-operator (WTO), write-to-operator-with-reply (WTOR), and installation exits.

All consoles, except for the system console, require a console name to be specified. If no name is specified, the console definition is rejected. For the system console, if a name is not specified, a name is generated by the system.

A console defined to more than one system can be active on only one system in the sysplex at a time. If different attributes for the same console are defined in separate CONSOLxx members on different systems:

- In shared mode, the console attributes defined in the first active system in the sysplex take effect.
- In distributed mode, the same console can have different attributes on different systems.

Besides defining the console by device number, you can also specify initial values for the console's attributes with the CONSOLE statement.

You must have a CONSOLE statement for each device that you want to use as a console. Use the Add Device panel in hardware configuration definition (HCD) to specify the device number and the unit type for MCS consoles only. SMCS consoles require the terminals to be defined for use via z/OS Communications Server. For more information about defining devices for use by z/OS Communications Server, see *z/OS V2R1.0 Communications Server: SNA Resource Definition Reference*. The entries on the panel must match the CONSOLE statement. A CONSOLxx member can include multiple CONSOLE statements; one for each console in the configuration. For more information about HCD's Add Device Panel, see *z/OS HCD User's Guide*.

A system command (SETCON DELETE,CN=) and a sample service (IEARELCN) allow you to remove the definition of an MCS, SMCS, HMCS, or subsystem console from a system or sysplex. Before you can delete the definition, the console needs to be inactive.

The IEARELCN service is available as a sample job in Samplib member. For information about the console service, see *z/OS MVS Planning: Operations*.

INIT statement

You can take the following steps using the INIT statement:

- Specify the limits for WTL, WTO, and WTOR buffers.
- Specify one or more MPFLSTxx members to use with this CONSOLxx member.
- Specify the PFKTABxx member to use with this CONSOLxx member.
- Specify the MMSLSTxx member to use with this CONSOLxx member.
- Specify the CNGRPxx member to set up console groups.
- Specify the Parmlib member that contains component trace options for the operations services (OPS) component.
- Activate the action message retention facility.
- Activate the WTO installation exit IEAVMXIT.
- Specify the MONITOR command to display mount messages.
- Specify an MVS command delimiter so an operator can issue multiple commands.

- Specify the maximum number of seconds the ROUTE *ALL or ROUTE *systemgroupname* command waits before aggregating responses.
- Specify the z/OS Communication Server APPLID that SMCS is to use on this system.
- Specify the z/OS Communication Server Generic Resource name that SMCS is to use in this sysplex.
- Specify the MSGFLDxx parmlib member to define your installation policy for controlling message flooding situations.

The INIT statement is optional. If you do code an INIT statement, you can code only one in a CONSOLxx member. See "Syntax and parameters for an INIT statement" on page 250.

HARDCOPY statement

You can take the following steps using the HARDCOPY statement:

- Specify whether the hardcopy medium active at initialization is SYSLOG, OPERLOG or both
- Specify the routing codes for messages to be included in the hardcopy message set
- Specify the kinds of messages (commands, responses or status display) to be included in the hardcopy message set
- Specify whether hardcopy records should have a 4-digit year

The HARDCOPY statement is optional. If you do not specify HARDCOPY, the system uses SYSLOG as the hardcopy medium. You can have only one HARDCOPY statement in a CONSOLxx member. See "Syntax and parameters of the HARDCOPY statement" on page 255.

For more information about the hardcopy message set, see *z/OS MVS Planning: Operations*.

DEFAULT statement

You can take the following steps on the DEFAULT statement:

- Define the default routing codes for unsolicited WTO and WTOR messages that have no routing codes, no descriptor codes, and no console ids assigned.
- Define whether operators must log on to the system before issuing commands from MCS, SMCS, or HMCS consoles.
- Specify whether you want hold mode for consoles.
- Specify the maximum value of a reply ID in the sysplex.
- Define a console group to receive synchronous WTO or WTOR messages that the system issues.

The DEFAULT statement is optional. If you specify DEFAULT, you can have only one DEFAULT statement in a CONSOLxx member. See "Syntax and Parameters of the DEFAULT statement" on page 257.

If you do not code the DEFAULT statement, the system assigns routing codes 1 through 16 to messages that have no other routing attributes, and MCS and HMCS consoles do not require LOGON. SMCS consoles will require logon.

Parameter in IEASYSxx (or supplied by the operator)

```
CON=  {aa}
      {(aa[,L][,NOJES3])}
      {modespec}
      {NONE}
      {(NONE[,L][,NOJES3][,modespec])}
      {NOJES3}
      {(NOJES3[,modespec])}
```

The two alphanumeric characters (aa) are appended to CONSOL to form the name of the CONSOLxx member of Parmlib. If you specify the L option in IEASYSxx, or in reply to the 'SPECIFY SYSTEM PARAMETERS' message, the system lists (displays) on the operator's console, all of the statements that are in the CONSOLxx member. If CON=NONE is specified, the system is initialized with the IBM defaults for the values in CONSOLxx.

Specify the NOJES3 option if JES3 is installed on your system, but is not to be used. For systems that have both JES2 and JES3 installed, but run only JES2, the NOJES3 option allows you to omit the comma separating the REPLY ID from the command text when using short form replies to responding to system requests.

The value for the console support mode option *modespec* is either DISTRIBUTED or SHARED. The default value is DISTRIBUTED.

Syntax rules for CONSOLxx

Use the general syntax rules listed in "General syntax rules for the creation of members" on page 9 with the following exceptions and additions:

- In shared mode, you can define up to 100 consoles (including the system console) within a CONSOLxx member. The maximum number of consoles within a system or sysplex is 99, plus one system console for each system.
- In distributed mode, you can define up to 251 consoles (including the system console) within a CONSOLxx member. The maximum number of consoles per system in the sysplex is 250 and 99 of which can be concurrently active, plus one system console for each system.
- Data must be contained in columns 1-71; the system ignores columns 72-80.
- Comments can appear in columns 1-71 and must begin with /* and end with */.
- You do not need to code delimiters between parameters; you can use either blanks or commas between parameters.
- One or more blanks can precede or follow the statement types (CONSOLE or INIT, for example).
- At least one blank must immediately follow the statement types (INIT, DEFAULT, HARDCOPY, and CONSOLE).
- Parameter values must be set off by parentheses. If you code multiple values on certain keywords, separate the values with a comma. Some keywords will allow you to separate multiple values with a blank. See the keyword documentation for specific syntax.
- Do not use blanks in the middle of a keyword, in the middle of a value, or between the parameter and the left parentheses before the value.
- A statement type must be the first data item on a record.
- You can code comments before any statement type.

- A statement type continues to the next statement type in the member or until the end of the member.
- System symbolics can be used anywhere within the CONSOL*xx* member.
- For the CONSOLE statement, use the following rules:
  - On each CONSOLE statement, DEVNUM is required and must be the first parameter.
  - Any values you specify for the NAME parameter on the CONSOLE statement must be unique across the sysplex.
  - Do not use a name on the NAME parameter that might be confused with a device number. For example, the console name BEAD is not recommended, since it might be confused with device number X'BEAD'.

When you define consoles within a sysplex, note the different console constraints in shared mode and distributed mode shown in Table 17.

Table 17. Console constraints within a sysple

Console services mode	Consoles can be defined	Active consoles allowed
shared mode	Up to 99 MCS, SMCS, HMCS, and subsystem consoles per sysplex, plus one system console per system	Up to 99 MCS, SMCS, HMCS, and subsystem consoles per sysplex, plus one system console per system
distributed mode	Up to 250 MCS, SMCS, HMCS, and subsystem consoles plus one system console per system	Up to 99 MCS, SMCS, HMCS, and subsystem consoles plus one system console per system

IBM-supplied default for CONSOLxx

None. You can create a CONSOLxx member by modifying the IBM-provided sample member IEACONXX in SYS1.SAMPLIB. If no CONSOLxx member is supplied, the system messages go to the system console and the system uses the defaults for the other values in CONSOLxx. For information about when to initialize the system with CON=NONE, see *z/OS MVS Planning: Operations*.

Statements/parameters for CONSOLxx

CONSOLxx includes the following statements:

- "Syntax and parameters of a CONSOLE statement"
- "Syntax and parameters for an INIT statement" on page 250
- "Syntax and parameters of the HARDCOPY statement" on page 255
- "Syntax and Parameters of the DEFAULT statement" on page 257

Syntax and parameters of a CONSOLE statement

The following shows the syntax of the CONSOLE statement.

```
CONSOLE   DEVNUM  {(devnum)             }
                  {(SUBSYSTEM)          }
                  {(SYSCONS)            }
                  {(SMCS)               }
                  {(HMCS)      }

          UNIT  {(unittype)}
                {(PRT)     }

          NAME  (conname)

          AUTH  {(MASTER)           }
                {(INFO)             }
                {([SYS][,IO][,CONS])}
                {(ALL)              }

          USE  {(FC)}
               {(MS)}
               {(SD)}

          ROUTCODE  {(ALL)                   }
                    {(NONE)                  }
                    {(nnn[,nnn-nnn][,nnn]...)}

          LEVEL  {(ALL)                       }
                 {([ALL][,NB])                }
                 {([R][,I][,CE][,E][,IN][,NB])}

          CON  {(Y)}
               {(N)}

          SEG(nn)

          DEL  {(Y) }
               {(R) }
               {(RD)}
               {(N) }
               {(W) }

          RNUM  {(nn)}
                {(5) }

          RTME  {(nnn)}
                {(2)  }

          MFORM  {(M)              }
                 {([J][,S][,T][,X])}

          AREA  {(nn[,nn]...)}
                {(NONE)      }

          PFKTAB(tablname)

          MONITOR({JOBNAMES[-T]}{,SESS[-T]}{,STATUS})

          MSCOPE  {([sysname|*][,sysname|*]...)}
                  {(*ALL)                      }

          CMDSYS  {(sysname)}
                  {(*)      }

          RBUF  {(nn)}
                {(15)}
```

```
AUTOACT  (groupname)

SYSTEM  (sysname)

LU  (nnnnnnnn)

LOGON  {(REQUIRED)}
       {(OPTIONAL)}
       {(AUTO)    }
       {(DEFAULT) }

INTIDS{(Y)|(N)}

UNKNIDS{(Y)|(N)}

ALLOWCMD{(Y)|(N)}
SUPSBY{(Y)|(N)}
```

CONSOLE

CONSOLE indicates the beginning of a statement that defines the characteristics of a console.

```
DEVNUM     {(devnum)    }
           {(SUBSYSTEM) }
           {(SYSCONS)   }
           {(SMCS)      }
           {(HMCS)      }
```

DEVNUM specifies the type of console. DEVNUM is **required** and must be the first keyword on the CONSOLE statement.

devnum **must** be the same as the number that was specified for the device on the Add Device panel in HCD.

Note: The system pins UCBs for console devices defined in CONSOLxx at IPL time, and the UCBs are only unpinned when a console definition is removed using the console removal definition service, which may be invoked using the IEARELCN sample program that resides in SYS1.SAMPLIB. This means that to delete a console device with HCD, an IPL is required, unless the console definition is deleted using the SETCON DELETE, CN= command or using IEARELCN.

Value Range: 1 to 4 hexadecimal digits, optionally preceded by a slash (/).

SUBSYSTEM indicates that this console is reserved for subsystem use.

Note:

1. The only keywords that are valid with the DEVNUM(SUBSYSTEM) setting are AUTH and NAME.
2. A subsystem console can have MASTER authority, but it cannot be assigned at IPL. If this subsystem console has been specified with AUTH(Master) at IPL, the default value of the AUTH parameter is set to AUTH(ALL). Otherwise, the default is set to AUTH(INFO).

SYSCONS indicates that this console is the system console attached to this processor. The use of the SYSCONS keyword is optional. The first time you put the system console into problem determination (PD) mode (by issuing VARY CN(*),ACTIVATE from the system console), its attributes will be taken from the CONSOLE statement with DEVNUM(SYSCONS). If there is no such statement, a default set of attributes will be used (see “System Console Defaults” below). The system console's attributes can be changed with the VARY CN command.

The SYSCONS is an extended MCS console. See *z/OS MVS Planning: Operations* for more information.

Note: If AMRF(N) on the INIT statement is specified, then the DOM attribute for the SYSCONS extended MCS console will be set to DOM(ALL) instead of DOM(NORMAL). If AMRF(Y) on the INIT statement is specified or defaulted, then the DOM attribute for the SYSCONS extended MCS console will be set to DOM(NORMAL).

For more information about the DOM attribute for extended MCS consoles, see *z/OS MVS Planning: Operations*.

If your MVS system is running as a guest on the VM/ESA* system, and the VM system is at release level 1.1.1 or later, VM will simulate the system console hardware regardless of the processor type.

System Console Defaults: The system console receives the normal default values for the keywords that are valid with DEVNUM (SYSCONS), except for the following keywords:

- NAME — The system generates a name for the system console. (For additional information, see the following description of the NAME keyword.)
- AUTH — The system console is always forced to have MASTER authority.
- MSCOPE — The system console default is MSCOPE(*).
- LEVEL — The system console default is ALL,NB.

Note: The only keywords that are valid with DEVNUM(SYSCONS) are NAME, ROUTCODE, LEVEL, MONITOR, MSCOPE, AUTOACT, CMDSYS, INTIDS, UNKNIDS, and ALLOWCMD; all others are ignored.

SMCS: For a SMCS console, DEVNUM(SMCS) must be specified. DEVNUM(SMCS) is mutually exclusive with the UNIT and SYSTEM keywords.

HMCS: For a HMCS console, DEVNUM(HMCS) must be specified. DEVNUM(SMCS) is mutually exclusive with the UNIT and SYSTEM keywords.

For more information about using SMCS, devices or emulators that can be used with SMCS, and HMCS, see *z/OS MVS Planning: Operations*.

```
UNIT     {(3270-X)}
         {(3277-2)}
         {(3278-2)}
         {(3278-2A)}
         {(3278-3)}
         {(3278-4)}
         {(3279-2A)}
         {(3279-2B)}
         {(3279-2C)}
         {(3279-3A)}
         {(3279-3B)}
         {(PRT)   }
```

UNIT specifies the unit type of the console.

The unit type must be a valid console device listed here. See Table 18 on page 259 for the devices that can be used as MCS consoles.

PRT specifies the console is a printer. The Add Device panel in HCD must specify a valid device type that can be defined as a printer.

Note:

1. Specify 3270-X for devices that meet the following criteria:
   - The device supports the 3270 data stream and Read Partition Query (such as a 3471 or 3472 IBM InfoWindow terminal).
   - The device is attached to a control unit that supports Read Partition Query (should Read Partition Query fail, MCS will attempt to bring the device online with 3277-2 display attributes).
2. This keyword is mutually exclusive with DEVNUM(SMCS) and DEVNUM(HMCS). SMCS and HMCS will always determine the type of device that is being used.

Default: If you do not code the UNIT keyword, the system uses the information entered through HCD for the device number to determine the unit type. If the HCD information indicates that it is a display device, the system will default the UNIT value to 3270-X.

NAME(conname)

NAME specifies the console name that uniquely identifies the console. The console name is required on all consoles except for the system console. Any values you specify for the NAME parameter (names identifying the console) must be unique across the sysplex.

Attention: If you do not specify NAME for a console, the console definition will be rejected with an error message. The only exception to this rule is the system console.

Value Range: *conname* is from 2 to 8 characters. The first character of *conname* must begin with the letters A through Z or with a #, $, or @; the remaining characters can be A through Z, 0 through 9, or #, $, or @.

If you want to make the same console available to different systems in a sysplex, you must specify the same value for the NAME parameter on the CONSOLE statement in each system's CONSOLxx Parmlib member.

Do not specify a console name that could be confused with a device number. For example, do not use a console name like BEAD, since it can be confused with device number x'BEAD'.

Do not use HC, INSTREAM, INTERNAL, SYSIOSRS, UNKNOWN, OPERLOG, SYSLOG, LOGON, or LOGOFF for *conname*; these names are reserved for the system.

Naming the System Console: IBM strongly suggests that you specify a name for the system console in CONSOLxx. Select a unique name for the system console that cannot be confused with a valid device number.

Default: If your operator specifies CON=NONE, or if you do not name the system console in CONSOLxx, MVS tries to use the name of the system to which the console is attached as the name of the system console, as long as that name is unique and cannot be confused with a device number that the system can use. If you do not name the system console in CONSOLxx, use a system name that cannot be confused with a device number that the system can use.

If you specify a system name that the system can interpret as a valid device number, the system does not use SYSNAME as the name of the system console. If the system cannot use SYSNAME for the system console

name, or if the system console name is not unique, the name of the system console is SYSCNxxx, where xxx is a three-character suffix generated by the system.

If you try to define a console whose name is the same as an existing console, the system will reject the definition.

```
AUTH     {(MASTER)          }
         {(INFO)            }
         {([SYS][,IO][,CONS])}
         {(ALL)             }
```

AUTH specifies the group of operator commands that can be entered from the console. IBM strongly suggests using a security product, such as RACF, to control commands instead of using AUTH, especially with SMCS. For more information about SMCS and console security see *z/OS MVS Planning: Operations*.

MASTER indicates that this is a console with master-level authority.

From a console with master authority, you can enter all MVS operator commands. The corresponding authority levels for JES3 are:

- MASTER: JES3 authority level=15
- CONS: JES3 authority level=10
- I/O: JES3 authority level=10
- SYS: JES3 authority level=5
- INFO: JES3 authority level=0

INFO specifies that any informational commands can be entered from this console.

SYS specifies that system control commands and informational commands may be entered from this console.

IO specifies that I/O control commands and informational commands may be entered from this console.

CONS specifies that console control commands and informational commands may be entered from this console.

ALL specifies that information, system control, I/O control, and console control commands may be entered from this console.

Note:

1. AUTH is not valid with UNIT(PRT).
2. You can separate multiple values with a blank or a comma.

For information about which commands can be entered from a console with a specific authority level, see *z/OS MVS Planning: Operations*.

Default: INFO is the default for all consoles except the system console, which is forced to be AUTH(MASTER).

USE {(FC) | (MS) | (SD)}
USE specifies how the display console is used.

FC defines a full-capability console able to enter commands and receive status displays and messages. If the console is a display device, specify USE(FC).

MS defines a message stream console. If the console is a printer, specify USE(MS).

SD defines a status display console.

Note:

1. The USE keyword is ignored if specified with DEVNUM(SYSCONS).
2. SMCS and HMCS do not support USE(MS) or USE(SD) consoles. Only USE(FC) is accepted for an SCMS or HMCS console.

Default: If you do not specify USE:

- FC is the default if the console is a display device
- MS is the default if the console is a printer.

```
ROUTCODE     {(ALL)                  }
             {(NONE)                 }
             {(nnn[,nnn-nnn][,nnn]...)}
```

ROUTCODE specifies the routing codes assigned to the console.

Value Range: ALL specifies all routing codes, 1 through 128. nnn specifies a decimal value from 1 through 128. nnn-nnn specifies a range of decimal value with the lower value first. You can separate multiple values with a space or a comma.

Note: Do not assign routing code 11 to a console, because it is meant for programmer information rather than operator information.

Default: NONE is the default for all consoles.

```
LEVEL      {(ALL)                  }
           {([ALL][,NB])           }
           {([R][,I][,CE][,E][,IN])}
```

LEVEL specifies the message levels for the console.

ALL indicates that the console is to receive all messages.

NB This console is to receive no broadcast messages.

R This console is to receive the messages that require an operator reply.

I This console is to receive immediate action messages.

CE This console is to receive critical eventual action messages.

E This console is to receive eventual action messages.

IN This console is to receive informational messages.

To receive all message levels but broadcast messages, you must specify the following; you can separate multiple values with a space or a comma.

```
LEVEL (R,I,CE,E,IN)
```

Default: *ALL* (except for the system console, which defaults to LEVEL(ALL,NB)).

```
CON      {(Y)}
         {(N)}
```

CON indicates whether the console is to function in conversational or nonconversational mode.

Y indicates conversational mode; on this console you must verify all messages selected for message deletion with the cursor, or selector pen or through the CONTROL command.

N indicates nonconversational mode; all messages selected for deletion are automatically deleted.

Note: CON is not valid when UNIT is PRT, or DEVNUM is SYSCONS.

Default: *N*

SEG (nn)
SEG specifies the number of lines in the message area that can be deleted when the CONTROL E, SEG command is entered.

Value Range: *nn* is a decimal value. The minimum is 1; the maximum is 99 or the number of lines in the message area, whichever is smaller.

Note: SEG is not valid when UNIT is PRT, or DEVNUM is SYSCONS.

Default: Default values for SEG are shown in Table 19 on page 259.

DEL {(Y) }
{(R) }
{(RD)}
{(W) }
{(N) }

DEL specifies the message deletion mode of the console.

Y indicates that all messages selected for deletion are deleted when the screen becomes full.

R indicates roll mode. In roll mode, the system deletes a specified number of messages from the screen when an interval elapses. Deletion occurs only if the screen is full and messages are waiting to be displayed.

RD specifies roll mode except for messages awaiting actions. These messages are displayed at the top of the screen.

W indicates wrap mode. In wrap mode, after the screen is filled with messages, new messages overlay the old messages beginning at the top of the screen. A separator line appears following the most recent message on the screen. When a console is in wrap mode, WTORs and action messages will not be retained on the screen.

Note: When wrap mode (DEL(W)) is specified, the areas that were defined by the AREA keyword exist, but are unavailable until the console is put into a non-wrap mode.

N indicates that no automatic message deletion is in effect. Messages must be deleted manually.

Note: DEL is not valid when UNIT is PRT or DEVNUM is SYSCONS.

Default: *RD*

RNUM (nn)
RNUM specifies the maximum number of lines included in one message roll.

Value Range: *nn* is a decimal value. The minimum is 1; the maximum is 99 or the number of lines in the message area, whichever is smaller.

Note: RNUM is not valid when UNIT is PRT or DEVNUM is SYSCONS.

Default: The ***default*** value for all unit types is 5 lines or the number of lines in the message area, whichever is smaller.

RTME (seconds)
RTME specifies the number of seconds between message rolls or wraps.

Value Range: *nnn* is an integer value from 1 to 999 or the fractions 1/2 or 1/4 (for example, RTME(1/2)).

Note:

1. RTME is not valid when UNIT is PRT or DEVNUM is SYSCONS.
2. Do not use the values 1/2 or 1/4 for a 3290 device.

Default: The ***default*** value for all unit types is 2 seconds.

```
MFORM     {(M)              }
          {([J][,S][,T][,X])}
```

MFORM specifies the display format of the messages.

M indicates that the system is to display the message text only. The message display does not include time stamp, job ID, or job name information, or the system name. M is the default.

J specifies that the display is to include the job ID or name.

S specifies that the display is to include the name of the system originating the message.

T specifies that the display is to include a time stamp.

X specifies that the system suppress the job name and system name for JES3 messages issued from the global processor.

Note:

1. The MFORM keyword is ignored if specified with DEVNUM(SYSCONS).
2. You can separate multiple values with a blank or a comma.

Default: M

```
AREA      {(nn[,nn]...)}
          {(NONE)      }
```

AREA specifies the size(s) of the out-of-line display area(s) on the display console. *nn* is a decimal specifying the number of lines in one display area. The first number defines the bottom area of the screen. Subsequent numbers define areas working toward the top of the screen. The minimum number of lines in an area is 4. The maximum number of areas is 11. The sum of the lines in all of the areas must not exceed the screen size. No individual area can be larger than 99. You can separate multiple values with a blank or a comma.

Note: AREA is not valid when UNIT is PRT, or DEVNUM is SYSCONS.

Default: For screen sizes and maximum and default values for AREA, see Table 19 on page 259.

PFKTAB (tablname)

PFKTAB specifies the name of the table that defines the Program Function Key (PFK) definitions.

tablname identifies the 1-8 alphanumeric name for the table and must be the same as one defined in a PFKTABxx member of parmlib. The PFKTABxx member must be active, that is, identified in the PFK parameter on the INIT statement in CONSOLxx.

For more information about PFK tables, see Chapter 75, "PFKTABxx (program function key table definition)," on page 693.

Note: PFKTAB is not valid when UNIT is PRT or DEVNUM is SYSCONS.

Default: If you do not specify a valid PFK table, the system uses the IBM-supplied default PFK definitions that are described in *z/OS MVS System Commands*.

[MONITOR({JOBNAMES[-T]}{,SESS[-T]}{,STATUS})]

MONITOR is an optional keyword that allows you to have the system report on selected events. You can specify one or more parameter options for MONITOR; you can separate multiple values with a blank or a comma:

```
MONITOR(JOBNAMES-T,SESS-T,STATUS)
```

JOBNAMES specifies that the system is to display the names of jobs when they start and end. The system also:

- Displays input/output device allocation at the start of a job step
- Includes the job name in a diagnostic message if the job ends abnormally.

SESS specifies that the system is to display the user ID for each TSO/E session when the session starts and ends. The system also includes the userid in a diagnostic message if a session ends abnormally.

-T specifies that the system is to display the time with the job name or the user ID. Code *-T* with the *JOBNAMES* or the *SESS* parameter. The time is displayed in hh:mm:ss format. When specified, *-T* is activated for all consoles that have MONITOR turned on.

STATUS specifies that the system is to display the names and volume serial numbers of data sets having dispositions of KEEP, CATLG, or UNCATLG whenever these data sets are freed.

Default: If you do not specify the MONITOR keyword (and a valid parameter or combination of parameters), the system does not monitor any job names, TSO/E users, or data sets.

MSCOPE {(sysname|*[,sysname]...)}
{(*ALL)}

MSCOPE allows you to specify those systems in the sysplex from which this console is to receive messages that are not explicitly routed to this console. An asterisk (*) indicates the system on which this CONSOLE statement is defined. You can separate multiple values with a blank or a comma.

**ALL* indicates that unsolicited messages from all systems in the sysplex are to be received by this console.

Default: *ALL is the default for all consoles except the system console, which defaults to *.

CMDSYS {(sysname)}
{(*)}

Indicates the system (in the sysplex) where commands entered on this console are to be sent for processing. An asterisk (*) indicates that commands entered on this console are to be processed on the system where this console is defined.

Note: A value of * is recommended for SMCS consoles since the console is not tied to a system, and the use of z/OS Communication Server Generic Resources may place the console on different systems at different times.

Default: (*) An asterisk within parentheses.

RBUF (nn)

Specifies the number of previously entered commands that can be retrieved on this console by pressing the PA1 key. Set RBUF to a value greater than 1 to allow the operator to retrieve multiple commands, without having to retype each command. For example, to allow the operator to retrieve the last five commands that are entered on this console, specify RBUF(5).

Commands are retrieved as they were entered on the console. That is, commands entered with delimiters (or "stacked commands") are treated as a single command. For example, when RBUF(2) is specified, and the previously entered commands were:

```
'D T'   followed by    'D C;D R;K E,D'
```

- Pressing the PA1 key once retrieves: 'D C;D R;K E,D'
- Pressing the PA1 key a second time retrieves: 'D T'

Commands entered through PF keys cannot be retrieved, unless the keys are in conversational mode (CON=Y).

Note: Any commands entered through PF keys are retrieved only in case of RBUF(1), even if the keys are in non-conversational mode (CON=N).

Value Range: *nn* is a decimal value from 1 to 15. If you specify a value that is not between 1 to 15, an error message is issued and the console defaults to RBUF(15).

Default: 15 (The last 15 commands entered can be retrieved).

AUTOACT (groupname)

AUTOACT specifies the "automatic activate group" for the system console. It is only valid when DEVNUM(SYSCONS) is specified. If AUTOACT is specified, *groupname* is the name of a console group, as defined in CNGRP*xx*.

Value Range: 1 to 8 alphanumeric or national (#, @, $) characters.

While the AUTOACT group is defined and not suspended:

- The system console will automatically be placed into PD mode when all of the consoles in AUTOACT are inactive.
- The system console will automatically be removed from PD mode when any console in the AUTOACT group becomes active.

To suspend AUTOACT processing, issue a VARY CN(*),ACTIVATE or DEACTIVATE command from the system console. This manual intervention will override automatic processing until the opposite command is issued.

SYSTEM (sysname)

Specifies the system you expect the console to be attached to and activated on.

The SYSTEM keyword is mutually exclusive with DEVNUM(SMCS) and DEVNUM(HMCS)..

Default: None

Note: It is possible that a device will not be ready (not turned on) when the system (or sysplex) is being initialized. The device might even be attached to another system as the sysplex is initialized (for example, during an error recovery situation). When you decide to use the device,

first turn it on or reattach it to the proper system, then issue a VARY CN,ONLINE command for the console.

This parameter will primarily be used for consoles which are physically attached to multiple systems and managed by a physical switch. In this case, the SYSTEM parameter determines which system should attempt to activate the console. You cannot define multiple CONSOLE statements with the same DEVNUM and specify a different SYSTEM on each statement. Only the first statement will be processed, and an error message will be issued if additional statements exist.

LU (nnnnnnnn)
This keyword defines the Logical Unit that may only use this console. The LU keyword is optional, but may only be specified with DEVNUM(SMCS). If LU is specified, the console can only be activated at that LU, and no other console can be activated with that LU. For more information, see the*z/OS MVS Planning: Operations* book.

Value Range: *nnnnnnnn* is a character value from 1 to 8 characters. The first character of *nnnnnnnn* must begin with the letters A through Z or with a #,@,$; the remaining characters can be A through Z, 0 through 9, or #,@,or $.

```
LOGON {(REQUIRED)}
      {(OPTIONAL)}
      {(AUTO)}
      {(DEFAULT)}
```

You can use this optional parameter to override the LOGON value that is specified on the DEFAULT statement (if any). The system treats the system console as LOGON(OPTIONAL) no matter what LOGON value is specified on the DEFAULT statement.

IBM suggests that SMCS consoles be LOGON(REQUIRED), either by the system-wide specification on the DEFAULT statement or by the individual CONSOLE statement.

(REQUIRED) specifies that an operator must log on to a console before issuing commands from that console. For MCS consoles, commands may be issued without the operator logging on under the following condition:

- When issuing commands from a console with master authority before a security product is active.

If an operator is not logged on to the console, the system rejects commands issued from that console.

(OPTIONAL) specifies that the operators can optionally log on to the console.

(AUTO) specifies this console is automatically logged on when the console is activated. The user ID will be the console name.

(DEFAULT) specifies that this console will use the LOGON specification on the DEFAULT statement.

Default: DEFAULT, unless the console is a SMCS console that does not specify the LU keyword, in which case the default is REQUIRED.

INTIDS
This optional keyword indicates whether the specified console will receive messages that are directed to console ID zero.

(Y)
The specified console is to receive these messages.

(N)
The specified console is not to receive these messages. This is the default value.

UNKNIDS
This optional keyword indicates whether the specified console will receive messages that are directed to "unknown" console IDs. Messages issued to a now unresolvable 1-byte console ID are delivered to consoles receiving UNKNIDS.

(Y)
The specified console is to receive these messages.

(N)
The specified console is not to receive these messages. This is the default value.

ALLOWCMD
ALLOWCMD specifies when the system console is able to issue commands.

(Y)
The system console is permitted to issue commands even if it is not running in problem determination (PD) mode.

(N)
The system console is restricted from issuing commands until it enters PD mode. This is the default value.

Note:

1. ALLOWCMD is only valid when DEVNUM is SYSCONS.
2. If there are no entries with DEVNUM(SYSCONS), the system uses the default ALLOWCMD(N) as part of the IBM-defined console attributes.

SUPSBY
SUPSBY indicates if the console supports standby mode or not. It is valid only with DEVNUM.

(Y)
The console supports standby mode.

(N)
The system console does not support standby mode. This is the default value.

Syntax and parameters for an INIT statement

```
INIT      MLIM  {(nnnn)}
                {(1500)}

          LOGLIM  {(nnnnnn)}
                  {(1000  )}

          RLIM  (nnnn)

          AMRF  {(Y)}
                {(N)}

          UEXIT  {(Y)}
                 {(N)}

          MPF  {(NO         )}
               {(xx[,xx,...])}

          MMS  {(NO)}
               {(xx)}

          PFK  {(NONE)}
               {(xx)  }

          CNGRP {(xx[,xx,...])}
                {(NO         )}

          MONITOR([SPACE][,DSNAME])

          CMDDELIM  {(c)    }
                    {(X'hh')}

          CTRACE  {(CTnOPSxx)}
                  {(CTIOPS00)}

          ROUTTIME  {(nnn)}
                    {(30 )}

          APPLID  (nnnnnnnn)

          GENERIC  (nnnnnnnn)

          MSGFLD  (xx[,(wrd)])
```

INIT

Specifies that the communications task initialization values follow.

MLIM (nnnn)

MLIM specifies the maximum number of buffers that the system can use to process write-to-operator (WTO) messages. Each buffer is 384 bytes, and is obtained from private storage (subpool 229) in the CONSOLE address space.

Value Range: *nnnn* is a decimal value from 20 to 9999.

Default: 1500 (meaning 1500 buffers).

LOGLIM (nnnnnn)

LOGLIM specifies the maximum number of buffers that the system can use for SYSLOG. Each buffer is 156 bytes, and is obtained from the extended common service area (ECSA).

Value Range: *nnnnnn* is a decimal value from 1000 to 999999.

Default: 1000 (meaning 1000 buffers).

Note:

1. Ensure that there is storage in extended CSA available for the LOGLIM value specified. Otherwise, an error will result.
2. JES3 systems using DLOG require a higher default for the LOGLIM parameter to prevent SYSLOG buffer shortages. IBM suggests setting the LOGLIM value based on the JES3 configuration at your installation at least equal to:

```
LOGLIM = (4000 + (2000 x the-number-of-JES3-locals))
```

RLIM (nnnn)

RLIM specifies the maximum number of buffers that the system can use to process write-to-operator-with-reply (WTOR) messages. Each buffer is 104 bytes, and is obtained from the extended common service area (ECSA).

Value Range: *nnn* is a decimal value from 5 to 9999. It is suggested that you specify a value higher than 99.

Default: The default is 10 (meaning 10 buffers), unless you have set the PLEXCFG= parameter of the IEASYSxx parmlib member to a value other than XCFLOCAL or MONOPLEX. Here, the default is equal to the value of the RMAX parameter on the DEFAULT statement of the CONSOLxx parmlib member of the first system that joined the sysplex.

For more information see the PLEXCFG parameter.

AMRF (Y) | (N)

AMRF specifies whether the action message retention facility is to be active.

Note: If AMRF(N) is specified, then the DOM attribute for the SYSCONS extended MCS console will be set to DOM(ALL) instead of DOM(NORMAL). If AMRF(Y) is specified or defaulted, then the DOM attribute for the SYSCONS extended MCS console will be set to DOM(NORMAL).

For more information about the DOM attribute for extended MCS consoles, see *z/OS MVS Planning: Operations*.

Value Range: *Y* indicates that AMRF is to be active, and *N* indicates that it is not to be active.

Default: *Y* (AMRF is active).

UEXIT {(Y)} | {(N)}

UEXIT specifies whether the WTO/WTOR installation exit IEAVMXIT is to be active. A default IEAVMXIT is no longer shipped in SYS1.LINKLIB.

Value Range: *Y* indicates that IEAVMXIT is to be active and *N* indicates that it is not to be active.

Default: Y (IEAVMXIT is activated if it exists).

MPF {(NO) } | {(xx[,xx,...])}

MPF indicates whether the message processing facility is to be active.

Value Range: *xx* must be two alphanumeric characters indicating the MPFLSTxx member to use. You can specify one or more MPFLSTxx parmlib members. Multiple members are concatenated.

Default: *NO* (the message processing facility is not active).

MMS {(NO)} | (xx)}
MMS indicates whether the MVS message service is to be active.

Value Range: *xx* must be two alphanumeric characters indicating the MMSLSTxx parmlib member to use.

Default: *NO* (the MVS message service is not active).

PFK {(NONE)} | {(xx) }
PFK specifies the PFKTABxx member in parmlib that contains any PFK tables for your consoles.

xx must be two alphanumeric characters, matching the suffix of the PFKTABxx member to be used.

NONE indicates that no PFK tables are defined; the system is to use the IBM-supplied default PFK definitions that are described in *z/OS MVS System Commands*.

Default: *NONE*

CNGRP {(xx[,xx,...])} | {(NO) }
CNGRP indicates whether the system or sysplex is to activate console group definitions.

Value Range: *xx* must be two alphanumeric characters indicating the CNGRPxx parmlib member to use. You can specify a maximum of 38 two-character suffixes.

Default: *NO* is the default; no active console group definitions are to be activated from the system or sysplex.

MONITOR ([SPACE][,DSNAME])
MONITOR specifies whether the system is to add status information to mount messages displayed on consoles.

Value Range:

SPACE indicates that, in mount messages, the system is to display the available space on the direct access volume.

DSNAME indicates that, in mount messages, the system is to display the name of the first non-temporary data set allocated on the volume to which a message refers.

Default: There is no default value for this keyword. That is, if you do not specify a value for MONITOR on the INIT statement, neither the data set name nor the available space is displayed in MOUNT messages.

Note: If a data set has a disposition of delete, its name does not appear in the mount messages.

CMDDELIM {(c)} | {(X'hh')}
CMDDELIM specifies an MCS, SMCS, and HMCS command delimiter that the installation chooses. CMDDELIM allows the operator to enter multiple commands on the console at one time. Each command separated by the delimiter is processed separately. Do not use the command delimiter within a quoted string. If you do, the system considers the delimiter as part of the text and it will not process the delimiter. The order in which the system processes the commands may be different from the order in which the operator entered them. PF keys continue to use a semicolon (;) as the delimiter.

The command delimiter is only honored when it is entered from an MCS, SMCS, or HMCS console. It is not honored when entered from a Subsystem (SS) console. If a command is entered from an extended MCS console, such as a TSO CONSOLE, SDSF LOG, or ULOG functions, the recognition of the command delimiter is dependent on the interface reading the command.

Choose the command delimiter carefully because the delimiter character affects MVS command processing. For example:

- Do not use % or ? as a command delimiter if the SLIP command is to be used with indirect addressing.
- Do not use ; as a command delimiter if the online test executive program (OLTEP) is to be used.
- Do not use & as a command delimiter if the installation uses system symbols.
- Use of &' as a command delimiter will cause the DUMPDS NAME function to fail if the command is issued from an MCS, SMCS, or HMCS console.
- Use of : as a command delimiter will cause the SETIOS TIME function to fail if the command is issued from an MCS, SMCS, or HMCS console.

The command delimiter may be specified in EBCDIC (c) or in hexadecimal (X'hh') format. Valid values are:

EBCDIC	HEXADECIMAL
<	4C
>	6E
&	50
;	5E
%	6C
?	6F
:	7A
"	7F

Default: There is no default value for this keyword. If you do not specify a value for CMDDELIM, your operators cannot enter more than one command at a time.

CTRACE (member-name)
Specifies the member of Parmlib that contains the tracing options for the operations services (OPS) component.

Value Range: The 8-character member name must be in the format CTnOPSxx.

n An alphanumeric character that specifies the source of the member. IBM-supplied members will use "I".

xx Any two characters to identify the member.

Default: CTIOPS00.

For more information about specifying options for the operations services component trace, see *z/OS MVS Diagnosis: Tools and Service Aids*. For information about specifying the component trace member of Parmlib see Chapter 26, "CTncccxx (component trace parameters)," on page 283.

ROUTTIME (nnn)
In a sysplex, ROUTTIME specifies the maximum number of seconds the

ROUTE *ALL, ROUTE *OTHER, or ROUTE *systemgroupname* command waits for responses from each system to the command being routed before aggregating responses. The value specified for ROUTTIME in the INIT statement of the first system in the sysplex applies to all systems which join the sysplex, unless you change the value with a CONTROL M command.

Value Range: *nnn* is a decimal value between 0 and 999. If the value is zero, MVS does not aggregate command responses, but sends separate responses for each command back to the originator of the ROUTE command.

Default: 30

APPLID (nnnnnnnn)
Specifies the z/OS Communications Server APPLID that SMCS is to use on this system. Each system must have a unique APPLID. If an APPLID is not specified, SMCS will not be available for the life of the system.

Value Range: *nnnnnnnn* is a character value from 1 to 8 characters. The first character of *nnnnnnnn* must begin with the letters A through Z or with a #,@,$; the remaining characters can be A through Z, 0 through 9, or #,@,or $.

GENERIC (nnnnnnnn)
Specifies the z/OS Communications Server Generic Resource name that SMCS is to use in this sysplex. The first system that IPLs into the sysplex that specifies GENERIC will set the GENERIC for the life of the sysplex unless modified by the CONTROL M command.

This keyword is optional, but if it is specified, APPLID must also be specified.

Value Range: *nnnnnnnn* is a character value from 1 to 8 characters. The first character of *nnnnnnnn* must begin with the letters A through Z or with a #,@,$; the remaining characters can be A through Z, 0 through 9, or #,@,or $.

Note: The following rules must be true for GENERIC to work:

- z/OS Communications Server must be an APPN node.
- The system must be in a sysplex. It is not valid for a monoplex or a single system.
- A Coupling Facility must be available.
- The z/OS Communications Server ISTGENERIC structure must be defined in the Coupling Facility policy.
- z/OS Communications Server must be able to communicate with the Coupling Facility.

MSGFLD (xx[,(wrd)])
Specifies the MSGFLDxx parmlib member to be used and whether Message Flood Automation is to be automatically enabled when the parmlib member is loaded during system initialization.

xx Either the word NONE or a two-character MSGFLDxx parmlib member suffix composed of the alphanumeric characters 0-9 and A-Z and the national characters @, # and $. Specifying the word NONE indicates that no MSGFLDxx parmlib member is to be loaded. If NONE is specified, the built-in policy will be used if ON is specified for *wrd*.

wrd Either the word ON or OFF. If ON is specified, Message Flood Automation policy is automatically enabled. If OFF is specified, Message Flood Automation policy is not automatically enabled. Specification of ON or OFF is optional.

Default: OFF.

Syntax and parameters of the HARDCOPY statement

```
HARDCOPY   DEVNUM  {(SYSLOG)        }
                   {(OPERLOG)       }
                   {(SYSLOG,OPERLOG)}

           ROUTCODE  {(ALL)                  }
                     {(NONE)                 }
                     {(nn[,nnn-nnn][,nnn]...)}

           CMDLEVEL  {(NOCMDS)}
                     {(INCMDS)}
                     {(STCMDS)}
                     {(CMDS)  }

           HCFORMAT {(CENTURY)}
                    {(YEAR   )}
```

HARDCOPY

HARDCOPY identifies the hardcopy medium, defines the hardcopy message set, defines routing codes for included messages, specifies the kind of messages included in the hardcopy data set, or specifies whether hardcopy records should have a 4-digit year, depending on the options specified.

```
DEVNUM     {(SYSLOG         }
           {(OPERLOG)       }
           {(SYSLOG,OPERLOG)}
```

DEVNUM specifies the output device.

SYSLOG indicates that the system log is to be the hardcopy medium.

OPERLOG indicates that the operations log will be activated and will receive the hardcopy message set. You can specify OPERLOG alone or with SYSLOG. When you specify OPERLOG with SYSLOG, both OPERLOG and SYSLOG will receive the hardcopy message set.

You can specify both subparameters in either order; (OPERLOG,SYSLOG) is equivalent to (SYSLOG,OPERLOG).

Default: The system defaults to SYSLOG as the hardcopy medium in any of the following cases:

- You do not code a HARDCOPY statement.
- You specify OPERLOG alone but the operations log is unusable.

```
ROUTCODE     {(ALL)                  }
             {(NONE)                 }
             {(nnn[,nnn-nnn][nnn]...)}
```

ROUTCODE specifies the routing codes the system is to use to select messages for the hardcopy message set. At system initialization, processing of the HARDCOPY statement sets up a minimum set of routing codes (1,2,3,4,7,8,10, and 42), in addition to any others specified for the hardcopy message set.

ALL indicates that all routing codes (1-128) are used to select messages for the hardcopy message set.

NONE indicates that no routing codes are used to select messages for the hardcopy message set. It will contain the minimum set established at initialization.

nnn is a decimal number from 1 to 128. You can specify a range of routing codes by coding an ascending range of numbers. You can separate multiple values with a blank or a comma.

Default: ALL

```
CMDLEVEL     {(NOCMDS)}
             {(INCMDS)}
             {(STCMDS)}
             {(CMDS)  }
```

CMDLEVEL specifies the types of commands and command responses that are to be included in the hardcopy message set.

NOCMDS specifies that the system is not to include operator or system commands or their responses in the hardcopy message set unless they meet other criteria, such as routing codes. The system ignores this option and assumes *CMDS*, unless **all** of the following conditions are met:

- The system is a member in a sysplex.
- Only one console is active on the system and the console is a non-display console, such as a printer.

INCMDS specifies that the system is to include operator and system commands and their responses, excluding any status displays, in the hardcopy message set.

STCMDS or *CMDS*specifies that the system is to include operator and system commands, their responses, and static status displays in the hardcopy message set. STCMDS and CMDS are equivalent.

Default: *CMDS*

HCFORMAT{(CENTURY|YEAR)}

Specifies the format of the Julian date as it is to appear in the SYSLOG and Master Trace hardcopy records, where:

YEAR

Indicates that the Julian date is to use the format *yyddd*, showing a 2-digit year and 3-digit day.

CENTURY

Indicates that the Julian date is to use the format *yyyyddd*, showing a 4-digit year and 3-digit day.

Note: If you are using CENTURY, examine programs that use SYSLOG to ensure that they can handle the new 4-digit year format.

Default: The default is YEAR.

Syntax and Parameters of the DEFAULT statement

```
DEFAULT    ROUTCODE  {(ALL)                  }
                     {(NONE)                 }
                     {(nnn[,nnn-nnn][,nnn]...)}

           LOGON     {(REQUIRED)}
                     {(OPTIONAL)}
                     {(AUTO)    }

           HOLDMODE  {(YES)}
                     {(NO) }

           RMAX      {(nnnn)}
                     {(99)  }

           SYNCHDEST(groupname)
```

DEFAULT

DEFAULT allows you to define the system defaults used by the Communications Task.

ROUTCODE
{(ALL) }
{(NONE) }
{(nnn[,nnn-nnn],[nnn]...)}

ROUTCODE specifies one or more default routing codes for messages that do not have any routing information. *ALL* specifies 1 through 128 are to be assigned. *NONE* specifies that no routing codes are to be assigned. *nnn* is a decimal number from 1 to 128. You can specify a range of routing codes by coding an ascending range of numbers. You can separate multiple values with a blank or a comma.

Default: If you do not code ROUTCODE, the ***default*** is routing codes 1 through 16.

LOGON
{(REQUIRED)}
{(OPTIONAL)}
{(AUTO) }

LOGON lets you define whether operators must issue LOGON and LOGOFF commands on MCS, SMCS, and HMCS consoles. This definition only applies to consoles that do not specify LOGON on the CONSOLE statement, or specify LOGON(DEFAULT) on the CONSOLE statement. (For information about securing access to system commands, see *z/OS MVS Planning: Operations*.) The system console is always treated as LOGON(OPTIONAL), no matter which value is specified on the DEFAULT statement.

IBM suggests that SMCS consoles be LOGON(REQUIRED), either by the system-wide DEFAULT or by the individual CONSOLE statement.

(REQUIRED) specifies that an operator must log on to an MCS console before issuing commands from that console except under the following condition:

- When issuing commands from the master authority console before a security product is active.

If an operator is not logged on to the console, the system rejects commands that are issued from that console.

(OPTIONAL) specifies that the operators can optionally log on to the MCS, SMCS, or HMCS consoles.

(AUTO) specifies that consoles are automatically logged on when the consoles are activated. The userid is the console name.

Default: OPTIONAL

```
HOLDMODE     {(YES)}
             {(NO) }
```

Specifies whether you want hold mode for the console when the console is roll, roll-deletable, hold mode, or wrap mode. Specifying HOLDMODE(YES) allows an operator to press the ENTER key to suspend or hold messages on the console screen. An operator can enter and exit hold mode by pressing the ENTER key with no input.

Default: NO

```
RMAX     {(nnnn)}
         {(99)  }
```

Specifies the maximum value of a reply ID in the sysplex. RMAX also determines the size of the reply ID displayed in the message text. For example, specifying an RMAX value of 9999 will result in all messages having a 4 character reply ID.

Value Range: 99 - 9999

Note: The value you specify for RMAX can affect the size of the XCF couple data set. For more information, see *z/OS MVS Setting Up a Sysplex*.
MVS issues message IEA403I when forcing the RMAX value to 99.

Default: 99

SYNCHDEST(groupname)

Indicates that synchronous messages (those messages issued using WTO and WTOR macros with SYNCH=YES) should be sent to the first available MCS or HMCS console in the group specified. The console must be attached to this member of the sysplex and must be active when the message is issued. Synchronous messages cannot be sent to another member of the sysplex.

Note: When a synchronous message is issued, **it is extremely important** to respond promptly. The synchronous message is issued as a synchronous WTOR, which will prevent the system from updating its status on the sysplex couple data set. This, in turn, could lead to Sysplex Failure Management (SFM) deciding that the system is not responding normally, and removing it from the sysplex.

Consoles in this console group must be MCS consoles, the system console, or HMCS consoles. SMCS consoles cannot display synchronous messages. The SYNCHDEST console group can include other types of consoles, such as SMCS consoles or extended MCS consoles, but the system will skip these types of consoles when processing synchronous messages. See Chapter 17, "CNGRPxx (specify console groups)," on page 203 for information about specifying console groups.

Value Range: 1 - 8 alphanumeric or special (#,@,$) characters. The reserved console name has special meanings when used as part of this console group:

Console name	Meaning
SYSCON	When used in a group that is specified on the SYNCHDEST keyword, this routes a synchronous message to the system console.

Default: The SYNCHDEST keyword is optional. If you do not specify a SYNCHDEST value, the system routes synchronous messages to the system console.

Devices used as MCS consoles

Table 18 summarizes the devices that can be used as MCS consoles

Table 18. Devices used as MCS consoles

Device	Input/Output Devices	Output Only Devices	Specified As	Description
1403		X		Printer
3203-5		X		Printer
3211		X		Printer
3284-1		X		Printer
3284-2		X		Printer
3286-1		X		Printer
3286-2		X		Printer
3277-2	X	X		Display
3278-2	X	X		Display
3278-2A	X	X		Display
3278-3	X	X		Display
3278-4	X	X		Display
3278-5	X	X	3270-X	Display
3279-2A	X	X		Display
3279-2B	X	X		Display
3279-2C	X	X		Display
3279-3A	X	X		Display
3279-3B	X	X		Display
3279-2X	X	X	3270-X	Display
3279-3X	X	X	3270-X	Display
3279-S2B	X	X	3270-X	Display
3279-S3G	X	X	3270-X	Display
3290	X	X	3270-X	Display

Maximum and default specifications for AREA and SEG

Table 19 lists the maximum and default specifications for AREA and SEG.

Table 19. Maximum and default specifications for AREA and SEG

Unit	Full capability AREA size		Status display AREA size		Screen size	SEG default
	Max	Default	Max	Default		
3270-X	See note 2 on page 260					
3277-2	19	14	23	11,12	24x80	9
3278-2	20	14	24	12,12	24x80	10
3278-2A	16	14	20	10,10	20x80	8
3278-3	28	14	32	16,16	32x80	14
3278-4	39	14	43	21,22	43x80	19
3278-5	24	14	27	13,14	27x132	12

Table 19. Maximum and default specifications for AREA and SEG (continued)

Unit	Full capability AREA size		Status display AREA size		Screen size	SEG default
	Max	Default	Max	Default		
3279-2A	20	14	24	12,12	24x80	10
3279-2B	20	14	24	12,12	24x80	10
3279-2C	16	14	20	10,10	20x80	8
3279-2X	20	14	24	12,12	24x80	10
3279-3A	28	14	32	16,16	32x80	14
3279-3B	28	14	32	16,16	32x80	14
3279-3X	28	14	32	16,16	32x80	14
3279-S2B	20	14	24	12,12	24x80	10
3279-S3G	28	14	32	16,16	32x80	14
3290	20	14	24	12,12	24x80	10
3290	24	14	27	13,14	27x132	12
3290	27	14	31	15,16	31x80	13
3290	28	14	31	15,16	31x160	14
3290	28	14	32	16,16	32x80	14
3290	39	14	43	21,22	43x80	19
3290	46	14	50	25,25	50x106	23
3290	58	14	62	31,31	62x80	29
3290	59	14	62	31,31	62x160	28
SMCS	See note 1					
HMCS	39	14	N/A	N/A	43x80	19

Notes on:

1. SMCS screen size requires a minimum screen size of 24 rows x 80 columns to display the SMCS Console Selection Screen. SMCS consoles that do not have a predefined LU should be at least 24x80.
2. 3270–X and SMCS values are calculated dynamically as follows:

   Full Capability AREA Size

   Max
   #rows - #rows of entry area

   Default
   14 or the Max Area, whichever is smaller.

   Status Display AREA Size

   Max
   #rows

   Default
   2 areas, each is #rows/2. (For odd screen sizes, the B area is one row larger so that it takes up the entire screen.)

   Screen Size
   See SMCS Screen Size Note

SEG Default
(#rows - (3 or 4))/2, where 3 or 4 depends on the number of lines in the entry area.

Message Stream
#rows - 1

Chapter 24. COUPLExx (cross-system coupling facility (XCF) parameters)

The cross-system coupling facility (XCF) provides the MVS coupling services that allow authorized programs on MVS systems in a multisystem environment to communicate (send and receive data) with programs on the same MVS system or other MVS systems. The cross-system extended services (XES) allow authorized programs on MVS to use a coupling facility to share data with other programs on the same MVS system or other MVS systems. The eventual goal for the coupling services provided by XCF and XES is to allow an installation to view multiple MVS systems as one MVS system.

For more information about using XCF, see *z/OS MVS Setting Up a Sysplex*.

You must specify a COUPLExx parmlib member for each system in the sysplex. Use the COUPLExx member of SYS1.PARMLIB to specify the following installation values for this system in your sysplex:

- The sysplex name
- The names of the sysplex couple data sets
- The names of other types of couple data sets
- The failure detection interval
- The operator notification interval
- The cleanup interval
- The default message buffer space
- The default message size for a signalling path
- The transport classes
- The outbound signalling paths
- The inbound signalling paths
- The default retry limit
- The local message buffer space
- XCF CTRACE parmlib member
- XES CTRACE parmlib member
- CPUID toleration indicator for VM guest systems
- Initial status of optional XCF/XES functions

Parameter in IEASYSxx (or supplied by the operator)

```
COUPLE=xx
```

The two-character identifier (xx) is appended to COUPLE to identify the COUPLExx member of parmlib.

Syntax rules for COUPLExx

The following syntax rules apply to COUPLExx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- At least one delimiter (space or comma) is required between a statement and keyword. Delimiters are not required between keywords.
- Use at least one delimiter to separate multiple keyword values within parentheses.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

Syntax format of COUPLExx

```
[COUPLE     SYSPLEX(sysplex-name)]
           [CFRMPOL(CFRMPOLNAME)]
           [CFRMOWNEDCFPROMPT(NO|YES)]
           [PCOUPLE(primary-dsname[,primary-volume])     ]
           [ACOUPLE(alternate-dsname[,alternate-volume]) ]
           [INTERVAL(time-interval)                      ]
           [OPNOTIFY(time-interval)                      ]
           [CLEANUP(cleanup-interval)                    ]
           [MAXMSG(default maxmsg)                       ]
           [RETRY(default retry-limit)                   ]
           [CLASSLEN(default class-length)               ]
           [CTRACE(parmlib-member)                       ]
           [VMCPUIDTOLERATION(NO|YES)                    ]
[FUNCTIONS {ENABLE(function[,function ...]) }
           {DISABLE(function[,function ...])}            ]
[CLASSDEF                                                ]
[           CLASS(class-name)                            ]
[          [CLASSLEN(class-length)           ]           ]
[          [MAXMSG(max-messages)             ]           ]
[          [GROUP(group-name[,group-name]...)]           ]
[PATHIN                                                  ]
[          {DEVICE(device-number[,device-number]...) }   ]
[          {STRNAME(strname[,strname]...)            }   ]
[          [MAXMSG(max-messages)]                        ]
[          [RETRY(retry-limit)  ]                        ]
[PATHOUT                                                 ]
[          {DEVICE(device-number[,device-number]...) }   ]
[          {STRNAME(strname[,strname]...)            }   ]
[          [MAXMSG(max-messages)]                        ]
[          [RETRY(retry-limit)  ]                        ]
[          [CLASS(class-name)]                           ]
[LOCALMSG                                                ]
[           MAXMSG(max-messages)                         ]
[          [CLASS(class-name)]                           ]
[DATA                                                    ]
[           TYPE(name[,name]...)                         ]
[           PCOUPLE(primary-dsname[,primary-volume])     ]
[          [ACOUPLE(alternate-dsname[,alternate-volume])]]
```

IBM-supplied default for COUPLExx

The default parmlib member, COUPLE00, contains the following information, which brings a system up in XCF-local mode because the member does not include the PCOUPLE statement.

```
COUPLE SYSPLEX(LOCAL)
```

Statements/parameters for COUPLExx

COUPLE

Describes the sysplex (systems complex) in which the system participates.

CFRMPOL(CFRMPOLNAME)

Specifies the name of the CFRM policy that is to be started at IPL-time if there is no other previously-activated CFRM policy. If there is a previously-activated CFRM policy, the CFRMPOL specification is ignored. The CFRM couple data set that contains the policy must be accessible to all members of the sysplex. See the PCOUPLE and ACOUPLE statements for a description of the couple data sets.

VALUE RANGE: CFRMPOLNAME is a 1-8 character name of a CFRM policy. Valid characters are A-Z and 0-9. The policy name must start with an alphabetic character.

DEFAULT: None. If you omit the CFRMPOL keyword, and there is no previously-activated CFRM policy, the system does not automatically start a CFRM policy at IPL time.

CFRMOWNEDCFPROMPT

For IPL of a sysplex, allows the installation to request operator prompting to control usage of coupling facilities previously owned by the same sysplex.

When a coupling facility (CF) is first made available to a sysplex, ownership is established by setting the coupling facility authority data with the ownership information (sysplex name, date, and time of day) and saving the ownership information in the CFRM active policy maintained in the CFRM couple data set. A zero value for coupling facility authority data indicates that the coupling facility is not owned and not in use by any sysplex. After a sysplex has established ownership of a coupling facility and the coupling facility remains defined in the CFRM active policy, the sysplex will reestablish ownership without operator prompting when a system gains connectivity after a total loss of connectivity to the coupling facility from all systems in the sysplex.

Operator prompting is accomplished by issuing two message sets, both of which require a YES response to allow the sysplex to use the coupling facility. The message sets are:

1. IXC500I to describe the coupling facility and IXC501A to reply to use the coupling facility.
2. IXC559I to further warn about sysplex usage and IXC560A to confirm usage by the sysplex.

The rules for operator prompting are:

1. CF is not owned (zero value for authority data). Here the CF cannot be in use by any sysplex so CFRM proceeds to gain ownership of the CF without operator prompting.
2. CF is owned (nonzero value for authority data) and the value matches the ownership information in the CFRM active policy. Here it is assumed that the CF was last used by the same sysplex so CFRM proceeds to gain ownership of the CF without operator prompting.

   Note: Specifying or defaulting to NO for CFRMOWNEDCFPROMPT causes the ownership information in the CFRM active policy to remain unchanged when the CFRM couple data set is first used by the sysplex. The saved ownership information can match the nonzero coupling facility authority data.
3. CF is owned (nonzero value for authority data) and the value does not match the ownership information in the CFRM active policy. Here

CFRM prompts the operator and only gains ownership if the operator responds YES to allow sysplex usage of the coupling facility.

Note: Specifying YES for CFRMOWNEDCFPROMPT causes the ownership information in the CFRM active policy to be cleared when the CFRM couple data set is first used by the sysplex. The cleared ownership information will not match the nonzero coupling facility authority data.

VALUE RANGE: one of the following:

NO Specifies that coupling facilities previously owned by the same sysplex can be used by the sysplex without prompting the operator. The CFRM active policy coupling facility ownership information saved when CFRM last gained ownership of coupling facilities for a sysplex is not cleared when the CFRM couple data set is first used by the sysplex. See rule 2 on page 265.

YES
Specifies that coupling facilities previously owned by the same sysplex cannot be used by the sysplex without prompting the operator. The CFRM active policy coupling facility ownership information saved when CFRM last gained ownership of coupling facilities for a sysplex is cleared when the CFRM couple data set is first used by the sysplex. See rule 3 on page 265.

DEFAULT: NO

Note: APAR OA10197 provides the support for CFRMOWNEDCFPROMPT. The APAR is available for systems at z/OS Version 1 Release 4 and higher.

SYSPLEX(sysplex-name)
Specifies the name of the sysplex in which the system participates. The sysplex name is also the substitution text for the &SYSPLEX system symbol. Specify a sysplex name that is different than any of the system names in the sysplex.

The optional SYSPLEX parameter in the LOADxx parmlib member also specifies the sysplex name. LOADxx defines the substitution text for &SYSPLEX early in system initialization so other parmlib members can use it. Therefore, if you plan to use the &SYSPLEX system symbol in parmlib, specify the sysplex name in LOADxx.

When you specify a sysplex name on this parameter, consider the content of LOADxx:

- If the sysplex name is specified in LOADxx, code the &SYSPLEX system symbol on this parameter, as follows:

  ```
  SYSPLEX(&SYSPLEX.)
  ```

  The use of &SYSPLEX ensures that the sysplex name is the *same* in both LOADxx and COUPLExx. **Do not code a name that is different from the name specified in LOADxx.** If you do, the substitution text for &SYSPLEX changes during system initialization, and might generate unpredictable results. See "Step 5. Code support for system symbols in LOADxx" on page 49 for details about how to specify the sysplex name in LOADxx.
- If you specify the sysplex name *only* in COUPLExx, code the name on this parameter. Be aware that the system will not define the substitution text for &SYSPLEX until late in system initialization. The system will

substitute the text LOCAL for &SYSPLEX in all parmlib members that are processed until the COUPLExx parmlib member is completely processed. If you want to resolve the substitution for &SYSPLEX to anything other than LOCAL during the processing of COUPLExx, you need to specify the substitution in LOADxx.

System logger also may use the sysplex name as part of the staging data set name for a DASD-only log stream. The sysplex name specified here allows you to use a digit as the first character, but this is not valid in a staging data set name. If you use a digit as the first character in the sysplex name, system logger substitutes 'STG' for the digit and uses part of the system name in the low level qualifier for the staging data set name. For complete information, see *z/OS MVS Setting Up a Sysplex* and look for the topic on *Naming Conventions for the Log Stream and DASD Log data sets.*

Value range: One of the following:

- A 1- through 8-character sysplex name. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters. The name must match the name specified on the SYSPLEX parameter of the XCF couple data set format utility.

Default: None. A value must be specified for this parameter.

PCOUPLE(primary-dsname[,primary-volume])

Specifies the name of the primary sysplex couple data set and, optionally, the volume on which that data set resides.

Value range: The *primary-dsname* must be a valid data set name of up to 44 characters. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters and periods (.).

The *primary-volume* is the 1 to 6 character name of a mounted and ready DASD volume. Valid characters are alphanumeric and national characters. If the volume is not specified, the data set must be cataloged in the master catalog.

Default: None. If you omit the PCOUPLE keyword the system comes up in XCF-local mode. Also, the system ignores the PATHIN and PATHOUT statements and the ACOUPLE keyword.

ACOUPLE(alternate-dsname[,alternate-volume])

Specifies the name of the alternate sysplex couple data set and, optionally, the volume on which that data set resides. XCF maintains information about the sysplex in both the primary and alternate data sets concurrently. XCF automatically switches to the alternate couple data set when the primary couple data set fails. You cannot specify the ACOUPLE parameter without a PCOUPLE parameter.

Value range: The *alternate-dsname* must be a valid data set name of up to 44 characters. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters and periods (.).

The *alternate-volume* is the 1 to 6 character name of a mounted and ready DASD volume. Valid characters are alphanumeric and national characters. If the volume is not specified, the data set must be cataloged in the master catalog.

Default: None. If you omit the ACOUPLE keyword, XCF is started with only the XCF primary couple data set specified on the PCOUPLE parameter (if it was specified).

INTERVAL(seconds)

Specifies the failure detection interval, the amount of elapsed time at which XCF on another system is to initiate system failure processing for this system because XCF on this system has not updated its status within the specified time.

IBM suggests that the default value (the derived spin failure detection interval) be used. If you specify the INTERVAL value explicitly, it should be no more than twice the derived spin failure detection interval based on the following considerations: the user-specified INTERVAL must be greater or equal to the derived spin failure detection interval so that the system can have a chance to recover before XCF initiates system failure processing; however, if the value is too large, it might allow sympathy sickness to spread throughout the sysplex.

Value range: 3 - 86400 (seconds)

Default: XCF derives the default INTERVAL value from the excessive spin processing parameters in EXSPATxx.

The value is computed as follows:

```
(N+1)*SpinTime + 5
```

where `N` is the number of excessive spin recovery actions, +1 indicates the implicit SPIN action, and `SpinTime` is the excessive spin loop timeout interval. The SPINRCVY statement and the SPINTIME keyword in the EXSPATxx parmlib member determine the number of spin actions and the spin loop timeout, respectively. If the excessive spin parameters are not explicitly set on an EXSPATxx parmlib member, the IBM supplied default spin parameters are used: the derived spin failure detection interval can be 45 seconds or 165 seconds depending on the configuration:

- For systems running in a native or basic mode, or in an LPAR with dedicated CPs, the default spin failure detection interval is 45 seconds.
- For systems running on VM or in an LPAR with shared CPs, the default spin failure detection interval is 165 seconds.

The value is derived from the default EXSPATxx settings:

- The default spin loop timeout value varies by configuration: 10 seconds for systems that have dedicated CPs, or 40 seconds for system that have shared CPs or running under VM.
- The default spin recovery actions are ABEND, TERM, and ACR, and the implicit SPIN.

Note:

1. If USERINTERVAL is set or defaulted to be disabled on the FUNCTIONS statement, the effective failure detection interval is the greater of the spin failure detection interval derived from the EXSPATxx parameters, and the user specified INTERVAL (if any). If USERINTERVAL is enabled, XCF uses the user-specified INTERVAL as the effective failure detection interval.
2. Changes to EXSPATxx parameters cause new spin failure detection interval to be computed, and then cause the effective failure detection interval to change.

OPNOTIFY(seconds)

Specifies the amount of elapsed time (in seconds) at which XCF on another system is to notify the operator (by using message IXC402D) that this system

has not updated its status. The OPNOTIFY value can either be specified as an absolute value (in the form of *nnnnn*) or as a relative value (in the form of *+nnnnn*).

- When an absolute value is specified, the OPNOTIFY value must be greater than or equal to the INTERVAL value if INTERVAL is specified. The effective OPNOTIFY value used by the system is the greater one of the indicated OPNOTIFY value and the effective failure detection interval.
- When a relative value is specified, the effective OPNOTIFY value used by the system is the sum of the effective failure detection interval plus the relative OPNOTIFY value.

Value range:

- For absolute value: 3 to 86400 (seconds)
- For relative value: 0 to 86400 (seconds)

Default: A relative value of +3 is used. Thus the effective OPNOTIFY value used by the system equals three plus the effective failure detection interval.

CLEANUP(seconds)
Specifies how many seconds the system waits between notifying members that this system is terminating and loading a non-restartable wait state. This is the amount of time that XCF group members have to perform cleanup processing.

Value range: 0 - 86400 (seconds)

Default: 15

MAXMSG(nnnnnn)
Specifies a value XCF uses to determine the allotment of message buffers when the MAXMSG parameter is not specified on any one of following:

- The CLASSDEF statement
- The PATHIN statement
- The SETXCF START,CLASSDEF command
- The SETXCF START,PATHIN command.

For more information about determining message buffer space, see *z/OS MVS Setting Up a Sysplex*.

Value range: 1 - 999999 K-byte blocks of storage.

Default: 2000

RETRY(nnn)
Indicates the system default for how much tolerance XCF has for signalling path failures. The lower the value, the sooner XCF stops the signalling path. A higher value indicates that XCF will tolerate more signalling path failures before XCF stops the signalling path. The system uses this value when the RETRY parameter is not on any one of the following:

- The PATHOUT statement
- The PATHIN statement
- The SETXCF START,PATHOUT command
- The SETXCF START,PATHIN command.

Value range: 3 - 255

Default: 10

CLASSLEN(nnnnn)
Specifies the system default for message length that the system uses when CLASSLEN is not specified on either:

- The CLASSDEF statement
- The SETXCF START,CLASSDEF command.

Value range: 0 - 62464

Default: 956 bytes

CTRACE(parmlib-member)
Specifies the member of SYS1.PARMLIB that contains component trace options. You can specify either a CTnXCFxx or a CTnXESxx parmlib member, or both (by specifying the CTRACE keyword more than once).

Value range: The 8-character member name must be in the format CTnXCFxx or CTnXESxx, where:

n An alphanumeric character to specify the source of the member. IBM-supplied members will use "I".

xx Any two characters.

Defaults: CTIXCF00 and CTIXES00

The contents of CTIXCF00 are:

```
TRACEOPTS ON
```

The contents of CTIXES00 are:

```
TRACEOPTS
      ON
```

For more information about specifying options for the XCF component trace, see *z/OS MVS Diagnosis: Tools and Service Aids*. For information about coding the component trace member of SYS1.PARMLIB, see Chapter 26, "CTncccxx (component trace parameters)," on page 283.

VMCPUIDTOLERATION(NO|YES)
Controls the verification of CPU identification information that:

- A VM guest system will perform at SCF initialization with respect to any other active systems in the sysplex
- An LPAR system will perform at XCF initialization with respect to any other active z/VM guest systems in the sysplex

when a simulated external time reference identifier (SIMETRID) is being used.

The keyword has no effect if the system is not running as either a z/VM guest or LPAR system. The keyword also has no effect on system operation subsequent to XCF initialization.

Value range: One of the following:

NO Specifies that a z/VM guest system or LPAR system IPLing into a sysplex will verify its CPU identification information against that of other systems in the sysplex. The system will not tolerate disparate CPU identification information with respect to other systems in the sysplex when SIMETRID is being used.

YES Specifies that a z/VM guest system or LPAR system IPLing into a sysplex will not verify its virtual CPU identification information against that of other systems in the sysplex. It will tolerate disparate CPU identification information with respect to the other systems in the sysplex when SIMETRID is being used.

Also, an LPAR system IPLing into a sysplex will not verify its CPU identification information against the virtual CPU information of any z/VM guest systems active in the sysplex. It will tolerate disparate CPU identification information with respect to z/VM guest systems in the sysplex, but not with respect to other LPAR or native systems in the sysplex when SIMETRID is being used.

VMCPUIDTOLERATION(YES) allows z/VM guest systems participating in a sysplex under z/VM to have disparate CPU identification information, which might be useful in using z/VM to simulate a sysplex comprised of systems on different CECs.

When this option is specified, the system has no way to verify that the z/VM guest systems in the sysplex are in fact guests of the same z/VM host system. The installation must ensure that all systems participating in a sysplex are guests of the same z/VM host system, in order to ensure that the simulated Sysplex Timer support (SIMETRID) provides a common time reference.

Default: NO

FUNCTIONS

Specifies the initial enablement state of XCF/XES optional functions on the current system. You can specify either or both of the two subordinate keywords: ENABLE and DISABLE.

ENABLE(*function***[,***function* **...])**

Enables one or more optional functions.

Value range: The acceptable function names are described in topic "The FUNCTIONS Statement" in *z/OS MVS Setting Up a Sysplex*.

Default: If a given function is not specified on the FUNCTIONS statement, its initial enablement state will be the default state described in topic "The FUNCTIONS Statement" in *z/OS MVS Setting Up a Sysplex*.

DISABLE(*function***[,***function* **...])**

Disables one or more optional functions.

Value range: The acceptable function names are described in topic ""The FUNCTIONS Statement" in *z/OS MVS Setting Up a Sysplex*.

Default: If a given function is not specified on the FUNCTIONS statement, its initial enablement state will be the default state described in topic "The FUNCTIONS Statement" in *z/OS MVS Setting Up a Sysplex*.

CLASSDEF

Each optional CLASSDEF statement defines an XCF transport class (in addition to the DEFAULT transport class) for message traffic. You can define a maximum of 62 different transport classes. If you do not define the DEFAULT transport class, the system will create a default transport class with these characteristics:

- CLASS - The default classname is DEFAULT
- CLASSLEN - The default message length that is specified with the CLASSLEN parameter of the COUPLE statement
- MAXMSG - The amount of message buffer space that is specified with the MAXMSG parameter of the COUPLE statement
- GROUP - Although no groups are explicitly assigned to the default transport class, the DEFAULT transport class handles the messages for any group not explicitly assigned to a specific transport class. The collection of these groups

is referred to by the pseudo-group name UNDESIG. Groups are explicitly assigned to a transport class by using the GROUP parameter on the CLASSDEF statement or the SETXCF command.

CLASS(classname)
Indicates the name of the transport class. Each transport class must have a unique class name on a system. You can specify the same class name on other systems in the sysplex for convenience; but there is no relationship between transport classes on different systems. To modify the definition of the default transport class, use the classname of DEFAULT.

Value range: 1 to 8 alphanumeric characters (A-Z and 0-9) and national characters (@, #, and $).

Default: None. A value must be specified.

CLASSLEN(nnnnn)
Defines the message length (in bytes) for this transport class. Specify a length equal to the length of the longest messages that are most often sent by the group(s) assigned to the transport class. The class length you specify is the message length for which the signalling class will optimize its processing. The class length establishes the size of the message buffers that the signalling service will provide in the transport class.

Value range: 0 - 62464 bytes

Default: The value specified or defaulted on the CLASSLEN parameter of the COUPLE statement.

MAXMSG(nnnnnn)
Specifies the default amount of 1K message buffers allotted for messages sent in this transport class. The system uses this value when MAXMSG is not specified on either of the following:

- The PATHOUT statement
- The SETXCF START,PATHOUT command.

For more information about determining message buffer space, see *z/OS MVS Setting Up a Sysplex*.

Value range: 1 - 999999

Default: The value specified or defaulted on the MAXMSG parameter on the COUPLE statement.

GROUP(groupname,groupname ...)
Specifies one or more groups assigned to this transport class. A group is the set of related members defined to XCF by a multisystem application in which members of the group can communicate between MVS systems with other members of the same group. A group can be assigned to more than one transport class.

By explicitly assigning a group to a transport class, you give the group preferential access to the signalling resources (signalling paths and message buffer space) of the transport class.

Value range: 1 to 8 alphanumeric characters, including @, #, and $.

Default: If the GROUP parameter is omitted, no groups are explicitly assigned to the transport class. In this case, all groups that are not explicitly defined to a transport class (by specifying their groupname on the GROUP parameter on at least one CLASSDEF statement) are implicitly assigned to this transport class. A group that has not been explicitly

assigned to any transport class is called an undesignated group. To explicitly assign the collection of undesignated groups to a transport class, use the pseudo-group name UNDESIG. The pseudo-group name UNDESIG then refers to the collective set of groups that are not explicitly defined to a transport class.

PATHIN
Describes the XCF signalling paths for inbound message traffic. More than one PATHIN statement can be specified. The PATHIN statement is not required for a single system sysplex.

DEVICE(devnum[,devnum ...])
Specifies the device number(s) of a signalling path used to receive messages sent from another system in the sysplex.

Value range: *devnum* is 3 or 4 hexadecimal digits

Default: None. Either this parameter or STRNAME must be specified on the PATHIN statement.

STRNAME(strname[,strname ...])
Specifies the name of one or more coupling facility list structures that are to be used to establish XCF signalling paths.

Either the STRNAME keyword or the DEVICE keyword is required. If you specify the STRNAME keyword, you must provide at least one structure name.

Value range: The *strname* can be 1 to 16 characters, including alphanumeric characters (A-Z and 0-9), national characters (@,#,$), and an underscore (_), where the first character is uppercase alphabetic. XCF signalling structures must begin with the letters IXC.

Default: None. Either this parameter or DEVICE must be specified on the PATHIN statement.

MAXMSG(max-messages)
Specifies the amount of message buffers space (in 1K units) that XCF can use to receive messages over each inbound XCF path defined on the PATHIN statement.

You can specify MAXMSG with either a DEVICE or STRNAME statement.

Value range: 1 - 999999 K byte blocks of storage

Default: The value specified or defaulted on the MAXMSG parameter of the COUPLE statement.

RETRY(nnn)
Specifies how much tolerance XCF has for failures on the inbound paths defined on this PATHIN statement. The lower the value, the sooner XCF stops the path. A higher value indicates that XCF will tolerate more path failures before XCF stops the path.

You can specify RETRY with either a DEVICE or STRNAME statement.

Value range: 3 - 255

Default: The value specified or defaulted on the RETRY parameter of the COUPLE statement.

PATHOUT

Describes the XCF signalling paths for outbound message traffic. More than one PATHOUT statement can be specified. The PATHOUT statement is not required for a single system sysplex.

DEVICE(devnum[,devnum...])

Specifies the device number(s) of a signalling path used to send messages to another system in the sysplex.

Value range: *devnum* is 3 or 4 hexadecimal digits

Default: None. Either this parameter or STRNAME must be specified on the PATHOUT statement.

STRNAME(strname[,strname...])

Specifies the name of one or more coupling facility list structures that are to be used to establish XCF signalling paths.

Either the STRNAME keyword or the DEVICE keyword is required. If you specify the STRNAME keyword, you must provide at least one structure name.

Value range: The *strname* can be 1 to 16 characters, including alphanumeric characters (A-Z and 0-9), national characters (@,#,$), and an underscore (_), where the first character is uppercase alphabetic. XCF signalling structures must begin with the letters IXC.

Default: None. Either this parameter or DEVICE must be specified on the PATHOUT statement.

MAXMSG(max-messages)

Specifies how much buffer space is contributed to the total amount of buffer space XCF can use to send messages over outbound XCF paths defined on the PATHOUT statement. This value represents a portion of the maximum amount of buffer space for the transport class that this path is assigned to.

You can specify MAXMSG with either a DEVICE or STRNAME statement.

Value range: 1 - 999999 K byte blocks of storage. You must specify enough buffer space to contain at least one message whose size is equal to the CLASSLEN value on the CLASSDEF statement for the transport class that this path is defined to.

Default: The value specified or defaulted on the MAXMSG parameter of the CLASSDEF statement for the XCF transport class this path is assigned to.

RETRY(nnn)

Specifies how much tolerance XCF has for failures on the outbound paths defined on this PATHOUT statement. The lower the value, the sooner XCF stops the path. A higher value indicates that XCF will tolerate more path failures before XCF stops the path.

You can specify RETRY with either a DEVICE or STRNAME statement.

Value range: 3 - 255

Default: The value specified or defaulted on the RETRY parameter of the COUPLE statement.

CLASS(classname)
Assigns the outbound XCF path(s) to a transport class. A corresponding CLASSDEF statement must define this class, unless you specify a *classname* of DEFAULT.

You can specify CLASS with either a DEVICE or STRNAME statement.

Value range: 1 to 8 alphanumeric characters, including @, #, and $.

Default: DEFAULT (The classname of the default XCF transport class).

LOCALMSG
Specifies resources for local message traffic between members running on the same system. This statement is optional.

MAXMSG(nnnnnn)
Specifies how much additional buffer space XCF can use for local message traffic beyond the defaults specified on the COUPLE and CLASSDEF statements. The total amount of buffer space used for local messages is this value added to the value specified or defaulted on the MAXMSG parameter on the CLASSDEF statement for the transport class indicated by the CLASS parameter.

Value range: 1 - 999999 K byte blocks of storage. You must specify enough buffer space to contain at least one message whose size is equal to the CLASSLEN value on the CLASSDEF statement for this transport class.

Default: None. This parameter must be specified.

CLASS(classname)
Assigns the local message buffer space to a transport class. A corresponding CLASSDEF statement must define this class, unless you specify a *classname* of DEFAULT.

Value range: 1 to 8 alphanumeric characters, including @, #, and $.

Default: DEFAULT (The classname of the default XCF transport class).

DATA
Defines the use of additional couple data sets. More than one DATA statement is permitted.

TYPE(name,name...)
Specifies the names of the coupling services that are to use the couple data set. Names of the coupling services include:

- ARM for automatic restart management
- CFRM for coupling facility resource management
- LOGR for system logger
- SFM for sysplex failure management
- WLM for workload management
- BPXMCDS for OMVS Sysplex wide mount table

PCOUPLE(primary-dsname[,primary-volume])
Specifies the name of a primary couple data set to support the service, and optionally, the volume on which the data set resides. The PCOUPLE keyword is required with a DATA statement. If the volume is specified, the master catalog is not used to locate the couple data set. If the volume is not specified, the data set must be cataloged in the master catalog.

The PCOUPLE specification is ignored if the service is already operational in the sysplex. If the service is already operational, the system uses the couple data sets for the service that are currently in use by the sysplex.

Value range: The *primary-dsname* must be a valid data set name of up to 44 characters. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters and periods (.). The *primary-volume* is the 1 to 6 character name of a mounted and ready DASD volume. Valid characters are alphanumeric and national characters.

Default: None.

ACOUPLE(alternate-dsname[,alternate-volume])
Specifies the name of the alternate couple data set to support the service, and optionally, the volume on which the data set resides. The ACOUPLE keyword is optional with a DATA statement. If the volume is specified, the master catalog is not used to locate the couple data set. If the volume is not specified, the data set must be cataloged in the master catalog.

The ACOUPLE specification is ignored if the service is already operational in the sysplex. If the service is already operational, the system uses the couple data sets for the service that are currently in use by the sysplex.

Value range: The *alternate-dsname* must be a valid data set name of up to 44 characters. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters and periods (.). The *alternname-volume* is the 1 to 6 character name of a mounted and ready DASD volume. Valid characters are alphanumeric and national characters.

Default: None. If the ACOUPLE keyword is omitted, the system is initialized with only the primary couple data set to support the service name on the TYPE keyword.

Chapter 25. CSVLLAxx (library lookaside (LLA) list)

Use the CSVLLAxx member to specify which libraries (in addition to the LNKLST concatenation) library lookaside (LLA) is to manage. If you do not supply a CSVLLAxx member by the LLA=xx parameter of the LLA procedure on the first starting of LLA for this IPL, or if you specify a parameter of LLA=NONE, LLA will, by default, manage only the libraries that are accessed through the LNKLST concatenation. If you have started LLA successfully with a CSVLLAxx member and then stop LLA, a subsequent start of LLA will use that CSVLLAxx member unless you supply another CSVLLAxx member. If you want to get back to the default settings, specify LLA=NONE.

You can also use CSVLLAxx to specify the following:

- Libraries to be added or removed from LLA management while LLA is active.
- Whether LLA is to hold an enqueue for the libraries it manages.
- Libraries for which LLA is to use the performance-enhancing FREEZE | NOFREEZE option.
- Members of libraries, or whole libraries, to be selectively refreshed in the LLA directory.
- Multiple CSVLLAxx members to be used to control LLA processing. These members can reside in data sets other than SYS1.PARMLIB. Use the PARMSUFFIX(xx) parameter to specify additional data sets. (See *z/OS MVS Initialization and Tuning Guide* for an example.)
- Whether exit routines are to be called during LLA processing.

The operator can change these options dynamically by specifying alternative CSVLLAxx members through the MODIFY LLA,UPDATE=xx command.

In MVS/ESA 4.3 with DFSMS 1.1.0 and SMS active, you can produce a program object, an executable program unit that can be stored in a partitioned data set extended (PDSE) program library. Program objects resemble load modules in function, but have fewer restrictions and are stored in PDSE libraries instead of PDS libraries. LLA can only manage PDSE libraries containing program objects. LLA manages both load and program libraries.

For guidance information about specifying values for CSVLLAxx, see *z/OS MVS Initialization and Tuning Guide*.

Starting LLA

The START LLA,LLA=xx command identifies the CSVLLAxx parmlib member to be used to build the LLA directory. This command is issued by the IBM-supplied IEACMD00 parmlib member during system initialization; the command can also be entered by the operator. For more information about starting LLA, see *z/OS MVS Initialization and Tuning Guide*.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for CSVLLAxx

The syntax for CSVLLAxx is:

- Data, including comments, must be contained in columns 1-71; the system ignores columns 72-80.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- There is no limit to the number of times a keyword can be specified.
- Commas or blanks in any combination constitute a delimiter.
- Delimiters or comments can precede the keywords.
- Delimiters are not required between keywords with value; the right parenthesis after the specified keyword is sufficient.
- Continuation is indicated by placing a comma followed by at least one blank after the last name on a record. Comments are allowed between keywords or values.
- The LLA command will not be processed if there are any errors in the contents of the CSVLLAxx parmlib member(s).

Syntax format of CSVLLAxx

```
LIBRARIES(libname1,libname2,...[-LNKLST-],...) MEMBERS(mmbr1,mmbr2,..)

LNKMEMBERS(mmbr1,mmbr2,...)

REMOVE(libname1,libname2,...[-LNKLST-],...)

GET_LIB_ENQ(YES|NO)

PARMLIB(dsn) SUFFIX(xx)

FREEZE(libname1,libname2,...[-LNKLST-],...)

NOFREEZE(libname1,libname2,...[-LNKLST-],...)

EXIT1(ON|OFF)

EXIT2(ON|OFF)

PARMSUFFIX(xx)
```

IBM-supplied default for CSVLLAxx

None.

Statements/parameters for CSVLLAxx

LIBRARIES(*libname1,libname2,...[-LNKLST-],...***)**

The LIBRARIES statement lists the names of libraries (*libname1,libname2,...*) that are to be added to LLA, if they are not already being managed by LLA, or selectively refreshed, if they are already being managed by LLA.

A library name must be 1 to 44 characters long.

-LNKLST- (the hyphens are coded) can be used to designate the whole LNKLST concatenation. It is a shorthand way to identify all the data sets in the LNKLST instead of listing each of them, that is: (*-LNKLST-,non-LNKLST libname1,non-LNKLST libname2*). In a Dynamic LNKLST environment, *-LNKLST-*

designates a list of data sets compiled from **all** active LNKLST sets. When specified and a new LNKLST set is activated by the Dynamic Lnklst function, any data set in the new LNKLST set that is not currently managed is added to LLA's management. If a new LNKLST data set is the target of a prior REMOVE request, it is not dynamically added to LLA.

Default Value: None

MEMBERS(*mmbr1,mmbr2,...***)**
The MEMBERS statement allows you to refresh dynamically members in the previously named production libraries.

Each MEMBERS statement must be preceded by a LIBRARIES statement that identifies the libraries that contain the new versions of the members.

The member names (*mmbr1,mmbr2...*) must be 1 to 8 characters long.

The LLA directory for each of the listed libraries is updated with the directory entries for the listed members found in each of the data sets. That is, LIBRARIES(LIB.1, LIB.2, LIB.3) MEMBERS(A,B,C) result in the LLA directory for LIB.1 being refreshed with directory entries found for members A, B and C in the DASD directory for LIB.1; the LLA directory for LIB.2 being refreshed with directory entries found for members A, B and C in the DASD directory for LIB.2, and so forth.

If *-LNKLST-* is in the LIBRARIES statement list, LLA will dynamically refresh the specified members in the LNKLST. All occurrences of the member name in the LNKLST concatenation are used to refresh the LLA directory. In a Dynamic LNKLST environment, the specified member is refreshed in **all** active LNKLST sets.

Default Value: None

LNKMEMBERS(*mmbr1,mmbr2,mmbr3,...***)**
The LNKMEMBERS statement lists the member names (*mmbr1,mmbr2,mmbr3...*) that are in the LNKLST concatenation and that are to be selectively refreshed. LNKMEMBERS is equivalent to specifying *LIBRARIES(-LNKLST-) MEMBERS(mmbr1,mmbr2,mmbr3...*).

The member names (*mmbr1,mmbr2,mmbr3...*) must be 1 to 8 characters long.

Default Value: None

REMOVE(*libname1,libname2,...***)**
The REMOVE(*libname1,libname2,...*) statement allows you to remove libraries from the list of libraries managed by LLA. That is, LLA will no longer be used to provide directory entries or staged modules for these libraries.

Specify REMOVE(*-LNKLST-,...*) to remove each of the specified LNKLST libraries from LLA management. Note that REMOVE(*-LNKLST-,...*) does not change the contents of the LNKLST concatenation. In a Dynamic LNKLST environment, this parameter designates a list of data sets compiled from **all** active LNKLST sets.

Libraries should be removed from LLA management before they are compressed. For LNKLST libraries, see “Removing or compressing a data set in an active LNKLST set” on page 705 in Chapter 76, “PROGxx (authorized program list, exits, LNKLST sets and LPA),” on page 697. For non-LNKLST libraries, remove them from LLA management, compress the library, and add the library back to LLA management using CSVLLAxx.

Note: It is not valid to specify the following:

- A library specified on the LIBRARIES statement or on the FREEZE|NOFREEZE option of this member or a member concatenated through use of the PARMLIB(dsn) SUFFIX(xx) statement.
- A data set in the LNKLST if *-LNKLST-* is specified on the LIBRARIES statement of this member or a member concatenated through use of the PARMLIB(dsn) SUFFIX(xx) statement.
- You cannot specify a library to be added to LLA management and designate it to be removed in the same operation.

Note: When REMOVE is specified for a data set that is in a LNKLST, it does change how LLA provides the directory and module for any member on that data set. All data sets in the LNKLST concatenation, except those that were the target of a REMOVE statement, continue to get full LLA management. As the LNKLST concatenation is searched, I/O is done to DASD storage for members on REMOVEd data sets as required. In certain circumstances, this may have a significant performance impact.

Default Value: None

GET_LIB_ENQ(YES|NO)
The GET_LIB_ENQ keyword specifies whether LLA obtains a shared enqueue for the libraries it manages.

If you specify GET_LIB_ENQ (YES), which is the default, LLA obtains a shared enqueue for the libraries it manages. The shared enqueue allows your job to read the libraries, but not to move or erase them. To update these libraries, you must first remove them from LLA management (through the REMOVE keyword).

If you specify GET_LIB_ENQ(NO), LLA does not obtain an enqueue for libraries it manages. Your installation's jobs can update, move, or delete libraries while LLA is managing them. GET_LIB_ENQ(NO) is generally not recommended, however, because of the risks it poses to data integrity. IBM suggests that you use GET_LIB_ENQ (NO) only when necessary.

The system processes the GET_LIB_ENQ keyword only when LLA is started; you cannot set this value while LLA is active. If you attempt to do so (for example, by specifying the GET_LIB_ENQ value on an LLA UPDATE command), the system ignores the GET_LIB_ENQ value without issuing a corresponding message.

If you specify the GET_LIB_ENQ keyword multiple times in a CSVLLAxx member, the system uses only the first occurrence of the keyword and ignores subsequent occurrences without issuing a corresponding message.

Default Value: YES

PARMLIB(*dsn*)
SUFFIX(*xx*)
The PARMLIB (*dsn*) SUFFIX(*xx*) statement allows you to specify an additional CSVLLAxx member to be processed. The system processes this member completely when encountering this statement.

PARMLIB(*dsn*) identifies the data set where the CSVLLAxx member identified by (*xx*) should be found. Note that the PARMLIB keyword allows you to include CSVLLAxx members from data sets other than SYS1.PARMLIB, thereby allowing you to control LLA's specifications without having update access to SYS1.PARMLIB.

Default Value: None. No additional CSVLLAxx members are read.

FREEZE|NOFREEZE(*libname1,libname2,...[-LNKLST-],.....***)**
The FREEZE|NOFREEZE option allows you to specify for a library whether to have LLA use the directory it maintains in its own storage (FREEZE) or to use the directory on DASD storage (NOFREEZE). When a LLA library is specified with FREEZE, the installation takes full advantage of LLA's I/O reduction for directory search for the library and for fetching load modules.

References to members in the data sets of the LNKLST concatenation, when referenced via the appropriate LNKLST DCB, are always treated as FREEZE and cannot be changed by this parameter.

References to members in the data sets of the LNKLST concatenation, when referenced outside the LNKLST (that is, through JOBLIB, STEPLIB, or TASKLIB) are processed as individual libraries. This parameter changes how LLA provides the directory for those references.

-LNKLST- can be used to designate a list of data sets for this parameter, which is derived from **all** active LNKLST sets. When specified, LLA builds this list for you as part of its normal processing. It is a shorthand way to identify all of the data sets in the LNKLST instead of listing each of them using (*-LNKLST-*, non-LNKLST libname1, non-LNKLST libname2). As an example, specifying FREEZE (*-LNKLST-*) directs LLA to set each of the data sets in the LNKLST concatenations list in FREEZE mode. (LLA provides directories from the saved information in LLA storage for requests via JOBLIB, STEPLIB, or TASKLIB.) Specifying NOFREEZE (*-LNKLST-*) directs LLA to manage each of the data sets (in the LNKLST concatenations list) in NOFREEZE mode, whenever the data set is accessed **outside** of the LNKLST. That is, LLA provides directories by doing I/O to the DASD storage for these data sets specified via JOBLIB, STEPLIB, or TASKLIB. However, references to members via any active LNKLST concatenation are always provided from LLA storage, so NOFREEZE does not affect that request.

You can change the FREEZE|NOFREEZE status of a LLA library at any time by using the MODIFY LLA command.

For more information about using the FREEZE|NOFREEZE option, see *z/OS MVS Initialization and Tuning Guide*.

Default Value: By default, LNKLST data sets that are accessed through the LNKLST concatenation are in FREEZE mode and cannot be changed with this parameter. All other libraries (specified by *-LNKLST-* or individually by library name) are set to NOFREEZE by default. To take full advantage of LLA, it is recommended that all libraries be explicitly specified as FREEZE.

EXIT1(ON|OFF)
EXIT1 indicates whether the system is to add the CSVLLIX1 exit routine to the CSVLLIX1 dynamic exit.

The EXIT1 statement is processed only when LLA starts, otherwise it is ignored unless it is not syntactically correct.

When EXIT1(ON) is specified, or if EXIT1 is not specified, in the CSVLLAxx parmlib member used to start LLA, the system attempts to add the CSVLLIX1 exit routine to the CSVLLIX1 dynamic exit when no exit routines are already added by PROGxx or SETPROG processing.

If exit routines are already added by PROGxx or SETPROG processing, or if EXIT1(OFF) is specified, the system does not attempt to add the CSVLLIX1 exit routine. If you require an exit routine to be added, you must explicitly add the exit routine using PROGxx or SETPROG. If one of the already added exit routines is in the active state, LLA calls the CSVLLIX1 exit. To prevent an exit

routine being called after it has been added, you can use the dynamic exit facility to change the state of the exit routine to be inactive.

You can use CSVLLIX1 to:

- Monitor and collect fetch statistics
- Control the 2000 fetch default limit
- Cause staging to happen regardless of statistics for all libraries or even just one library

For more information about CSVLLIX1, see *z/OS MVS Installation Exits*.

Default Value: EXIT1(ON)

EXIT2(ON|OFF)

EXIT2 indicates whether the system is to add the CSVLLIX2 exit routine to the CSVLLIX2 dynamic exit.The EXIT2 statement is processed only when LLA starts, otherwise it is ignored unless it is not syntactically correct.

When EXIT2(ON) is specified, or if EXIT2 is not specified, in the CSVLLAxx parmlib member used to start LLA, the system attempts to add the CSVLLIX2 exit routine to the CSVLLIX2 dynamic exit when no exit routines are already added by PROGxx or SETPROG processing.

If exit routines are already added by PROGxx or SETPROG processing, or if EXIT2(OFF) is specified, the system does not attempt to add the CSVLLIX2 exit routine. If you require an exit routine to be added, you must explicitly add the exit routine using PROGxx or SETPROG. If one of the already added exit routines is in the active state, LLA calls the CSVLLIX2 exit. To prevent an exit routine being called after it has been added, you can use the dynamic exit facility to change the state of the exit routine to be inactive.

You can use CSVLLIX2 to:

- Analyze fetch statistics provided in the LLA module staging parameter list
- Influence the calculation of the LLA value (which determines if a module should be staged) by altering the weighting factors
- Direct that the module must be staged or must not be staged by setting the appropriate return and reason codes.

For more information about CSVLLIX2, see *z/OS MVS Installation Exits*.

Default Value: EXIT2(ON)

PARMSUFFIX(xx)

The PARMSUFFIX statement allows you to specify an additional CSVLLAxx member to be processed. The system processes this member completely when encountering this statement.

This statement is very similar to the PARMLIB(dsn) SUFFIX(xx) statement. The difference is that instead of having to specify a data set name, PARMSUFFIX searches the logical parmlib for the CSVLLAxx member.

Default Value: None. No additional CSVLLAxx members are read.

Chapter 26. CTncccxx (component trace parameters)

Use the CTncccxx parmlib member to specify tracing options for a component trace of either an MVS component or an application. The specific tracing options for those MVS components that support component tracing can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. For tracing options for non-IBM supplied applications, refer to the documentation for those applications.

The CTncccxx parmlib member is specified as an 8-character member in the following format:

CT Indicates that the member contains component trace options

n Specifies the source of the member. IBM-supplied members use "I".

ccc Identifies the component being traced.

xx Any two alphanumeric characters.

The CTncccxx parmlib member is specified on:

- The CTRACE macro (with the DEFINE parameter)
- The TRACE CT operator command
- The parmlib member for some IBM-supplied components.

Tracing of MVS components

The CTncccxx parmlib member is specified in the PARM parameter of the TRACE CT operator command. See *z/OS MVS System Commands* for information about the TRACE CT command.

Tracing of installation-provided applications

The CTncccxx parmlib member is specified in the PARM parameter of the CTRACE macro in the application. See *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information about the CTRACE macro.

The CTncccxx member specified on the CTRACE macro allows an application to establish component tracing options at initialization. A CTncccxx member can specify options for only one trace; if an application has multiple, concurrent traces, called ***sublevel traces***, a CTncccxx member must be specified for each sublevel trace.

Tracing options specified in the CTncccxx parmlib member identified by the CTRACE macro can be overridden by a CTncccxx parmlib member identified on a TRACE CT command. You can use the PARM parameter on a TRACE CT command:

- To specify a CTncccxx member that contains the desired options. This trace will begin immediately.
- To specify a CTncccxx member that contains a PRESET parameter to turn on the trace later when the CTRACE macro is invoked.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for CTncccxx

The following syntax rules apply to CTncccxx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- A comma must be used to separate multiple keyword values within a list.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- The parameters must be in the order shown in the syntax format, to meet the following requirements:
  - The SUB parameter must be followed by a PRESET, ON, OFF, or LIKEHEAD parameter.
  - The SUB parameter must not be before the WTRSTART parameter.

Syntax examples

The following is an example of a CTncccxx member that could be used on a TRACE CT command to preset multiple sublevel traces.

```
TRACEOPTS

SUB(SUB201)
PRESET(DEFINE)
ON
ASID(000E,0002,0003)
JOBNAME(TEST1,TEST2,TEST3,TEST5,SMF,JES2) BUFSIZE(10K)
OPTIONS('new optionlist to record problem data')

SUB(SUB202)
PRESET(DEFINE)
LIKEHEAD

SUB(TEST5NODE1.TEST5NODE2.ASID(12,22,23))
PRESET(DEFINE)
OFF
```

The following example member turns on a trace that writes the trace data to a data set.

```
TRACEOPTS
WTRSTART(CTWTR)
ON
WTR(CTWTR)
ASID(0001,0002,0003)
JOBNAME(TEST1,TEST2,TEST3,SMF,JES2)
BUFSIZE(10K)
OPTIONS('list of options')
```

The following example member stops an external writer for a trace. Before this member is used, the trace must be turned off with a TRACE CT,OFF operator command or with a CTncccxx member that specifies TRACEOPTS and OFF. Do not turn off the writer in the same parmlib member as you turn off the trace.

```
TRACEOPTS
WTRSTOP(CTWTR)
```

The following CTnXESxx member turns on three sublevel SYSXES traces and starts an external writer for each sublevel trace. The last two sublevel traces will start later, when they are defined.

```
TRACEOPTS

ON
  BUFSIZE(4M)

  WTRSTART(XWTRGLO)
  WTRSTART(XWTRCO1)
  WTRSTART(XWTRCO2)

  SUB(GLOBAL)
  ON
  WTR(XWTRGLO)
  OPTIONS('CONNECT,RECOVERY')

  SUB(LT01.ASID(18).CONN1)
  PRESET(DEFINE)
  ON
  WTR(XWTRCO1)
  OPTIONS('GLOBAL,REQUEST')

  SUB(LT01.ASIG(1A).CONN2)
  PRESET(DEFINE)
  ON
  WTR(XWTRCO2)
```

Syntax format of CTncccxx

```
TRACEOPTS
      { WTRSTART(membername){WRAP  }   }
      {                     {NOWRAP}   }

      { SUB(subname)                   }

      { PRESET{(DEFINE)}               }
      {       {(DELETE)}               }

      { ON{ASID(asid)             }    }
      {   {JOBNAME(jobname-list)  }    }
      {   {BUFSIZE(nnnnK|M)       }    }
      {   {OPTIONS(options)       }    }
      {   {WTR{(membername)  }    }    }
      {   {   {(DISCONNECT)  }    }    }

      { OFF                            }

      { LIKEHEAD                       }

TRACEOPTS WTRSTOP(jobname)
```

IBM-supplied default for CTncccxx

IBM supplies defaults for specific components that support component tracing. For more information about default CTncccxx parmlib members, see the following:

- Chapter 13, “BPXPRMxx (z/OS UNIX System Services parameters),” on page 137
- Chapter 24, “COUPLExx (cross-system coupling facility (XCF) parameters),” on page 263
- Chapter 34, “GRSCNFxx (global resource serialization configuration),” on page 345

- Chapter 23, "CONSOLxx (console configuration definition)," on page 231
- SYSLOGR Component Trace in *z/OS MVS Diagnosis: Tools and Service Aids*

Statements/parameters for CTncccxx

To determine if the component to be traced allows the following parameters, see component traces in *z/OS MVS Diagnosis: Tools and Service Aids*.

TRACEOPTS
Specifies the component tracing options for an MVS component or an application.

WTRSTART(membername){WRAP|NOWRAP}
Identifies a member containing source JCL for a started task that the system uses to start the component trace external writer and to open the data sets that the writer uses.

You must also specify the WTR parameter.

WRAP or NOWRAP
If you specify WRAP, when the system reaches the end of the data set or group of data sets, it writes over the oldest data at the start of the data set or the start of the first data set. If you specify NOWRAP, the system stops writing to the data set or sets when the data set or sets are full.

If the WTRSTART parameter on the CTncccxx parmlib member or TRACE CT command specifies NOWRAP, the system uses the primary and secondary extents of the data set or sets. If the WTRSTART parameter specifies WRAP or omits the parameter, the system uses only the primary extent or extents.

Default: WRAP

SUB(subname)
Identifies a sublevel trace for a component or application with multiple traces. The subname is defined by the component or installation-supplied application.

If subname is a head level, all of the head's sublevel traces that are defined with a LIKEHEAD=YES parameter inherit the options specified in this parmlib member. Therefore, the options you specify for a head level can affect many sublevel traces.

If you specified the SUB parameter on the parmlib member to activate these trace options, the operator should not specify the SUB parameter on the TRACE CT command.

SUB(subname) cannot be specified in a parmlib member activated during system initialization.

The SUB parameter must be followed by a PRESET, ON, OFF, or LIKEHEAD parameter.

Value Range: Specify the sublevel trace as it was defined through the CTRACE macro as:

- A name
- An ASID
- A job name.

Default: None.

PRESET(DEFINE | DELETE)

PRESET(DEFINE) specifies that the component trace options established in the CTncccxx member are to be remembered until the component trace is defined. When a CTRACE macro later defines the trace, the preset information is used to set the options.

Use the PRESET parameter to set up the trace options in anticipation of a job entering the system or an address space being run. Later, when the CTRACE macro is invoked in the job or address space, the system will override the tracing options set up at initialization and instead use the options in the CTncccxx parmlib member with the PRESET.

The PRESET parameter is valid only in a parmlib member specified on the TRACE CT command; the PRESET parameter is ignored if specified from the CTRACE macro.

To use the preset options, a sublevel trace (SUB) can be defined as either ASID or JOBNAME. If there are both ASID and JOBNAME presets, the system uses only the ASID preset.

Once the information established through the preset has been used for a trace, the preset no longer exists.

If you specified a valid preset and then a CTRACE macro is issued with a PARM option, the system ignores the information in the parmlib member specified by the CTRACE macro and, instead, uses the preset information. But, if your preset was not valid, the system will issue an error message and then proceed to process the trace options specified on the CTRACE macro; the preset information is then deleted.

You can change the preset options by using the TRACE CT command with the PARM parameter and specifying PRESET(DEFINE) in the parmlib member.

PRESET(DELETE) deletes the existing preset.

ON

If the component trace is currently off, the parmlib member turns the trace on. If the component trace is currently on and can be changed, the parmlib member changes the trace. An installation-supplied application trace can be changed if the CTRACE macro that defined the trace contained a MOD=YES parameter.

When a head trace that was defined with HEADOPTS=YES is turned on, all sublevel traces currently defined as LIKEHEAD are also turned on. An installation-supplied application trace can also have head and sublevel traces, if specified in the CTRACE macro that defined the trace.

Whenever a trace that has sublevel traces is changed, all sublevel traces currently in the LIKEHEAD state will also be changed. Therefore, a change may cascade down a number of levels.

A head trace may have been defined so that it is not allowed to be changed (HEADOPTS=NO on the CTRACE macro). If this is the case, the trace is really just a place holder for options for other traces.

ASID(asid-list)

Specifies the address space identifiers (ASIDs) of address spaces to be used as a filter for tracing. Events in these ASIDs are to be recorded by the component trace.

Value Range: A list of 0 to 16 hexadecimal ASIDs separated by commas. An empty ASID list, ASID(), turns off filtering by address spaces. In the ASID parameter, list all address spaces to be traced; address spaces specified for previous traces will not be traced unless listed.

JOBNAME(jobname-list)
Specifies the names of jobs to be used as a filter for tracing. Events in these jobs are to be recorded by the component trace.

Value Range: A list of 1 to 16 job names separated by commas. An empty job list, JOBNAME(), turns off filtering by jobs. In the JOBNAME parameter, list all jobs to be traced; jobs specified for previous traces will not be traced unless listed.

BUFSIZE(nnnnK | nnnnM)
Specifies the size, in kilobytes (K) or megabytes (M), of the trace buffer you want the system to use.

Value Range:

nnnnK
The buffer size in kilobytes, where *nnnn* is a decimal number from 1 to 9999.

nnnnM
The buffer size in megabytes, where *nnnn* is a decimal number from 1 to 2047.

When the size is not specified, the system uses the component-defined default or, for some components, the size specified in a TRACE CT command.

The size specified for an installation-supplied application trace must be within the range specified on the CTRACE macro for the trace; see the programmer for the size value.

OPTIONS
Specifies component-specific options for tracing.

For the component options that support component tracing, see the topic on component trace in *z/OS MVS Diagnosis: Tools and Service Aids*.

Value Range: A list of component-specific options with each option enclosed in quotation marks and the options separated by commas. For example:

```
OPTIONS('optionA','optionB','optionN')
```

The options for some IBM-supplied component traces can be changed while the trace is running; to change the options for others, stop the trace and restart it with the new options. An installation-supplied application trace, defined with MOD=YES in the CTRACE macro, can be changed while running.

The options for a head level defined with HEADOPTS=NO cannot be changed. When a head level that was defined with HEADOPTS=YES is changed, all of the sublevel traces currently in LIKEHEAD status will also be changed. Therefore, a change may cascade down a number of levels.

Omit OPTIONS to allow the component to use its default options.

WTR(membername|DISCONNECT)
Connects or disconnects the component trace external writer and the trace. The member name identifies the member that contains the source JCL that invokes the external writer. The membername in the WTR parameter must match the membername in the WTRSTART parameter.

WTR(DISCONNECT) disconnects the writer and the trace. The component continues tracing and placing the trace records in the address-space buffer, but stops passing trace records to the external writer.

You must also specify a WTRSTART or WTRSTOP parameter to start or stop the writer.

OFF
The system is to stop tracing for the component. If the component is connected to a component trace external writer, the system forces an implicit disconnect.

Some components do not stop tracing completely but reduce the tracing activity to the minimum required for serviceability data in a dump. An installation-supplied application trace will reduce tracing to a minimum if the head level is defined with MINOPS. All sublevel traces with the LIKEHEAD attribute will also reduce tracing to the minimum. Component trace writes a message to the operator when tracing is reduced to the minimum.

A head level defined with HEADOPTS=NO cannot be turned off.

When a head level defined with HEADOPTS=YES is turned off, all sublevel traces currently defined as LIKEHEAD will also be turned off. An installation-supplied application trace can have head level and sublevel traces, if specified in the CTRACE macro that defined the trace.

LIKEHEAD
Specifies that the trace specified in the SUB parameter is to use the options that are defined to that sublevel trace's head level.

If you use the TRACE CT command to change a trace that is currently ON or OFF to LIKEHEAD, be aware of the following:

- The attributes of the sublevel trace must match the head level trace.
- The sublevel trace inherits the state (ON or OFF) and the options (ASID, JOBNAME, OPTIONS, and BUFSIZE) of the head level.
- The head level must have been defined with HEADOPTS=YES.

Component traces are defined by the IBM-supplied component; for components that use heads and subheads, see information about the component trace in *z/OS MVS Diagnosis: Tools and Service Aids*. Application traces are defined by the application using the CTRACE macro; for more information, see the documentation for the application.

WTRSTOP(jobname)
Identifies the name of the job for a currently running component trace external writer that the system is to stop. The system also closes the data sets the writer used. The jobname is either:

- Member name, if the source JCL is a procedure
- Job name, if provided on a JOB statement within the source JCL

You must also specify the WTR parameter. Otherwise, traces connected to the writer remain connected after the system stops it.

Chapter 27. CUNUNIxx (Unicode Services environment)

CUNUNIxx contains information that Unicode services uses to define or change its environment. Through the use of the different statements, you can modify the environment in the following ways:

- Add, delete, or replace conversion tables, case support, normalize support, collation support and locale support to the Unicode environment.
- Specify the upper storage limit for pages to be used by conversion images.
- Identify the name of the conversion image to be selected.
- Delete an inactive conversion environment.

Selecting a CUNUNIxx member

Select a CUNUNIxx parmlib member in one of the following ways:

- Specify the UNI keyword in IEASYSxx.
- Issue the SET UNI command after system initialization.

Two alphanumeric characters are appended to CUNUNI to form the name of the CUNUNIxx parmlib member. After initialization, you can issue the SET UNI command to change the CUNUNIxx member; however, this change is temporary. At the next IPL, the system uses the CUNUNIxx member specified in IEASYSxx. For information about IEASYSxx, see Chapter 54, "IEASYSxx (system parameter list)," on page 431. For descriptions of the SET command, see *z/OS MVS System Commands*.

Parameter in IEASYSxx

```
UNI= {xx}
```

The two character identifier (xx) is appended to CUNUNI to identify the CUNUNIxx member of SYS1.PARMLIB.

Syntax rules for CUNUNIxx

These rules apply to the creation of CUNUNIxx:

- Use columns 1–71 to code statements.
- Comments must begin with "/*" and end with "*/".
- Comments can span statements.
- The keywords DELETE and IMAGE cannot be specified in the same member.

Syntax format of CUNUNIxx

```
ADD FROM(xxxxx) TO(yyyyy)[TECHNIQUE|TECH(zzzzzzzz)]
                [PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

ADD CASE([LOCAL|SPECIAL|NORMAL])[UNIVER(univer)][PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

ADD NORMALIZE|NORM([UNI301|UNI320|UNI401|UNI410|UNI600])[PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

ADD COLLATE|COLL([UCAver])[PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]
                [[LOCALE(locale)[DSNAME(dsname)][VOLUME|VOL(volser)]] |
                [COLRULES(colrules)[DSNAME(dsname)][VOLUME|VOL(volser)]]]

ADD IMAGE=zzzzzzzz[PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

ADD STRPROFILE=NAME[PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

ADD BLDLOCALE|BLDLOC=locname
                [,CCSID(nnnnn)][TECHNIQUE|TECH(zzzzzzzz)][PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

DELETE FROM(xxxxx) TO(yyyyy)[TECHNIQUE|TECH(zzzzzzzz)]FORCE(YES)

DELETE CASE([LOCAL|SPECIAL|NORMAL])[,UNIVER(univer)] FORCE(YES)

DELETE NORMALIZE|NORM([UNI301|UNI320|UNI401|UNI410|UNI600]) FORCE(YES)

DELETE COLLATE|COLL([UCAver])[[LOCALE(locale)|[COLRULES(colrules)],FORCE(YES)

DELETE STRPROFILE=NAME[DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

DELETE BLDLOCALE|BLDLOC=locname[,CCSID(nnnnn)][TECHNIQUE|TECH(zzzzzzzz)]

REPLACE FROM(xxxxx) TO(yyyyy)[TECHNIQUE|TECH(zzzzzzzz)][PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)][FREE(NO|(YES,FORCE)]

REPLACE CASE([LOCAL|SPECIAL|NORMAL])[,UNIVER(univer)][PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)][FREE(NO|(YES,FORCE)]

REPLACE NORMALIZE|NORM([UNI301|UNI320|UNI401|UNI410|UNI600])[PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)][FREE(NO|(YES,FORCE)]

REPLACE COLLATE | COLL([UCAver])[PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]
                [[LOCALE(locale)[DSNAME(dsname)][VOLUME|VOL(volser)]] |
                [COLRULES(colrules)[DSNAME(dsname)][VOLUME|VOL(volser)]]]
                [FREE(NO|(YES,FORCE)]

REPLACE STRPROFILE=NAME[PAGEFIX(YES|NO)][DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

REPLACE BLDLOCALE|BLDLOC=locname[,CCSID(nnnnn)]
                [TECHNIQUE|TECH(zzzzzzzz)][PAGEFIX(YES|NO)]
                [DSNAME|DSN(dsname)][VOLSER|VOL(volser)]

REALSTORAGE[(]nnnnnn[K|M|G|][)][;]

DELETE INACTIVE[;]

DELETE ALL,FORCE(YES)
```

IBM-supplied default for CUNUNIxx

None.

Statements/parameters for CUNUNIxx

ADD FROM(*xxxxx***) TO(***yyyyy***) [TECHNIQUE | TECH(***zzzzzzzz***)]**
[PAGEFIX(YES|NO)][DSNAME | DSN(*dsname***)][VOLSER | VOL(***volser* **)]**

Adds specific tables to the Unicode environment.

xxxxx
Specifies the source CCSID of the character conversion tables to be added. *xxxxx* is a 1- through 5-character name that identifies the tables.

yyyyy
Specifies the target CCSID of the character conversion tables to be added. *yyyyy* is a 1- through 5-character name that identifies the tables.

zzzzzzzz
Specifies the technique search order for the character conversion tables to be removed. *zzzzzzzz* is a 1- through 8-character alphanumeric field. Possible values are one or more of the following:

- R - Roundtrip conversion
- E - Enforced subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

If no technique search order is specified, the default is RECLM.

Note: You may get different results for different orders of the techniques placed in the *zzzzzzzz* field.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

ADD CASE([LOCAL | SPECIAL |
NORMAL])[UNIVER(*univer***)][PAGEFIX(YES|NO)][DSNAME | DSN(***dsname***)][VOLSER |**
VOL(*volser* **)]**

Adds the character case conversion tables to the Unicode environment. Local, Special, and Normal are optional and can be defined in the same statement only once.

univer
Specifies the Unicode standard version to be loaded. Valid values are:

- UNI300
- UNI301
- UNI320
- UNI401
- UNI410

- UNI500
- UNI600

dsname
Specifies the name of the data set that contains the specific tables. The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

ADD NORMALIZE | NORM([*normver***])[PAGEFIX(YES|NO)][DSNAME | DSN(***dsname***)][VOLSER | VOL(***volser***)]**
Adds the normalization tables to the Unicode environment.

normver
Specifies the Unicode standard table version to be loaded. Possible values are one of the following:

- UNI301
- UNI320
- UNI401
- UNI410
- UNI600

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

ADD COLLATE | COLL([*UCAver***])[PAGEFIX(YES|NO)][DSNAME | DSN(***dsname***)] [VOLSER | VOL(***volser***)] [[LOCALE(***locale***)[DSNAME(***dsname***)] [VOLUME|VOL(***volser***)]] | [COLRULES(***colrules***)[DSNAME(***dsname***)] [VOLUME|VOL(***volser***)]]]**
Adds the collation tables to the Unicode environment.

UCAver
Specifies the Unicode Collation Algorithm (UCA) versions. Possible values are one or more of the following:

- UCA301
- UCA400R1
- UCA410
- UCA600

dsname
Specifies the name of the data set that contains the specific tables. The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

locale
Specifies the local member name where collation rules are to be loaded.

colrules
Specifies the User Collation Rules (UCR) member name where collation rules are to be loaded.

ADD IMAGE=*zzzzzzzz*[PAGEFIX(YES|NO)][DSNAME | DSN(*dsname*)][VOLSER | VOL(*volser*)]
Adds an image to the Unicode environment, where the image is a member of the parmlib concatenation. If the image specified already exists in storage, the tables are not added.

zzzzzzzz
Specifies the name of the conversion image to be added. The image member specified must be present in SYS1.PARMLIB or in another data set in the logical parmlib concatenation. zzzzzzzz is an eight-character alphanumeric field.

Value Range: Any valid z/OS member name.

When an image is loaded with the ADD,IMAGE statement, the existing tables in the Unicode environment are not replaced; only those tables that are not currently available in the Unicode environment are loaded from the image.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

ADD STRPROFILE=NAME[,PAGEFIX(YES|NO)][,DSNAME | DSN(*dsname*)] [,VOLSER | VOL(*volser*)]
Adds the profile to the Unicode environment.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements.

If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

ADD BLDLOCALE|BLDLOC=locname[,CCSID(nnnnn)][,TECHNIQUE|TECH(zzzzzzzzz)] [,PAGEFIX(YES|No)][,DSNAME|DSN(dsname)][,VOLSER|VOL(volser)]]
Adds locales to the Unicode environment.

locname
Specifies the locale name to be added. The locale name is case sensitive and is converted to uppercase unless it is specified inside single quotes.

nnnnn
Specifies the CCSID of the locale to be added.

zzzzzzzz
Specifies the technique search order for the character conversion table to be added. *zzzzzzzz* is an eight-character alphanumeric field. Possible values are one or more of the following:

- R - Roundtrip conversion
- E - Enforced Subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

Note: You may get different results for different orders of the techniques placed in the *zzzzzzzz* field.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must have similar characteristics as the SYS1.SCUNTBL data set provided. The size of the data set can be unequal.

If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from one- to six-characters.

DELETE FROM(*xxxxx***) TO(***yyyyy***) [TECHNIQUE | TECH(***zzzzzzzz***)] FORCE(YES)**
Remove specific tables from the Unicode environment.

xxxxx
Specifies the source CCSID of the character conversion tables to be removed. *xxxxx* is a 1- through 5-character name that identifies the tables.

yyyyy
Specifies the target CCSID of the character conversion tables to be removed. *yyyyy* is a 1- through 5-character name that identifies the tables.

zzzzzzzz
Specifies the technique search order for the character conversion tables to be removed. *zzzzzzzz* is a 1- through 8-character alphanumeric field. Possible values are one or more of the following:

- R - Roundtrip conversion
- E - Enforced subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

If no technique search order is specified, the default is RECLM.

v **Note:** You may get different results for different orders of the techniques
v placed in the *zzzzzzzz* field.

FORCE(YES)
Specifies that the system does not check whether applications are currently using the tables. The storage occupied by the tables is returned to the system. FORCE(YES) is a required parameter.

DELETE CASE([LOCAL | SPECIAL | NORMAL])[UNIVER(*univer*)] FORCE(YES)
Removes the character case conversion tables from the Unicode environment. LOCAL, SPECIAL, and NORMAL are optional and can be defined in the same statement only once.

univer
Specifies the Unicode standard version to be loaded. Valid values are:
- UNI300
- UNI301
- UNI320
- UNI401
- UNI410
- UNI500
- UNI600

FORCE(YES)
Specifies that the system does not check whether applications are currently using the tables. The storage occupied by the tables is returned to the system. FORCE(YES) is a required parameter.

DELETE NORMALIZE | NORM([*normver*]) FORCE(YES)
Removes the normalization tables from the Unicode environment.

normver
Specifies the Unicode standard table version to be deleted. Possible values are one of the following:
- UNI301
- UNI320
- UNI401
- UNI410
- | UNI600

FORCE(YES)
Specifies that the system does not check whether applications are currently using the tables. The storage occupied by the tables is returned to the system. FORCE(YES) is a required parameter.

DELETE COLLATE | COLL([*UCAver*])[[LOCALE(*locale*)] | [COLRULES(*colrules*)]], FORCE(YES)
Removes the collation tables to the Unicode environment.

UCAver
Specifies the Unicode Collation Algorithm (UCA) versions . Possible values are one or more of the following:
- UCA301
- UCA400R1
- UCA410

- UCA600

locale
Specifies the local member name where collation rules are to be loaded.

colrules
Specifies the User Collation Rules (UCR) member name where collation rules are to be loaded.

FORCE(YES)
Specifies that the system does not check whether applications are currently using the tables. The storage occupied by the tables is returned to the system. FORCE(YES) is a required parameter.

DELETE STRPROFILE=NAME[,DSNAME|DSN(*dsname***)][,VOLSER | VOL(***volser***)]**
Removes the profile from the Unicode environment.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

DELETE BLDLOCALE|BLDLOC=locname[,CCSID(nnnnn)][,TECHNIQUE|TECH(zzzzzzzzz)]
Deletes locales from the Unicode environment.

locname
Specifies the locale name to be deleted.

nnnnn
Specifies the CCSID of the locale to be deleted.

zzzzzzzz
Specifies the technique search order for the character conversion table to be deleted. *zzzzzzzz* is an eight-character alphanumeric field. Possible values are one or more of the following:

- R - Roundtrip conversion
- E - Enforced Subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

Note: You may get different results for different orders of the techniques placed in the *zzzzzzzz* field.

REPLACE FROM(*xxxxx***) TO(***yyyyy***) [TECHNIQUE | TECH(***zzzzzzzz***)] [PAGEFIX(YES|NO)][DSNAME | DSN(***dsname***)] [VOLSER | VOL(***volser* **)] [FREE(NO | (YES,FORCE))]**
Replaces specific tables that might be currently in the Unicode environment. If a table to be replaced is not in storage, the system adds the table.

xxxxx
Specifies the source CCSID of the character conversion tables to be replaced. *xxxxx* is a 1- through 5-character name that identifies the tables.

yyyyy
Specifies the target CCSID of the character conversion tables to be replaced. *yyyyy* is a 1- through 5-character name that identifies the tables.

zzzzzzzz
Specifies the technique search order for the character conversion tables to be removed. *zzzzzzzz* is a 1- through 8-character alphanumeric field. Possible values are one or more of the following:
- R - Roundtrip conversion
- E - Enforced subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

If no technique search order is specified, the default is RECLM.

Note: You may get different results for different orders of the techniques placed in the *zzzzzzzz* field.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

FREE
Specifies whether the storage associated with the table is to be released.

NO Unicode will not release the storage associated with the table. When FREE is not specified, the default is **NO**.

(YES,FORCE)
Release the storage associated with the table. The system does not check whether applications are currently using the tables.

REPLACE CASE([LOCAL | SPECIAL | NORMAL])[UNIVER(*univer*)][PAGEFIX(YES|NO)][DSNAME | DSN(*dsname*)][VOLSER | VOL(*volser*)][FREE(NO | (YES,FORCE))]
Replaces the character case conversion tables currently in the Unicode environment. LOCAL, SPECIAL, and NORMAL are optional and can be defined in the same statement only once.

univer
Specifies the Unicode standard version to be loaded. Valid values are:
- UNI300
- UNI301
- UNI320

- UNI401
- UNI410
- UNI500
- UNI600

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

FREE
Specifies whether the storage associated with the table is to be released.

NO Unicode will not release the storage associated with the table. When FREE is not specified, the default is **NO**.

(YES,FORCE)
Release the storage associated with the table. The system does not check whether applications are currently using the tables.

REPLACE NORMALIZE | NORM([*normver***])[PAGEFIX(YES|NO)][DSNAME | DSN(***dsname***)][VOLSER | VOL(***volser***)][FREE(NO | (YES,FORCE))]**
Replaces the normalization tables currently in the Unicode environment.

normver
Specifies the Unicode standard table version to be replaced. Possible values are one of the following:
- UNI301
- UNI320
- UNI401
- UNI410
- UNI600

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements.

If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

FREE
Specifies whether the storage associated with the table is to be released.

NO Unicode will not release the storage associated with the table. When FREE is not specified, the default is **NO**.

(YES,FORCE)
Release the storage associated with the table. The system does not check whether applications are currently using the tables.

REPLACE COLLATE | COLL([*UCAver*])[PAGEFIX(YES|NO)][DSNAME | DSN(*dsname*)][VOLSER | VOL(*volser*)][[LOCALE(*locale*)[DSNAME(*dsname*)] [VOLUME|VOL(*volser*)]] | [COLRULES(*colrules*)[DSNAME(*dsname*)] [VOLUME|VOL(*volser*)]]][FREE(NO | (YES,FORCE))]

Replaces the collation tables currently in the Unicode environment.

UCAver
: Specifies the Unicode Collation Algorithm (UCA) versions. Possible values are one or more of the following:
  - UCA301
  - UCA400R1
  - UCA410
  - UCA600

dsname
: Specifies the name of the data set that contains the specific tables.

  The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
: Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

locale
: Specifies the local member name where collation rules are to be loaded.

colrules
: Specifies the User Collation Rules (UCR) member name where collation rules are to be loaded.

FREE
: Specifies whether the storage associated with the table is to be released.

  NO Unicode will not release the storage associated with the table. When FREE is not specified, the default is **NO**.

  (YES,FORCE)
  Release the storage associated with the table. The system does not check whether applications are currently using the tables.

REPLACE STRPROFILE=NAME[,PAGEFIX(YES|NO)][,DSNAME|DSN(*dsname*)] [,VOLSER | VOL(*volser*)]

Replaces the current profile in the Unicode environment.

dsname
: Specifies the name of the data set that contains the specific tables.

  The specified *dsname* must contain the same features as the official tables repository, for example, the SYS1.SCUNTBL. The size of the data set can be decided according to users' requirements. If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
: Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from 1- through 6-characters.

REPLACE BLDLOCALE|BLDLOC=locname[,CCSID(nnnnn)][,TECHNIQUE|TECH(zzzzzzzzz)] [,PAGEFIX(YES|No)][,DSNAME|DSN(dsname)][,VOLSER|VOL(volser)]]

Replaces locales in the Unicode environment.

locname
Specifies the locale name to be replaced.

nnnnn
Specifies the CCSID of the locale to be replaced.

zzzzzzzz
Specifies the technique search order for the character conversion table to be replaced. *zzzzzzzz* is an eight-character alphanumeric field. Possible values are one or more of the following:

- R - Roundtrip conversion
- E - Enforced Subset conversion
- C - Customized conversion
- L - Language Environment-Behavior conversion
- M - Modified for special use conversion
- 0-9 - User-defined conversions

See *z/OS Unicode Services User's Guide and Reference* for additional information on techniques.

Note: You may get different results for different orders of the techniques placed in the *zzzzzzzz* field.

dsname
Specifies the name of the data set that contains the specific tables.

The specified *dsname* must have similar characteristics as the SYS1.SCUNTBL data set provided. The size of the data set can be unequal.

If no *dsname* is specified, SYS1.SCUNTBL is used as the default.

volser
Specifies the volume serial number of the device on which the tables are to be loaded. *volser* can be from one- to six-characters.

REALSTORAGE *nnnnnn*[**K | M | G**] [**;**]
Defines the upper storage limit, in pages, to be used by the conversion environment. For information about the amount of storage required for a conversion environment, see *z/OS Unicode Services User's Guide and Reference* .

Value Range: 0 to 524287.

Example:

```
REALSTORAGE 0;      /* no explicit limit */
REALSTORAGE 12800;  /* 50 MB limit */
```

Note:

1. The request to load a new conversion environment is rejected when the value of the REALSTORAGE keyword is lower than the amount of storage needed.
2. The selection of '0' results in no limit.
3. The REALSTORAGE keyword is an optional parameter.

DELETE mode
Deletes partially or completely the Unicode environment.

Value Range:

INACTIVE
Delete all the unreferenced control entries within the current Unicode

environment and reorganize the Unicode environment to eliminate storage gaps in it. Specify the string literal INACTIVE.

Unreferenced control entries are entities that contain data of the current supported tables and can be obtained while replacing or deleting tables from the Unicode environment.

ALL Delete the whole Unicode environment removing all control structures and tables from the environment. FORCE(YES) is required for this keyword.

FORCE(YES)
Specifies that the system will not check whether applications are currently using the tables. The storage occupied by the tables will be returned to the system.

Samples for parmlib member CUNUNI*xx*

Figure 15 shows how to use the CUNUNI parmlib member to activate a new conversion environment.

```
/*************************************************************/
/*                                                           */
/* CUNUNIXX - UNICODE CONVERSION CONTROL PARAMETERS          */
/*                                                           */
/*************************************************************/
/* ESTABLISH A NEW ENVIRONMENT                               */
/*************************************************************/
/* OPTIONAL KEYWORD REALSTORAGE                              */
/*  MAXIMAL USED PAGES OF REAL STORAGE, MIN=0 MAX=524288     */
/*    WHERE 0 MEANS NO EXPLICIT LIMIT (=524288)              */
/*************************************************************/
/*                                                           */
REALSTORAGE 0;
/*                                                           */
/*************************************************************/
/* OPTIONAL KEYWORD ADD                                      */
/*************************************************************/
/*                                                           */
/*   ADD SPECIFIC TABLES FOR CHARACTER CONVERSION SERVICE    */
/*                                                           */
ADD,FROM=1047 TO=1200,TECH=RECLM
/*                                                           */
/*   ADD CASE CONVERSION SERVICE                             */
/*                                                           */
ADD CASE
/*                                                           */
/*   ADD COLLATION SERVICE                                   */
/*                                                           */
ADD COLL
/*                                                           */
/*   ADD NORMALIZATION SERVICE                               */
/*                                                           */
ADD NORM
/*
```

Figure 15. Example: activate a new conversion environment

Figure 16 on page 304 shows how to delete an inactive conversion environment.

```
/*********************************************************/
/*                                                       */
/* CUNUNIYY - UNICODE CONVERSION CONTROL PARAMETERS      */
/*                                                       */
/*********************************************************/
/*               d e l e t e                             */
/*   an unused inactive Unicode conversion environment   */
/*********************************************************/
/* REQUIRED EXCLUSIVE KEYWORD                            */
/* - NO OTHER KEYWORDS ARE ALLOWED                       */
/*********************************************************/
DELETE INACTIVE;
```

Figure 16. Example: delete an inactive conversion environment

Chapter 28. DEVSUPxx (device support options)

DEVSUPxx specifies the installation default for device support options. DEVSUPxx is processed during the NIP phase of IPL. After IPL, customers can use system command SET DEVSUP=XX to activate the DEVSUP changes.

If no installation default is provided through the DEVSUPxx member, and storing data in a compacted format is not explicitly requested on a DD statement, dynamic allocation request, the MOD= parameter on the JES3 *CALL DJ command, or DCB macro, the system uses the compaction default for the device. For example, the compaction default for a 3480 is NOCOMP. To determine the compaction default for a particular device, see the planning or migration information that accompanies the device.

DEVSUPxx also specifies the installation default for the VOLID facility. The VOLID facility allows tape volumes written in 3480-2 XF (36-track) format and which are mounted on a tape drive not capable of reading this format, to be re-labeled by the OPEN or EOV label editor routines. The label editor routines permit the re-label option only when RACF allows the user ALTER authority to the volume. The volume serial number passed to RACF is obtained from the VOLID mark written on the cartridge by the device, and placed in the sense data.

If no installation default for the VOLID facility VOLNSNS, is provided by using the DEVSUPxx member, the system assumes VOLNSNS=NO.

The MEDIAxx parameters of the DEVSUPxx parmlib member allow the customer to specify partitioning category codes. Library partitioning is the ability to partition volumes in an IBM tape library between different z/OS systems (or sysplexes). Partitioning allows each system (or sysplex) to limit its view of library volumes to only those volumes that it owns. This is accomplished when each system connected to a library uses unique category codes. The DEVSUPxx category codes are read during IPL and the codes are stored into the SSVT (replacing the default category codes).

The IBM Redbook *IBM TotalStorage 3494 Tape Library: A Practical Guide to Tape Drives and Tape Automation* (SG24-4632) lists all default Library Manager volume categories, the platforms on which they are used and their definitions. This book is available at the following URL:

`http://www.redbooks.ibm.com/abstracts/sg244632.html`

DEVSUPxx also specifies the installation default for creating AL tapes. Use the ALVERSION keyword to specify whether AL tapes are created using the ISO/ANSI/FIPS Version 3 or ISO/ANSI Version 4 label standards. ALVERSION is also used to specify if the specified version level should override (force) the current Version 3 or Version 4 labels of an AL tape. ALVERSION is valid only if the AL tape data set is being opened for output processing to the first file of the first or only volume of the data set. Otherwise, the current AL version level of the tape will not be changed, even if force is specified. The default for ALVERSION is Version 3.

The TAPEAUTHxxx keywords in the DEVSUPxx parmlib member enable new options for tape data set security. For more details about the new support, see “Enhanced security for tape data sets” on page 306.

Use the DEVSERV QLIB,CATS command, which is documented in the *z/OS MVS System Commands*, to dynamically change the library partitioning category codes without IPL.

The EXPIRATION_MESSAGE keywords in the DEVSUPxx parmlib member allow you to handle expiration date processing for the opening of any non-VSAM data set on DASD.

The following keywords are set to their default values prior to processing the current DEVSUPxx parmlib member:

KEYWORD:
DEFAULT

COMPACT
NO

ALVERSION
3

NON_VSAM_XTIOT
NO

JES3_ALLOC_ASSIST
NO

TAPE_MULTI_VOLUME_ANOMALY
NOT ALLOW NOT FAIL

COPYSCB
INPUT

EXPIRATION_MESSAGE
ALWAYS

For these keywords, in order for non default values to take affect, they must be specified in either all DEVSUPxx parmlib members processed at IPL or in the last or only DEVSUPxx parmlib member processed either during IPL or after IPL using the system command: SET DEVSUP=xx.

Enhanced security for tape data sets

DFSMS provides additional security functions for tape data sets through the DEVSUPxx parmlib options. DFSMSdfp can now issue RACROUTE in the DATASET class to enable tape data sets to be authorized in the same way that DASD data sets are authorized, regardless of the RACF SETROPTS in use on the system and regardless of the label types used. To enable all data sets on a tape volume or tape volume set to have the same or similar levels of authorization, DFSMSdfp provides a DEVSUPxx option to request that the users' authorization to a single data set on the volume can also extend to the entire volume. When you open a tape data set, DFSMSdfp can also check your authorization to the first file on the volume. The additional RACROUTE issued by DFSMSdfp depends on the DEVSUPxx keywords and the SETROPTS in use. When you select TAPEAUTHDSN=YES and TAPEAUTHF1=YES, the additional RACROUTE matches that issued for the TAPEAUTHDSN option. When you select TAPEAUTHDSN=NO, SETROPTS TAPEDSN, and TAPEAUTHF1=YES, the additional RACROUTE matches that issued for the SETROPTS TAPEDSN option. When all the data sets on a tape volume have a common or similar authorization requirement, an application program has less chance to gain access to unauthorized data by repositioning the tape to another data set.

The function in DFSMSdfp does not replace all the functional capabilities that the RACF TAPEDSN option, TAPEVOL class, and TVTOC provide. However, together with the functions that DFSMSrmm provides, you do have equivalent capability. The enhanced DFSMSdfp function addresses the authorization requirements for tape data sets and relies on your use of a tape management system such as DFSMSrmm to perform the following operations:

- Verify full 44 character data set names.
- Control the overwriting of existing tape files.
- Handle tape data set retention.
- Control the creation and destruction of tape labels.

When you overwrite an existing tape data set and use a different name, DFSMSdfp does not perform any authorization checking for the overwritten data set. However, when you set the TAPEAUTHF1=YES option, your tape management system can provide the data set name for any of the tape data sets causing additional RACROUTE authorization checks. DFSMSrmm exploits this capability to ensure that when the first file is overwritten, there is an additional RACROUTE issued for the overwritten data set. The tape management system controls how you can overwrite the existing tape data sets. See the DFSMSrmm MASTEROVERWRITE parmlib options for more information. Through your tape management system, you can ensure that DFSMSdfp checks the authorization of any overwritten data sets when the TAPEAUTHF1=YES DEVSUPxx parmlib option is in use.

To help you implement the new tape data set security function of DFSMSdfp, additional options allow access to tape data sets that either RACF does not protect or you are not authorized to access. The TAPEAUTHRC4 and TAPEAUTHRC8 options allow you this control. At CLOSE time, OPEN/EOV tracks the use of these options to change the results of RACF processing through SMF records 14 and 15.

You can now implement tape data set authorization checking whether the data set is on tape or DASD, regardless of tape volume profiles. You can use the following options for tape data set authorization:

- TAPEAUTHDSN keyword to cause RACROUTE to be issued in the DATASET class as if for a DASD data set. TAPEAUTHF1 optionally allows authorization checking for the first data set.
- SETROPTS TAPEDSN to cause RACROUTE to be issued in the DATASET class with DSTYPE=T. This optionally allows the use of TAPEVOL profiles with or without TVTOC, and optionally allows authorization check for the first data set using TAPEAUTHF1.
- Tape volume authorization only through TAPEVOL profiles.

For a complete list of the existing options, see *z/OS Security Server RACF Security Administrator's Guide*.

Parameter in IEASYSxx (or Issued By the Operator)

```
DEVSUP=    {aa }
           {(aa,bb...)}
```

The two alphanumeric characters (aa, bb, and so forth) are appended to DEVSUP to form the name of the DEVSUPxx member of SYS1.PARMLIB.

Syntax Rules for DEVSUPxx

The following rules apply to the creation of DEVSUPxx; Figure 17 shows an example of the syntax:

- Each DEVSUPxx parmlib member can contain any number of keyword entries. These entries are processed left to right, top to bottom.
- Comments may appear in columns 1-80 and must begin with "/*". After finding an opening comment delimiter, the rest of the line will be ignored. Therefore, comments may appear as the only entry in a line, or following DEVSUPxx keywords, such as COMPACT and VOLNSNS. Comments cannot span more than one line. Multiple line comments can be included as a series of one-line comments, each starting with the opening comment delimiter.
- One or more blanks may precede or follow the statement types.
- Uppercase or lowercase letters can be used.
- The equal sign "=" between keywords and their values is mandatory. Intervening blanks are allowed between the keyword and the equal sign, and between the equal sign and the assigned value.
- Multiple lines are allowed, but every keyword entry must be complete on a single line. To specify continuation lines, a comma must be coded following the last entry on the line, without intervening blanks and before any comments on that record.

```
COMPACT = YES, VOLNSNS = YES, MEDIA1  = 0011, MEDIA2  = 0012, MEDIA3  = 0013,
MEDIA4  = 0014,
MEDIA5  = 0015,
MEDIA6  = 0016,
MEDIA7  = 0017,
MEDIA8  = 0018,
MEDIA9  = 0019,
MEDIA10  = 001A,
MEDIA11  = 001B,
MEDIA12  = 001C,
MEDIA13  = 001D,
ERROR   = 001E,
PRIVATE = 001F,
ALVERSION = FORCE3
```

Figure 17. Example: Syntax for DEVSUPxx

Syntax Format of DEVSUPxx

```
COMPACT={[YES|NO]}
TAPEAUTHDSN = { YES|NO }
TAPEAUTHF1 = { YES|NO }
TAPEAUTHRC4 = { ALLOW|FAIL }
TAPEAUTHRC8 = { FAIL|WARN }
VOLNSNS={[YES|NO]}
MEDIAn=xxxx
ERROR=xxxx
PRIVATE=xxxx
ALVERSION={3|4|FORCE3|FORCE4}
TAPEBLKSZLIM={nnnnn|nnnnnK|nnnnnM|nG}
COPYSDB={YES|SMALL|LARGE|INPUT|NO}
ENFORCE_DC_MEDIA={ ALLMEDIATY|MEDIA5PLUS }
MTL_NO_DC_WORM_OK
{ENABLE | DISABLE} feature
NON_VSAM_XTIOT={ YES|NO }
TAPE_MULTI_VOLUME_ANOMALY={ALLOW|FAIL }
OCE_ABEND_DESCRIP={ YES|NO }
DDRSIZELIM={xxxx|xxxxM|1000M}
EXPIRATION_MESSAGE={NEVER|ALWAYS}
JES3_ALLOC_ASSIST={YES|NO}
MULTINCRFLC={YES|NO}}
EASYTIERHINTS={YES|NO}
```

IBM-supplied default for DEVSUPxx

- VOLNSNS = NO,
- MEDIA1 = 0001,
- MEDIA2 = 0002,
- MEDIA3 = 0003,
- MEDIA4 = 0004,
- MEDIA5 = 0005,
- MEDIA6 = 0006,
- MEDIA7 = 0007,
- MEDIA8 = 0008,
- MEDIA9 = 0009,
- MEDIA10 = 000A,
- MEDIA11 = 000B,
- MEDIA12 = 000C,
- MEDIA13 = 000D,
- ERROR = 000E,
- PRIVATE = 000F,
- ALVERSION = 3,
- TAPEAUTHDSN = NO,
- TAPEAUTHF1 = NO,
- TAPEAUTHRC4 = FAIL,
- TAPEAUTHRC8 = FAIL,
- DISABLE(REFVTOC),
- DISABLE(REFUCB),
- DISABLE(PPRCMT),
- DISABLE(PPRCSUM),

- NON_VSAM_XTIOT = NO,
- DDRSIZELIM = 1000M,
- TAPE_MULTI_VOLUME_ANOMALY= ALLOW
- OCE_ABEND_DESCRIP=NO
- EXPIRATION_MESSAGE=ALWAYS
- JES3_ALLOC_ASSIST=NO
- MULTINCRFLC=YES

The values listed for MEDIA1, MEDIA2, MEDIA3, MEDIA4, MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, MEDIA10, MEDIA11, MEDIA12, MEDIA13, ERROR, and PRIVATE are IBM defaults. If you use DEVSUPxx to specify these values, they must match the IBM supplied defaults. For example, MEDIA1=0002 is invalid. If you want to specify category 0001, it must be MEDIA1.

Statements/Parameters for DEVSUPxx

ALVERSION=

Specifies whether AL tapes are created using Version 3 or Version 4 standards.

Notes:

1. In all cases, the Volume Mount (VOLMT) exit can override the version specified by ALVERSION. See *z/OS DFSMS Installation Exits* for more information about VOLMT.
2. ALVERSION is valid only if the AL tape data set is being opened for output processing to the first file of the first or only volume of the data set.

3 Specifies that new AL labels are written as ISO/ANSI/FIPS Version 3. The current version 3 and 4 labels are preserved.

4 Specifies that new AL labels are written as ISO/ANSI Version 4. The current version 3 and 4 labels are preserved.

FORCE3

Specifies that all AL labels are forced as ISO/ANSI/FIPS Version 3, including any current version 3 and 4 labels.

FORCE4

Specifies that all AL labels are forced as ISO/ANSI Version 4, including any current version 3 and 4 labels.

COMPACT=

YES

Specifies that data is to be stored in a compacted format on each 3480, 3490, or 359*x* tape subsystem, unless overridden by the user.

NO Specifies that data is not to be stored in a compacted format on each 3480, 3490 or 359*x* tape subsystem, unless overridden by the user. If no installation default is provided through the DEVSUPxx member, and storing data in a compacted format is not explicitly requested on a DD statement, dynamic allocation request, the MOD=parameter on the JES3 *CALL, DJ command, or DCB macro, then the system uses the compaction default for the device. For example, the compaction default for a 3480 is NOCOMP. To determine the compaction default for a particular device, see the planning or migration documentation that accompanies the device.

COMPACT=

YES
Specifies that data is to be stored in a compacted format on each 3480, 3490, or 359*x* tape subsystem, unless overridden by the user.

NO Specifies that data is not to be stored in a compacted format on each 3480, 3490 or 359*x* tape subsystem, unless overridden by the user. If no installation default is provided through the DEVSUPxx member, and storing data in a compacted format is not explicitly requested on a DD statement, dynamic allocation request, the MOD=parameter on the JES3 *CALL, DJ command, or DCB macro, then the system uses the compaction default for the device. For example, the compaction default for a 3480 is NOCOMP. To determine the compaction default for a particular device, see the planning or migration documentation that accompanies the device.

COPYSDB=
Supplies the system-level default for the SDB keyword for IEBGENER. The system uses this value to set a code in the DFA that any application program can use. See *z/OS DFSMSdfp Advanced Services*. The keyword is designed for use by assembler language programs that copy data sets.

The meanings for the keyword values are described in *z/OS DFSMSdfp Utilities*. The default is no code in the DFA, which means that IEBGENER assumes SDB=INPUT.

Note: DFSORT's ICEGENER uses the DFSORT SDB installation value as its default. The IBM-supplied default is SDB=INPUT. See *z/OS DFSORT Installation and Customization* for details.

DDRSIZELIM=
Specifies the limit on storage usage for Tape DDR swap. The value xxxx is a number from 1 to 1000 and specifies the number of megabytes of main storage that is allowed to be used in a Tape DDR swap. The system stores this value in the data facilities area (DFA), for use by the system and by application programs. Tape DDR swap checks this value to make sure that the total amount of storage required in swap processing does not exceed the specified limit. If it does, DDR terminates the swap and prints out an error message. The default value for this parameter is 1000 megabytes.

{ENABLE | DISABLE}(*feature***)**
enables or disables a particular feature, where *feature* can be any one of the following:

NON_VSAM_XTIOT=

NO Disables support for XTIOT, uncaptured UCB, and DSAB control blocks that reside above the 16-MB line for data sets that use BSAM, QSAM, or BPAM. The default value for NON_VSAM_XTIOT is NO.

YES
Enables support for XTIOT, uncaptured UCB, and DSAB control blocks that reside above the 16-MB line for data sets that use BSAM, QSAM, or BPAM.

This controls whether the access method OPEN macro supports these three options of the data set dynamic allocation function. Set NO if you are concerned that some programs, including installed products, might not correctly handle these options. You can set YES if all programs that might process data sets that were dynamically allocated by other programs can handle these options. Setting YES but not using these options has no effect on virtual storage or performance.

PPRCSUM
Enables or disables the PPRCSUM feature of the Device Manager. Note that if the PPRCSUM feature is enabled or disabled after IPL, one device in every control unit must be varied online to activate the feature.

DISABLE(PPRCSUM)
When PPRCSUM feature is disabled, PPRC suspend notification for individual devices is displayed in message IEA494I. The default value for PPRCSUM is DISABLE(PPRCSUM).

ENABLE(PPRCSUM)
Enables the PPRCSUM feature of the Device Manager, which means using message IEA075I instead of IEA494I to report devices that transition to PPRC suspended state. The PPRCSUM feature significantly reduces the volume of messages which are written to the console when devices in a PPRC relationship are suspended. If you enable PPRCSUM, the system will issue an IEA075I message every 5 seconds or when the last device in the control unit has suspended to summarize the PPRC state for all devices in the control unit. This continues until all PPRC state transitions have completed.

Note: If GDPS® or other PPRC monitoring software is active, make sure the appropriate version of this software is installed before enabling PPRCSUM.

OCE_ABEND_DESCRIP=

Note: As of z/OS V1R13 with the installation of APAR OA37957, the enablement of this function is now provided through the MPFLSTxx parmlib member. For details, refer to "Controlling verbose message production" on page 657.

NO Specifies that OPEN, EOV and CLOSE abend messages will not include a descriptive text for the associated numeric abend and numeric return code. To diagnose and respond to the messages, it may be necessary to look up the codes using the z/OS MVS System Messages books or the LookAt tool. The default value for OCE_ABEND_DESCRIP is NO.

YES
Specifies that abend messages for selected OPEN, EOV and CLOSE determinant errors include descriptive text for the associated numeric abend code and numeric return code. This option can eliminate the need to look up the meanings of the abend and reason codes returned in the messages in the z/OS MVS System Messages books or the LookAt tool.

REFVTOC
Enables or disables the automatic REFVTOC function of the Device Manager:

ENABLE(REFVTOC)
When the REFVTOC feature is enabled and the system detects a volume expansion, the Device Manager causes the volume VTOC to be rebuilt. This allows the newly added space on the volume to be used by the system.

DISABLE(REFVTOC)
When the REFVTOC feature is disabled and the system detects a volume expansion, the system issues message IEA019I, but the VTOC

is not rebuilt. An ICKDSF Batch job must be submitted to rebuild the VTOC before the newly added space on the volume can be used. The default value for REFVTOC is DISABLE(REFVTOC).

EASYTIERHINTS=

NO Disables the Easy-Tier Copy Temperature function for software-defined storage. This disables Query/Set Temperature functions that is used to direct data placement in the Disk Controller. NO is the default.

YES
Enables the Easy-Tier Copy Temperature function for software-defined storage. This enables Query/Set Temperature functions that is used to direct data placement in the Disk Controller.

If the Easy-Tier Copy Temperature function for software-defined storage is enabled by setting EASYTIERHINTS=YES in the DEVSUPxx member of SYS1.PARMLIB, an IEA253I message is logged.

```
IEA253I DEVSUP EASY-TIER FOR SOFTWARE DEFINED STORAGE
```

If EASYTIERHINTS=NO is specified followed by issuing the SET DEVSUP=xx command to refresh the PARMLIB member, the IEA253I message is not logged.

If the Easy-Tier Copy Temperature function for software-defined storage is enabled by setting EASYTIERHINTS=YES in the DEVSUPxx member of the PARMLIB followed by an IPL of the system, the EASY-TIER FOR SOFTWARE DEFINED STORAGE status will be seen in the output of the F DEVMAN,REPORT console command. Refer to *z/OS MVS System Commands* for usage of the F DEVMAN,REPORT console command.

If the Easy-Tier Copy Temperature function for software-defined storage is disabled by issuing SET DEVSUP=XX with EASYTIERHINTS=NO in the PARMLIB member, the EASY-TIER FOR SOFTWARE DEFINED STORAGE status will not be seen in the output of the F DEVMAN,REPORT command.

ENFORCE_DC_MEDIA
Specifies whether dataclass media policies are enforced for stand-alone, non-specific mounts.

ALLMEDIATY
Specifies that dataclass media policies are enforced for all stand-alone, non-specific mounts.

MEDIA5PLUS
Specifies that dataclass media policies are enforced for all stand-alone, non-specific mounts for any of the 3592 tape cartridge media types.

EXPIRATION_MESSAGE=

NEVER
Disables expiration date processing when opening a non-VSAM data set on DASD. Set the parameter to NEVER if you want OPEN to disable expiration date processing when opening all non-VSAM data sets on DASD for output processing. Specifying NEVER eliminates the message IEC507D and the optional associated message IEC108I for all non-VSAM data sets on DASD. Therefore, any authorized user can open an expiration date protected non-VSAM DASD data set for output without requiring the operator to allow access.

ALWAYS
Normal expiration date processing will occur when opening a non-VSAM data set on DASD. Set the parameter to ALWAYS to allow OPEN to

process expiration date processing as it normally does, that is, by issuing message IEC507D and the optional associated TSO/E message IEC108I when any attempt is made to open for output a data set on DASD for which the expiration date has not yet occurred. The default value for EXPIRATION_MESSAGE is ALWAYS.

Note: This keyword has no affect for data sets on magnetic tape even if you specify NEVER.

JES3_ALLOC_ASSIST=

YES
Specifies that the allocation assist support (available with the TS7700 Virtualization Engine) is being enabled for usage with JES3. With this support enabled, scratch allocations can be directed (through TS7700's management class policies) to specific clusters (distributed libraries) and specific allocations are directed to a preferred list of clusters (distributed libraries) returned by the library.

Before enabling this support refer to the set-up steps in the "JES3 Considerations" section in *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries*. If you do not perform the setup before enabling this support, jobs may incur abends.

NO Specifies that the allocation assistance support (available with the TS7700 Virtualization Engine) is not to be used by JES3. JES3 scratch and specific allocation requests will continue to be directed to the single library image referred to as the composite library with no knowledge of the underlying clusters (distributed libraries). NO is the default.

MTL_NO_DC_WORM_OK
Specifies that you can use WORM tape in an MTL environment without a Dataclass specification for WORM.

MULTINCRFLC(NO|YES)

NO Disables Incremental FlashCopy Version 2 function. Multiple Incremental FlashCopy targets are not allowed.

YES
Enables the Incremental FlashCopy Version 2 function. Multiple Incremental FlashCopy targets are allowed.

Default: YES.

If Incremental FlashCopy Version 2 is enabled, an IEA253I message is logged, indicating the Incremental FlashCopy version with CHANGE RECORDING V2.

```
IEA253I DEVSUP MULTIPLE INCREMENTAL FLASHCOPY:
                CHANGE RECORDING V2
```

If MULTINCRFLC=NO is specified, followed by using the SET DEVSUP command to refresh the DEVSUPxx parmlib member, the IEA253I message is not logged.

If Incremental FlashCopy Version 2 is enabled by installing the enablement PTF, the output of the F DEVMAN,REPORT console command includes `MULTIPLE INCREMENTAL FLASHCOPY: CHANGE RECORDING V2`. For information about the F DEVMAN,REPORT console command, refer to the topic about MODIFY DEVMAN in *z/OS MVS System Commands*.

PPRCMT
Enables or disables the multi-target PPRC support.

ENABLE(PPRCMT)
When the PPRCMT feature is enabled, support is available to allow a device to be the primary of more than one PPRC pair.

DISABLE(PPRCMT)
When the PPRCMT feature is disabled, functionality will be in single target mode.

Default: DISABLE(PPRCMT)

REFUCB
Enables or disables the automatic REFUCB function of the Device Manager:

ENABLE(REFUCB)
When the REFUCB feature is enabled, the system automatically updates the UCB when device support software detects that a DSS COPY, RESTORE, or ICKDSF REFORMAT NEWVTOC operation has changed either the volser or the VTOC location. In the case of a volser or VTOC location change, the system invokes the DEVMAN REFUCB service on each system in the sysplex that has REFUCB enabled.

- If the device is ONLINE, REFUCB issues a VARY ONLINE, UNCONDITIONAL command, which updates both the volser and VTOC location in the UCB.
- If the device is OFFLINE, no action is taken.

DISABLE(REFUCB)
When the REFUCB feature is disabled, the system does not refresh the UCB when a DSS COPY, RESTORE or ICKDSF REFORMAT NEWVTOC operation has changed either the volser or the VTOC location. The default value for REFUCB is DISABLE(REFUCB).

TAPEAUTHDSN=

YES
Enables tape authorization checks in the DATASET class but without DSTYPE=T. All tape data set names created are RACF-protected.

DSTYPE=T indicates to RACF that the check is for data set on a tape volume and that special RACF tape data set and a tape volume processing is to be performed. Without DSTYPE=T RACF authorization checking considers only profiles in the DATASET class.

The system uses the data set name specified in the allocation or JCL to check your authorization to read or write the specified file.

In addition, the system determines the RACF erase-on-scratch setting from the RACF profile and passes it to your tape management system.

Use this option only when you have a tape management system, such as DFSMSrmm, installed and actively checking that the 44 character data set name specified by the user matches the data set name on tape. Without a tape management system, tape data set open processing can only validate the last 17 characters of the data set name against the tape volume labels.

When you request bypass label processing (BLP) and the mounted volume uses standard labels, OPEN issues the authorization check that the user is authorized to use BLP. This processing uses the existing ICHBLP resource

in the RACF FACILITY class. When you specify TAPEAUTHDSN=YES only, it replaces the check that RACF makes as part of tape volume authorization checking.

NO Indicates that OPEN processing to issue RACROUTEs based on the options set in RACF such as SETROPTS TAPEDSN and SETROPTS CLASSACT(TAPEVOL). The default value for TAPEAUTHDSN is NO.

TAPEAUTHF1=

YES
Enables additional tape authorization checks in the DATASET class for existing files on the same tape volume when any other file on the tape volume is opened. This function depends on the tape management system returning the 44 character data set name and data set sequence number to OPEN/EOV through the IFGTEP during the Volume Mount exit Volume Security function; if no data set name is returned by the tape management system, processing is as if this keyword had not been specified.

Although intended to enable an additional authorization check for the first data set when any other data set on the tape volume is opened, the implementation allows your tape management system to request one or more additional authorization checks when any data set on a tape volume is opened. Each additional data set name and data set sequence number returned results in an additional RACROUTE. Do not use this function unless you have a tape management system and it can return a data set name and data set sequence number. A data set sequence number is the label number normally specified in the JCL LABEL keyword and stored in the catalog.

When TAPEAUTHDSN=YES is in use, any additional RACROUTE matches that issued for TAPEAUTHDSN except for the data set name and data set sequence number. Otherwise TAPEAUTHF1 uses a RACROUTE that matches that used for SETROPTS TAPEDSN. When neither TAPEAUTHDSN nor SETROPTS TAPEDSN is in use, TAPEAUTHF1 support is not provided.

NO Disables additional tape authorization checks in the DATASET class for existing files on the same tape volume when any other file on the tape volume is opened. The default value is NO.

TAPEAUTHRC4=
This applies to authorization checks in the DATASET class, and applies only to the results of TAPEAUTHDSN=YES and TAPEAUTHF1=YES processing.

ALLOW
Allows accessing of data sets that are not protected by a security profile. 'RC4' refers to the return code value of 4 returned from SAF as a result of the RACROUTE issued by OPEN/CLOSE/EOV. A return code of 4 in general means that the resource is not protected.

FAIL
Denies accessing of data sets that are not protected by a security profile. TAPEAUTHRC4=FAIL and TAPEAUTHDSN=YES together ensure that all tape data set names created including temporary names generated by the tape system are RACF-protected. The default value is FAIL.

Use this keyword to control PROTECTALL processing for tape data sets. This applies to the results of RACROUTE processing when both TAPEAUTHDSN=YES and when TAPEAUTHF1=YES are specified.

TAPEAUTHRC8=

Provides a managed and controlled implementation of tape authorization checks in the DATASET class, and applies only to the results of TAPEAUTHDSN=YES and TAPEAUTHF1=YES processing. This keyword is provided as an aid to the implementation of TAPEAUTHDSN and TAPEAUTHF1.

WARN

Allows accessing of data sets that typically cannot be accessed. RACF issues an ICH408I message to indicate why access is not allowed; however OPEN/EOV allows access.

FAIL

Denies accessing of data sets that typically cannot be accessed. The default value is FAIL.

TAPEBLKSZLIM=

Specifies the default block size limit for the system to use when a user omits the block size limit on a DD statement for a tape data set and the data class does not supply one. The system stores this value in the DFA (data facilities area), for use by the system and by application programs. See *z/OS DFSMSdfp Advanced Services*. The system uses this value only in cases when all of the following are true:

- An application program uses the large block interface (LBI) of BSAM or QSAM to open a tape data set for output without DISP=MOD. Check the information for the program; if its maximum block size is 32760 or less, or it cannot write to tape, the program probably does not use the LBI.
- The BLKSIZE (block size) value is omitted from all sources.
- The DD statement or dynamic allocation equivalent and the data class do not specify a BLKSZLIM value.

Restriction: DFSMSdss only supports BLKSZLIM of 65,520 and higher.

An application program that uses EXCP can take this value from the DFA.

If you code K, M, or G at the end of the number, the system multiplies the number by 1024, 1,048,576 or 1,073,741,824 respectively. The minimum values are 32760 when specified in bytes, 32K when specified in kilobytes, 1M when specified in megabytes, and 1G when specified in gigabytes. The maximum values are 2147483648, 2097152K, 2048M and 2G. These maximum block size values are much larger than the system actually supports for BLKSIZE. Coding a large value, however, allows the system to choose the largest optimal block size for the device.

The default for this parameter is 32760. IBM suggests that you not code a value that exceeds 32760 in DEVSUPxx if both of the following are true:

- Your system has a job that writes on tape using the large block interface and the job does not supply a value for BLKSIZE or BLKSZLIM. Programs that use the large block interface include IEBGENER, ICEGENER, DFSORT, and programs compiled with COBOL for OS/390 and VM Version 2 Release 2.
- The tapes with a large block size might be read on a level of MVS that precedes OS/390 Version 2 Release 10 or might be read on another type of system that does not support such large blocks. OS/400® supports large blocks.

TAPE_MULTI_VOLUME_ANOMALY=

Specifies how the system handles any multivolume tape label anomaly

condition that is not yet resolved after the Label Anomaly exit has been called. This keyword is processed at IPL time, but can be changed by the operator using SET DEVSUP=*xx*.

If the Label Anomaly exit sets return code 12, it is honored and in all cases it fails the request, overriding the DEVSUP*xx* setting.

ALLOW
Use this setting to allow applications to process multi-volume tape data sets even when the volume set is incomplete or in the wrong sequence.

FAIL
Use this setting to prevent applications from processing multi-volume tape data sets when the volume set is incomplete or in the wrong sequence.

Default: ALLOW.

When a request is failed, either by the Label Anomaly exit return code 12, or by the FAIL option, the System Completion Codes are: 413-58 (OPEN RDBK), 413-5C (OPEN FIRST), 637-B4 (EOV OUTSEQ), and 637-B8 (EOV LAST).

DFSMSrmm attempts to recover from errors that are noted in the label anomaly exit using the volume sequence information that is recorded in the DFSMSrmm control data set (CDS). Based on information returned by DFSMSrmm, OPEN and EOV processing attempts to resolve the error by updating the volume list in the job file control block (JFCB) and any JFCB extensions.

Tape users can bypass the system multivolume tape label anomaly processing by specifying OPTCD=B in the JCL. See the topic Determining Volume Switch in *z/OS DFSMS Using Magnetic Tapes* , *z/OS MVS JCL User's Guide* , and *z/OS MVS JCL Reference* for details of the considerations using OPTCD=B.

The tape application can optionally recover from a tape label anomaly failure by providing a DCB abend exit for the applications' DCB. When the application DCB abend exit requests recovery, the missing or out of sequence volume condition is ignored.

VOLNSNS=

YES
Specifies that tape cartridges written at track capacities that the drive is not capable of reading (for example, a 36-track cartridge on a D/T3480, or a 256-track cartridge on a D/T3590 Model B1*x*), be relabeled at the device-capable track capacity by the OPEN or EOV label editor routines. This re-label editor option is permitted only if the user is RACF-authorized to the volume. The volume serial number that is passed to RACF is obtained from the VOLID mark written on the cartridge by the device, and placed in the sense data.

NO Specifies that tape cartridges written at track capacities that the drive is not capable of reading (for example, a 36-track cartridge on a D/T3480, or a 256-track cartridge on a D/T3590 Model B1*x*), are not allowed to be re-labeled. Attempts to re-label the cartridges are rejected when RACF protection for tape volumes is active. If no installation default for the VOLID facility VOLNSNS is provided by using the DEVSUPxx member, the system assumes VOLNSNS=NO.

Volume partitioning parameters

Each volume inside an SMS-managed tape library is assigned a two-byte Library Manager category number based on the 'use attribute' of the volume and the media type. The IBM Tape Library Manager category numbers are used by the operating system to group tape volumes together for specific purposes. Currently, DFSMS uses the following fixed set of category numbers:

X'0001'
All 3490 standard capacity(CST) scratch cartridges.

X'0002'
All 3490 enhanced capacity(ECST) scratch cartridges.

X'0003'
All 3590 high performance scratch cartridges.

X'0004'
All 3590 high performance scratch cartridges.

X'0005'
3592 Enterprise Tape Cartridge scratch cartridges.

X'0006'
3592 Enterprise WORM Tape Cartridge scratch cartridges.

X'0007'
3592 Enterprise Economy Tape Cartridge scratch cartridges.

X'0008'
3592 Enterprise Economy WORM Tape Cartridge scratch cartridges.

X'0009'
3592 Enterprise Extended Tape Cartridge scratch cartridges.

X'000A'
3592 Enterprise Extended WORM Tape Cartridge scratch cartridges.

X'000B'
3592 Enterprise Advanced Tape Cartridge scratch cartridges.

X'000C'
3592 Enterprise Advanced WORM Tape Cartridge scratch cartridges.

X'000D'
3592 Enterprise Advanced Economy Tape Cartridge scratch cartridges.

X'000E'
The ERROR category. Used for tape volumes for which the system has detected an error. Tape volumes are added to this category to prevent them from being mounted in response to a 'scratch' tape mount.

X'000F'
The PRIVATE category.

Library partitioning is the ability to partition volumes in an IBM tape library between different z/OS systems (or sysplexes). Partitioning allows each system (or sysplex) to limit its view of library volumes to only those volumes that it owns. Partitioning is accomplished when each system connected to a library uses unique category codes. DEVSUPxx parameters are used to specify category codes that replace the default system codes.

The following DEVSUPxx parameters are used to specify category codes for library partitioning:

MEDIA1=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3490 standard capacity (CST) scratch category code.

MEDIA2=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3490 enhanced capacity (ECST) scratch category code.

MEDIA3=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3590 high performance cartridge tape scratch category code.

MEDIA4=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3590 high performance cartridge tape scratch category code.

MEDIA5=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Tape Cartridge scratch category code.

MEDIA6=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise WORM Tape Cartridge scratch category code.

MEDIA7=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Economy Tape Cartridge scratch category code.

MEDIA8=xxxx
xxxxsspecifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Economy WORM Tape Cartridge scratch category code.

MEDIA9=xxxx
xxxx pecifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Extended Tape Cartridge scratch category code.

MEDIA10=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Extended WORM Tape Cartridge scratch category code.

MEDIA11=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Advanced Tape Cartridge scratch category code.

MEDIA12=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Advanced WORM Tape Cartridge scratch category code.

MEDIA13=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the 3592 Enterprise Advanced Economy Tape Cartridge scratch category code.

ERROR=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the error category code.

PRIVATE=xxxx
xxxx specifies a 2-byte hexadecimal value to be used as the private category code.

Note: xxxx must be a 4-character hexadecimal value within the range 0000-FEFF.

Chapter 29. DFHSSIxx (message-formatting initialization member)

You can use the DFHSSIxx parmlib member to specify message-formatting initialization parameters for the CICS subsystem of the SYS1.PARMLIB library, where xx is the suffix that identifies the SYS1.PARMLIB member used to define the CICS subsystem. For more information about console message handling, see the CICS Transaction Server for z/OS, Version 3.2 Installation Guide (http://publib.boulder.ibm.com/infocenter/cicsts/v3r2) in the IBM Knowledge Center for CICS Transaction Server.

Chapter 30. DIAGxx (control common storage tracking and GFS trace)

DIAGxx contains statements that control the following functions:

- Common service area (CSA), extended CSA (ECSA), system queue area (SQA), and extended SQA (ESQA) tracking.
- GETMAIN/FREEMAIN/STORAGE (GFS) trace.

  To obtain GFS trace data output, GTF must be started for USR F65 records. For more details, see the GFS trace information in *z/OS MVS Diagnosis: Tools and Service Aids*.
- The behavior of GETMAIN (and STORAGE OBTAIN) for CSA allocations (both above and below 64 megabytes).
- Terminating and restarting an initiator address space when its available region size has decreased.
- Preventing the use of user key CSA/ECSA.
- Options to control the behavior of 64 bit storage services: IARCP64 and IARST64.
- Options that control whether some system functions create system trace entries.
- Controlling the behavior of GETMAIN (and STORAGE OBTAIN) for user-region private area subpools (both above and below the line).

See the description of the VERBEXIT VSMDATA subcommand in *z/OS MVS IPCS Commands* for information about:

- How to format the data that the storage tracking function collects from a dump.
- How to identify jobs or address spaces that own CSA, ECSA, SQA, and ESQA storage.

Specifying the DIAGxx members

The following methods can be used to specify the current DIAGxx members:

- The operator can enter the SET DIAG=xx command at any time after IPL.
- You can specify the DIAG=xx parameter in the IEASYSxx parmlib member during IPL, and then select IEASYSxx in the LOADxx parmlib member or specify SYSP=xx in response to the SPECIFY SYSTEM PARAMETERS system message.
- You can specify DIAG=xx in response to the SPECIFY SYSTEM PARAMETERS system message.

You can specify one or more current DIAGxx parmlib members during IPL. For example, a DIAG=(03,04) system parameter tells the system to use DIAG03 and DIAG04 as current members. The system processes these members in the order they are specified. For all statements other than VSM TRACK, only the last statement for that option is honored. Before changing options with SET DIAG=xx, consider issuing a D DIAG command to see what options are currently in affect. Combine the displayed options with any new request that shares the same option statement.

The VSM TRACK parameter is cumulative. For example, if you specify VSM TRACK SQA(ON) in DIAG03 and VSM TRACK CSA(ON) in DIAG04, the system

turns SQA/ESQA tracking on when it processes DIAG03, and turns CSA/ECSA tracking on when it processes DIAG04 (so both are turned on). The VSM TRACE parameter, in contrast, is not. For example, if you specify VSM TRACE GETFREE(ON) SUBPOOL(127) in DIAG03 and VSM TRACE GETFREE(ON) SUBPOOL(0) ASID(1) in DIAG04, the second specification replaces the first, with the result that the system performs a GFS trace for ASID 1 only.

IBM provides the following parmlib members:

DIAG00
Sets storage tracking on and GFS trace off. This is the default; if you do not specify a DIAGxx parmlib member during IPL, the system processes the default member DIAG00.

DIAG01
Sets storage tracking on but does not change GFS trace settings.

DIAG02
Sets storage tracking off but does not change GFS trace settings.

Parameter in IEASYSxx (or specified by the operator)

DIAG=(xx[,yy...])

The two-character identifier *xx* is appended to DIAG to identify the DIAGxx parmlib member. You can also specify multiple DIAGxx members on this parameter. For example, you can specify two active members using the form DIAG=(xx,yy). If you do not specify the DIAG=xx parameter, the system processes the DIAG00 parmlib member.

Syntax rules for DIAGxx

- When creating DIAGxx, enter data only in columns 1 through 71. Do not enter data in columns 72 through 80; the system ignores these columns.
- If the system finds a syntax error in DIAGxx, the system issues an error message, and then attempts to continue processing the next keyword.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

Syntax format of DIAGxx

```
[VSM TRACE                                                    ]
[ {GETFREE(ON)|GET(ON|OFF) FREE(ON|OFF)                     } ]
[   [ASID({asid1|asid1-asidx}[,{asid2|asid2-asidx}]...)]      ]
[   [DATA(data1[,data2]...)                                   ]
[   [KEY({key1|key1-keyx}[,{key2|key2-keyx}]...)]             ]
[   [LENGTH({len1|len1-lenx}[,{len2|len2-lenx}]...)]          ]
[   [SUBPOOL({sub1|sub1-subx}[,{sub2|sub2-subx}]...)]         ]
[   [JOBNAME([job1,job2...])                                  ]
[   [ADDRESS([addr1|addr1-addrx][,addr2-|addr2-addrx..)]      ]
[   [LOCREAL(loc1[,loc2]...)                                  ]
[ {GETFREE (OFF)                                            } ]
[VSM TRACK                                                    ]
[ {CSA (ON|OFF)                                             } ]
[ {SQA (ON|OFF)                                             } ]
[ {CSA (ON|OFF) SQA (ON|OFF)                                } ]
[VSM CHECKREGIONLOSS(bbb{K|M},aaa{K|M})                       ]
[VSM ALLOWUSERKEYCSA(NO|YES)                                  ]
[VSM BESTFITCSA(NO|YES)                                       ]
[CBLOC [VIRTUAL24(structure1,structure2, ...)                 ]
[VIRTUAL31(structure1,structure2, ...)]                       ]
[REUSASID(NO|YES)                                             ]
[VSM USEZOSV1R9RULES(NO|YES)                                  ]
[TRAPS NAME ([trapname1,{trapname2}...])                      ]
[AUTOIPL SADMP(sadmp info) MVS(mvs info)                      ]
[CSRPOOLDIAG                                                  ]
```

IBM-supplied default for DIAGxx

DIAGxx has the following IBM-supplied default parmlib members.

IBM-supplied default parmlib member	Contents
DIAG00	• VSM TRACE GETFREE (OFF) • VSM TRACK CSA(ON) SQA(ON)
DIAG01	• VSM TRACK CSA(ON) SQA(ON)
DIAG02	• VSM TRACK CSA(OFF) SQA(OFF)

If you do not specify a DIAGxx parmlib member during IPL, the system uses the default member DIAG00, which turns on storage tracking and turns off GFS trace.

If you specify a DIAGxx parmlib member during IPL omitting either the storage tracking or GFS trace statement, the omitted element is turned off.

The system expects a default member to be present. If you do not specify a DIAGxx parmlib member during IPL, and none of DIAG00, DIAG01, or DIAG01 exists, the system uses the following options:

- VSM TRACE GETFREE (OFF)
- VSM TRACK CSA(ON) SQA(ON)

In this case:

- CSA and SQA tracking are active and will remain active until turned off through a SET DIAG=*xx* command.
- GFS tracing is off.
- The system issues informational message IEA301I to the operator.

Statements/parameters for DIAGxx

ADDRESS(addr1|addr1–addrx,[addr2|addr2–addrx]...)
Specifies that the system is to produce trace records only for requested storage of specific addresses. You can specify up to eight storage addresses. Each address must be a hexadecimal number from 0 to 7FFFFFFF. If you do not specify this parameter, the system produces trace records for requested storage of all addresses.

You can specify a specific address, a range of addresses, or any combination of both, as shown in the following examples:

- ADDRESS(0–FFFFFF) - The system produces trace records for all addresses less than 16 megabytes.
- ADDRESS(8000,70000000–7FFFFFFF) - The system produces trace records for addresses 8000 and 70000000–7FFFFFFF.

If you specify a range of addresses, ensure that the address at the end of the range is greater than or equal to the address of the beginning of the range. A request matches a range if any byte of the request is within the range.

ADDRESS filtering does not apply to a subpool FREEMAIN request, causing a trace record to be produced for the subpool FREEMAIN, and an associated Releasing Subpool trace record. (For a description of the "Releasing Subpool" trace record, refer to Formatted GFS Trace Output in *z/OS MVS Diagnosis: Tools and Service Aids*, under GETMAIN, FREEMAIN, STORAGE (GFS) Trace.)

ASID({asid1|asid1-asidx}[,{asid2|asid2-asidx}]...)
Indicates that the system is to produce trace records only for one or more specified address space identifiers (ASIDs). Each ASID must be a hexadecimal number from 0 to 7FFF. The ASID that the system checks when it determines whether to produce a trace record is as follows:

Private Storage=
The target address space of the storage.

CSA Storage=
If CSA tracking is on, the associated ASID is the one indicated by the OWNER parameter on GETMAIN or STORAGE. If CSA tracking is off (or for a request to release storage, was off when the storage was obtained), the associated ASID is unknown. ASID filtering is not done and a trace record is not produced.

SQA Storage =
If SQA tracking is on, the associated ASID is the one indicated by the OWNER parameter on GETMAIN or STORAGE. If SQA tracking is off (or for a request to release storage, was off when the storage was obtained), the associated ASID is unknown. ASID filtering is not done and a trace record is produced.

ASID 0 matches common storage requests with OWNER=SYSTEM.

If you omit this parameter, the system produces trace records for all ASIDs. You can specify from 1 to 32 ASIDs.

You can specify a specific ASID, a range of ASIDs, or any combination of both, as shown in the following examples:

- ASID(5,6,9) - The system produces trace records for ASIDs 5, 6, and 9.
- ASID(5-7,9,11-13) - The system produces trace records for ASIDs 5, 6, 7, 9, 11, 12, and 13.

If you specify a range of ASIDs, ensure that the ASID at the end of the range is greater than the ASID at the beginning of the range.

AUTOIPL SADMP(sadmp info) MVS(mvs info)

(sadmp info) is either (device,loadparm) or (NONE)

(mvs info) is (device,loadparm) or (LAST) or (NONE)

SADMP (device,loadparm)
If MVS is about to enter a wait state, SADMP is loaded from this volume with this load parameter.

SADMP (NONE)
If MVS is about to enter a wait state, SADMP is not loaded.

MVS (device,loadparm)
If MVS is about to enter a wait state, MVS is loaded from this volume with this load parameter, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

MVS LAST
If MVS is about to enter a wait state, MVS is loaded from the same volume and with the same load parameter as used for the current IPL, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

MVS NONE
If MVS is about to enter a wait state, MVS is not loaded, either immediately or following the completion of SADMP processing (if SADMP with a device and load parameter was also coded).

Note:

1. SADMP specified with a device and load parameter and MVS specified with NONE causes the AutoIPL function to IPL SADMP only.
2. SADMP specified with NONE and MVS specified with a device and load parameter or with LAST causes the AutoIPL function to re-IPL MVS immediately, with no SADMP taken.
3. SADMP with a device and load parameter and MVS with a device and load parameter or with LAST causes SADMP to be IPLed, followed by MVS.
4. Any valid specification of AUTOIPL causes any prior AutoIPL information to be replaced.
5. SADMP (NONE) and MVS (NONE) both specified will effectively deactivate the AutoIPL function.
6. The SADMP loadparm can be specified so that SADMP executes without prompts to the operator. For information on coding the SADMP load parameter, see *z/OS MVS Diagnosis: Tools and Service Aids*.
7. The MVS with LAST parameters do not support HyperSwaps in which the IPL, IODF, and SADMP secondary volumes are not part of an alternate subchannel set (that is, the primary and secondary device pairs do not have identical device numbers and all reside in a single subchannel set). Therefore, if MVS with LAST was specified in that environment and a HyperSwap® swapped different device numbers within subchannel set zero and no intervening IPL occurred, the MVS with LAST function will still attempt to use the original IPL, IODF, and SADMP volumes before the swap.

8. Device is a 4-digit device number that can be prefixed by *. The asterisk prefix denotes that the device in the currently active subchannel set should be used. If an asterisk does not prefix the device number, the device in subchannel set 0 is used.

   This function is intended for HyperSwap environments that use alternate subchannel sets. Environments without HyperSwap or HyperSwap environments that do not use an alternate subchannel set will not benefit from using *.
9. Only in a HyperSwap environment that uses an alternate subchannel set will the MVS with LAST parameters ensure that AUTOIPL will use the device number from the appropriate subchannel set in the event of a HyperSwap.

For more information about AutoIPL, see Exploiting the Automatic IPL Function in *z/OS MVS Planning: Operations*.

CBLOC [VIRTUAL24(structure1,structure2, ...)]
[VIRTUAL31(structure1,structure2, ...)]

VIRTUAL24
Provides the names of structures to be located in 24-bit virtual storage.

VIRTUAL31
Provides the names of structures to be located in 31-bit virtual storage.

Structure names:

IHALCCA
When a standard processor is brought online, the location of its LCCA is determined by which list includes IHALCCA. If neither of the lists include IHALCCA, the LCCA is in 31-bit virtual storage. If both lists include IHALCCA, the resulting location is undocumented. The LCCA of the IPL CPU is built in 24-bit storage before DIAGxx is processed.

If additional standard processors are brought online later in the IPL process, the IPL processor is automatically configured offline and back online, with its LCCA located according to the CBLOC specification.

IHAPCCA
When a standard processor is brought online, the location of its PCCA is determined by which list includes IHAPCCA. If neither of the lists include IHAPCCA, the PCCA is in 31-bit virtual storage. If both lists include IHAPCCA, the resulting location is undocumented. The PCCA of the IPL processor is built in 24-bit storage before DIAGxx is processed.

If additional standard processors are brought online later in the IPL process, the IPL processor is automatically configured offline and back online, with its PCCA located according to the CBLOC specification.

IHAASVT
When the system is IPLed, the location of its ASVT is determined by which list includes IHAASVT. If neither of the lists include IHAASVT, the ASVT is in 24-bit virtual storage. If both lists include IHAASVT, the resulting location is undocumented.

CSA (ON | OFF)
The status of the common service area (CSA and ECSA) tracking function. Specifying CSA(ON) might lead to a small performance degradation and an increase in ESQA use. Omitting this parameter leaves the status of CSA/ECSA tracking unchanged.

CSRPOOLDIAG
The get and free services of the CSR cellpool returns diagnostic data to the caller. Use this option when directed by IBM Level 2 only. Do not use this option on z/OS 1.13 if that system is using JES from a release that is earlier than z/OS 1.12.

DATA(data1[,data2]...)
Specifies the data items that you want to include in the trace, which can be one or more of the following:

Data	Information included in trace
ALL	All trace information (BASIC plus REGS). If you omit the DATA keyword, the default is DATA(ALL).
REGS	The contents of the caller's registers when the system processed the linkage instruction to GETMAIN, FREEMAIN, or STORAGE.
BASIC	All of the trace information except REGS. The BASIC data is included in every trace record.

Note: RETADDR, RETCODE, TYPE, SVCNUM, ADDR, LENGTH, SPKEY, FLAGS, ASID, and TCB are accepted for compatibility with older releases and are ignored.

GET(ON|OFF) FREE(ON|OFF)

GETFREE ON|OFF)
Specifies the current status of the GFS trace.

KEY({key1|key1-keyx}[,{key2|key2-keyx}]...)
Specifies the storage keys for which the system is to produce trace records. If you do not specify this parameter, the system produces trace records for all storage keys. You can specify any number of keys or key ranges. Each key must be a decimal number from 0 to 15.

If you specify a range of keys, ensure that the key at the end of the range is greater than or equal to the key at the beginning of the range.

You can specify a specific key, a range of keys, or any combination of both, as shown in the following examples:

- KEY(1,2) - The system produces trace records for storage keys 1 and 2.
- KEY(1-3,15) - The system produces trace records for storage keys 1, 2, 3 and 15.

KEY filtering does not apply to a subpool FREEMAIN request, causing a trace record to be produced for the subpool FREEMAIN, and an associated Releasing Subpool trace record. (For a description of the "Releasing Subpool" trace record, refer to Formatted GFS Trace Output in *z/OS MVS Diagnosis: Tools and Service Aids*, under GETMAIN, FREEMAIN, STORAGE (GFS) Trace.)

JOBNAME(job1{,job2}...)
Specifies that the system is to produce trace records only for one or more specified job names. Each job name must be from 1 to 8 alphanumeric or national characters. The wildcard characters ? and * can be included.

The job name that the system checks when it determines whether to produce a trace record is the job name for the ASID that would be used to match an ASID filter (see the ASID parameter).

Private Storage=
The target JOBNAME of the storage.

CSA Storage=
If CSA tracking is on, the associated JOBNAME is the one indicated by the OWNER parameter on GETMAIN or STORAGE. If CSA tracking is off (or for a request to release storage, was off when the storage was obtained), the associated JOBNAME is unknown, causing no JOBNAME filtering to be done and a trace record to be produced.

SQA Storage =
If SQA tracking is on, the associated JOBNAME is the one indicated by the OWNER parameter on GETMAIN or STORAGE. If SQA tracking is off (or for a request to release storage, was off when the storage was obtained), the associated JOBNAME is unknown, causing no JOBNAME filtering to be done and a trace record to be produced.

LENGTH({len1|len1-lenx}[,{len2|len2-lenx}]...)
Indicates that the system is to produce trace records only for requested storage of specific lengths (in bytes). You can specify up to eight storage lengths. Each length must be a decimal number from 1 to 10 digits (the maximum value is 2147483640 bytes). If you do not specify this parameter, the system produces trace records for requested storage of all lengths.

Specify each length as a multiple of 8 bytes. If you do not, the system rounds the value up to the next higher multiple of 8.

You can specify a specific length, a range of lengths, or any combination of both, as shown in the following examples:

- LENGTH(512,1024) - The system produces trace records for requested lengths of 512 and 1024 bytes.
- LENGTH(512,520-528,1024-1032) - The system produces trace records for requested lengths of 512, 520-528, and 1024-1032 bytes.

If you specify a range of lengths, ensure that the length at the end of the range is greater than or equal to the length at the beginning of the range. If the requested storage is of a variable length, the system uses the length of the storage that was actually obtained.

LENGTH filtering does not apply to a subpool FREEMAIN request, causing a trace record to be produced for the subpool FREEMAIN, and an associated Releasing Subpool trace record. (For a description of the "Releasing Subpool" trace record, refer to Formatted GFS Trace Output in *z/OS MVS Diagnosis: Tools and Service Aids*, under GETMAIN, FREEMAIN, STORAGE (GFS) Trace.)

LOCREAL(loc1[,loc2]...)
Specifies the real storage locations for which trace records should be produced, as specified by the LOC keyword on the GETMAIN and/or STORAGE macros. For example, LOCREAL(24,31) requests tracing for those storage requests that specified real storage backing below the line (LOC=24 or LOC=(xx,24)) and those storage requests that specified real storage backing in 31-bit storage (LOC=(xx,31)).

LOCREAL(64) traces all storage requests that specify LOC(xx,64) or LOC(xx,PAGEFRAMESIZE1MB).

Valid parameters for LOCREAL are 24, BELOW, 31, ANY, 64, and PAGEFRAMESIZE1MB:

24 | BELOW
The system produces trace records for requests that specify real storage backing below the line.

31 | ANY
The system produces trace records for requests that specify real storage backing in 31-bit storage.

64 The system produces trace records for requests that specify real storage backing in 64-bit storage.

PAGEFRAMESIZE1MB
The system produces trace records for requests that specify LOC(xx,PAGEFRAMESIZE1MB).

If you do not specify this parameter, trace records are produced for all storage locations.

REUSASID(NO|YES)
When a reusable ASID is requested by the START command or the ASCRE macro, a reusable ASID is assigned if REUSASID(YES) is specified in DIAGxx. If REUSASID(NO) is specified in DIAGxx, an ordinary ASID is assigned. The default is REUSASID(YES).

The use of reusable ASIDs might result in system 0D3 abends, if products or programs have not been upgraded to tolerate reusable ASIDs. For more information about reusable ASIDs, see *z/OS MVS Programming: Extended Addressability Guide*.

SQA(ON|OFF)
The status of the system queue area (SQA and ESQA) tracking function. Specifying SQA(ON) might lead to a small performance degradation and an increase in ESQA usage. Omitting this parameter leaves the status of SQA/ESQA tracking unchanged.

SUBPOOL({sub1|sub1-subx}[,{sub2|sub2-subx}]...)
Specifies the subpools for which the system is to produce trace records. If you omit this parameter, the system produces trace records for all subpools. You can specify any number of subpools.

You can specify a specific subpool, a range of subpools, or any combination of both, as shown in the following examples:

- SUBPOOL(129,132) - The system produces trace records for subpools 129 and 132.
- SUBPOOL(129-131, 227-229, 252) - The system produces trace records for subpools 129, 130, 131, 227, 228, 229 and 252.

If you specify a range of subpools, ensure that the subpool at the end of the range is greater than or equal to the subpool at the beginning of the range.

TRAPS NAME ([trapname1,{trapname2}...])
Specifies a list of the traps that should be activated. If you issue command SET DIAG=xx and the DIAGxx parmlib member contains a TRAPS NAME statement, the list of TRAPS specified replaces the previous list of activated traps. This deactivates any traps that are not in the new list.

Specifying a null list of trap names deactivates all the TRAPS. If your DIAGxx parmlib member does not contain a TRAPS statement, there is no change to the list of traps that are activated. If you specify multiple parmlib members (DIAG=(03,04)) and both contain TRAPS statements or you code multiple TRAPS statements in a single parmlib member, only the last TRAPS statement is honored.

The following traps are supported:

IarCp64InitFree
The IARCP64 services is to initialize storage to contain non-zero values on a REQUEST=FREE.

Note: This is intended to be used in a test environment to surface programs that reference storage after it has been freed with IARCP64 REQUEST=FREE. Since this will potentially cause additional pages to be backed, consume additional CPU cycles and possibly cause additional page faults, activating this option in a production environment should be avoided.

IarCp64InitGet
The IARCP64 service is to initialize storage to contain non-zero values on a REQUEST=GET.

Note: This service is intended to be used in a test environment to surface programs that obtain cells from a cell pool with IARCP64, but fail to initialize it before using it. Because this will likely cause additional pages to be backed and possibly additional page faults, activating this option in a production environment should be avoided.

IarCp64Trailer
The IARCP64 REQUEST=BUILD service is to activate trailer processing for cell pools, even if it means increasing the size of the cells to make room for the trailer.

Note: This service is intended to be used in a test environment to surface programs that modify storage past the end of the requested cell size. When a cell is freed with IARCP64 REQUEST=FREE, the service abends the caller if it detects that the trailer has been overlaid Since this will potentially increase the size of the cells, it can reduce the number of cells per extent, which can cause additional pages to be backed, consume additional CPU cycles and possibly cause additional page faults, activating this option in a production environment should be avoided.

IarSt64InitFree
The IARST64 service is to initialize storage to contain non-zero values on a REQUEST=FREE.

Note: This service is intended to be used in a test environment to surface programs that reference storage after it has been freed with IARST64 REQUEST=FREE. Since this will potentially cause additional pages to be backed, consume additional CPU cycles and possibly cause additional page faults,, activating this option in a production environment should be avoided.

IarSt64InitGet
The IARST64 service is to initialize storage to contain nonzero values on a REQUEST=GET.

Note: This service is intended to be used in a test environment to surface programs that obtain storage with IARST64, but fail to initialize it before using it. Since this will likely cause additional pages to be backed and possibly additional page faults, activating this option in a production environment should be avoided.

IarSt64Trailer
The IARST64 REQUEST=GET service is to force trailer processing for

storage requests, even if it means increasing the size of the obtained storage cell to make room for the trailer.

Note: This service is intended to be used in a test environment to surface programs that modify storage past the end of the requested storage size. When storage is freed with IARST64 REQUEST=FREE, the service abends the caller if it detects that the trailer has been overlaid Since this will potentially increase the size of the storage cell, it can reduce the number of cells per extent, which can cause additional pages to be backed, consume additional CPU cycles and possibly cause additional page faults, activating this option in a production environment should be avoided.

IeaSlipConfirm
Checks to see if JOBNAME or ASID is specified when MODE=HOME is specified on the SLIP command. If both JOBNAME and ASID parameters were omitted, SLIP issues message IEE088D asking if you would like to continue.

IEARPSGNLTRACE
Turns on system tracing of RPSGNL.

IEASCHEDULETRACE
Turns on system tracing of SCHEDULE and IEAMSCHD.

Note: Adding tracing of SRB scheduling could cause a very small degradation in performance.

IEARISGNLTRACE
Turns on system tracing of RISGNL.

VSM TRACE
Indicates that the statement defines a GFS trace of storage that is obtained and released.

VSM USEZOSV1R9RULES(NO|YES)
YES causes GETMAIN and STORAGE OBTAIN behavior to be unchanged from its historic behavior. NO causes GETMAIN and STORAGE OBTAIN behavior for user-region private area subpools that are both below and above the line to be implemented. Thus DQEs can be merged where possible. The default is YES to provide a seamless migration. However, IBM recommends that you specify USEZOSV1R9RULES(NO) to obtain a performance benefit for applications with long DQE/FQE chains for user-region private area subpools.

VSM ALLOWUSERKEYCSA(NO|YES)
NO prevents user key CSA from being allocated by failing any attempt to obtain user key from a CSA subpool (through GETMAIN or STORAGE OBTAIN) with a B04-5C, B0A-5C, or B78-5C abend. The default is NO. IBM recommends that you should not specify ALLOWUSERKEYCSA(YES). User key CSA creates a security risk because any unauthorized program can modify it.

VSM BESTFITCSA(NO|YES)
VSM BESTFITCSA(NO|YES) indicates how GETMAIN or STORAG|OBTAIN process requests for (E)CSA storage. NO indicates to use a "first fit" algorithm in certain situations such as when you use the STARTBDY and CONTBDY options and is the default. It matches the behavior on all current releases of z/OS. However, in some environments this option can lead to (E)CSA fragmentation. YES indicates to always use a "best fit" algorithm. Specify YES

for this option to minimize (E)CSA fragmentation and to prevent user and system outages because of requests for (E)CSA storage that the system cannot satisfy.

VSM CHECKREGIONLOSS(*bbb***{K|M},***aaa***{K|M})**
Indicates the amount of region size loss that can be tolerated in an initiator address space. The initiator remembers the initial maximum available region size (below and above 16 MB) before it selects its first job. After the termination of each job run in the initiator, if the maximum available region size (below or above 16 MB) has decreased from the initial value by more than the CHECKREGIONLOSS specification, the initiator terminates with message IEF093I or IEF094A, depending on whether the subsystem automatically restarts the initiator. JES2, JES3, WLM, OMVS, and APPC all automatically restart initiators, so initiator issues IEF093I in most cases. CHECKREGIONLOSS allows the installation to avoid 822 abends in subsequent jobs that are selected by an initiator. The available region size of the initiator has decreased because of storage fragmentation or problems that have caused storage not to be freed.

Because the initiator notifies the owning subsystem that the initiator is being terminated on the job termination SSI call, the CHECKREGIONLOSS detection must be done before some storage that will eventually get freed actually gets freed. The SWA subpool is an example (and for some jobs, a large example) of this. The detection processing recognizes storage that is part of the SWA subpool, and treats it as if it was freed. However, if you are looking at a dump of an IEF093I message, you still need to manually ignore the SWA storage.

Some fragmentation (especially above 16 MB) is normal. A job that uses a lot of SWA (or other system control blocks in high extended private) might cause fragmentation because this forces another system control block that persists across jobs to be allocated at a lower address. The VSM cell pool extents (VSMP eye catcher at the beginning of a 4 KB page) are an example of persistent storage. Compression of VSM cell pool extents requires a large amount of CPU time (much more than the CPU time for terminating and restarting an initiator). Recycling of initiators under the control of CHECKREGIONLOSS is the intended mechanism for dealing with fragmentation.

In addition to normal fragmentation, IEF093I might in some cases expose a storage leak problem that can be investigated and fixed. Differentiating between normal fragmentation and a storage leak is a time consuming manual process done by analyzing dumps.

Selecting the CHECKREGIONLOSS values:

- Select values small enough to avoid 822 abends for the jobs in your installation that have the largest REGION requirements. That depends on the size of your private area above and below 16M, and the REGION requirements of your largest jobs.
- Select values large enough so that the frequency of IEF093I messages is not excessive, and so that the frequency of initiator recycling is not a performance issue.

  Start with something like:

  ```
  CHECKREGIONLOSS(256K,10M)
  ```

  Decrease the values if you see 822 abends, and increase the values if you frequently see IEF093I messages.

bbb
Specifies the region size below 16 MB.

aaa
Specifies the region size above 16 MB.

{K|M}
Specifies the unit of measure that is used to define the region size.

VSM TRACK
Indicates that the system is to process the VSM tracking parameters.

Example 1: This example shows a DIAGxx parmlib member that starts GFS trace for requests to obtain or release virtual storage in subpools 1 through 127 and 229. The GFS trace includes requests for:
- Address spaces 1 and F
- A length of 4096 bytes
- Keys 8, 10, 11, and 12.

GFS trace includes all data in the trace, with the exception of register information.

```
VSM TRACE GETFREE(ON)
          SUBPOOL(1-127,229)
          ASID(1,F)
          LENGTH(4096)
          KEY(8,10-12)
          DATA(RETADDR,RETCODE,TYPE,SVCNUM,
               ADDR,LENGTH,SPKEY,FLAGS,ASID,TCB)
```

Example 2: This example shows a DIAGxx parmlib member that starts GFS trace for requests to obtain or release virtual storage in subpools 129 through 255. The GFS trace includes requests for:
- All address spaces (the ASID keyword is not specified)
- Lengths in the 4096-8192 byte range
- All keys (the KEY keyword is not specified)

GFS trace includes all data in the trace, with the exception of register information.

```
VSM TRACE GETFREE(ON)
          SUBPOOL(129-255)
          LENGTH(4096-8192)
          DATA(RETADDR,RETCODE,TYPE,SVCNUM,
               ADDR,LENGTH,SPKEY,FLAGS,ASID,TCB)
```

Example 3: This example shows a DIAGxx parmlib member that turns CSA/ECSA tracking on and turns SQA/ESQA tracking off:

```
VSM TRACK CSA(ON) SQA(OFF)
```

Example 4: Example 4: This example shows a DIAGxx parmlib member that controls termination and restarting of initiator address spaces when the amount of region loss exceeds 256K below 16M or 10M above 16M.

```
VSM CHECKREGIONLOSS(256K,10M)
```

Example 5: Example 5: This example shows a DIAGxx parmlib member that prevents the use of user key CSA.

```
VSM ALLOWUSERKEYCSA(NO)
```

Example 6: Example 6: This example shows a DIAGxx parmlib member that enables the reuse of ASID.

```
REUSASID(YES)
```

Example 7: Example 7: This example shows a DIAGxx parmlib member that specifies that LCCAs and PCCAs should be located above 16 MB.

```
CBLOC VIRTUAL31(IHALCCA,IHAPCCA)
```

Example 8: Example 8: This example shows a DIAGxx parmlib member that specifies several traps to be activated. This turns on all of the diagnostic options relating to IARCP64 requests.

```
TRAPS NAME(IARCP64INITGET,IARCP64INITFREE,IARCP64TRAILER)
```

Note: This example shows a DIAGxx parmlib member that turns off all traps. This turns off all of the TRAP options.

```
TRAPS NAME()
```

For examples of output from storage tracking, see *z/OS MVS Diagnosis: Tools and Service Aids*. For information about the location of GFS trace records, see *z/OS MVS Diagnosis: Tools and Service Aids*.

Chapter 31. EPHWP00 (BookManager topic extraction)

To use man pages on z/OS UNIX, you must have a SEPHTAB data set cataloged in your system. (See *z/OS UNIX System Services Planning* for information about cataloging the SEPHTAB data set.)

You must make the SEPHTAB data set name available to the BookRead service called by the "man" command. You do this with one of the following methods:

1. Use the default data set EPH.SEPHTAB.
2. Specify a non-default prefix of the SEPHTAB data set, right-justified on line one of the EPHWP00 member in SYS1.PARMLIB.
3. Specify a non-default prefix of the SEPHTAB data set, right-justified on line one of the default HFS configuration file: /etc/booksrv/bookread.conf
4. Specify a non-default prefix of the SEPHTAB data set, right-justified on line one of a NON-default HFS configuration file. Specify the non-default HFS configuration file by setting the EPHBookReadConfig environment variable to the HFS file name and path.
5. Specify a data set containing the members of SEPHTAB but not ending in SEPHTAB by entering "DSN=MY.FB4096.DATA.SET" right-justified without the double quotation marks on line one of 'SYS1.PARMLIB(EPHWP00)'.
6. Specify a data set containing the members of SEPHTAB but not ending in SEPHTAB by entering "DSN=MY.FB4096.DATA.SET" right-justified without the double quotation marks on line one of the default HFS configuration file: /etc/booksrv/bookread.conf
7. Specify a data set containing the members of SEPHTAB but not ending in SEPHTAB by entering "DSN=MY.FB4096.DATA.SET" right-justified without the double quotation marks on line one of a non-default HFS configuration file. Specify the NON-default HFS configuration file by setting the EPHBookReadConfig environment variable.

To use man pages on z/OS UNIX, you must have a SEPHTAB data set cataloged in your system and an EPHWP00 member in SYS1.PARMLIB. The SEPHTAB data set contains translation tables used to translate data from the internal BookManager® softcopy format to the code page BookManager is being asked to display. (See *z/OS UNIX System Services Planning* for information about cataloging the SEPHTAB data set.)

A sample EPHWP00 parmlib member is provided in SEPHSAMP. To use man pages, you must copy that sample from SEPHSAMP into SYS1.PARMLIB.

A sample EPHWP00 parmlib member is provided in SEPHSAMP. To override the default EPH.SEPHTAB data set name, you must copy that sample from SEPHSAMP into the configuration file /etc/booksrv/bookread.conf or into SYS1.PARMLIB.

The sample EPHWP00 parmlib member contains a left-justified statement of "EPH". This is the IBM-supplied prefix for the SEPHTAB data set. If you change the prefix of the SEPHTAB data set, you must manually change this left-justified statement to match the prefix of the SEPHTAB data set. For example, if you used a prefix of ABC in the SEPHTAB data set, you must change the EPHWP00 parmlib member from "EPH" to "ABC".

If you do not change the prefix of the SEPHTAB data set, no changes are required to /etc/booksrv/bookread.conf or to the EPHWP00 parmlib member.

Parameter in IEASYSxx (or issued by the operator)

None.

Syntax rules for EPHWP00

Line one of a bookread configuration file or EPHWP00 member of SYS1.PARMLIB starting with DSN= indicates a fully qualified data set name follows the "=".

"DSN=" NOT occurring on line one indicates that line one begins with a prefix for the SEPHTAB suffixed data set.

Comments cannot be entered on the first line. Comments can be entered on subsequent lines.

Syntax format of EPHWP00

There is no syntax for EPHWP00. If the EPH prefix of the SEPHTAB data set is changed, it must be manually changed in /etc/booksrv/bookread.conf or in SYS1.PARMLIB.

IBM-supplied default for EPHWP00

There is no default parmlib member supplied by IBM. A sample member, EPHWP00, is provided in SEPHSAMP.

Chapter 32. EXITxx (allocation installation exit list)

Use the EXITxx member of SYS1.PARMLIB to specify the following installation exits for the following allocation decisions.

Volume ENQ
When a job must wait to enqueue on a volume or a series of volumes, code the volume ENQ exit routine to make exceptions, if any, to the installation default policy for certain jobs and/or volumes. See Chapter 4, "ALLOCxx (allocation system defaults)," on page 71 for information about specification of the installation default policy. See *z/OS MVS Installation Exits* for information about coding the volume ENQ installation exit.

Volume Mount
When a job's allocation request requires a volume to be mounted, code the volume mount exit routine to make exceptions, if any, to the installation default policy for certain jobs and/or volumes. See Chapter 4, "ALLOCxx (allocation system defaults)," on page 71 for information about specification of the installation default policy. See *z/OS MVS Installation Exits* for information about coding the volume mount installation exit.

Specific Wait
When a job must wait for a specific volume or device to become available, code the specific waits exit routine to make exceptions (for certain jobs and/or volumes) to the installation default policy. See Chapter 4, "ALLOCxx (allocation system defaults)," on page 71 for information about specification of the installation default policy. See *z/OS MVS Installation Exits* for information about coding the specific waits installation exit.

Allocated/Offline Device
When a job must wait because the device it requested is offline or allocated to another job, code the allocated/offline device exit routine to make exceptions, if any, to the installation default policy for certain jobs and/or devices. See Chapter 4, "ALLOCxx (allocation system defaults)," on page 71 for information about specification of the installation default policy. See *z/OS MVS Installation Exits* for information about coding the allocated/offline device installation exit.

Note: IBM provides the PROGxx parmlib member as an alternative to EXITxx. PROGxx allows you to specify exits, control their use, and associate one or more exit routines with exits, at IPL or while the system is running. IBM suggests that you use PROGxx to specify exits whether or not you want to take advantage of these functions.

You can convert the format of EXITxx to PROGxx using the IEFEXPR REXX exec provided by IBM. For example, the following statement in EXITxx

```
EXIT EXITNAME(xx)MODNAME(yy)
```

would look like this in PROGxx:

```
EXIT ADD EXITNAME(xx) MODNAME(yy)
```

For information about how to use the IEFEXPR REXX exec to convert the exit definitions in EXITxx to equivalent definitions in PROGxx, see Chapter 41, "IEAAPFxx (authorized program facility list)," on page 371.

For information about how to use PROGxx to define and control the use of exits, see PROG.

Parameter in IEASYSxx (or issued by the operator)

EXIT= aa
The two alphanumeric characters, represented by aa are appended to EXIT to identify the EXITxx member of parmlib.

Syntax rules for EXITxx

The following syntax rules apply to EXITxx:

- Use columns 1 through 71. Do not use columns 72 through 80 for data; these columns are ignored.
- At least one delimiter (space or comma) is required between a statement and keyword. Delimiters are not required between keywords.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

Syntax format of EXITxx

```
EXIT  EXITNAME(exitname)
      MODNAME(module name)
```

IBM-supplied default for EXITxx

There is no default EXITxx parmlib member supplied by IBM. A sample parmlib member, EXIT00, is provided in SYS1.SAMPLIB.

Statements/parameters for EXITxx

EXIT
Specifies installation exit information for a single exit point and exit routine name. You must specify an EXIT statement for each installation exit that you want to get control.

EXITNAME
This required parameter specifies the name of an installation exit point. Valid values are:

- IEF_VOLUME_ENQ for the volume ENQ installation exit
- IEF_VOLUME_MNT for the volume mount installation exit
- IEF_SPEC_WAIT for the specific waits installation exit
- IEF_ALLC_OFFLN for the allocated/offline device installation exit.

See *z/OS MVS Installation Exits* for information about the individual exits.

MODNAME
This required parameter specifies the exit's load module name. The exit routine name can be any 1-8 character string of alphanumeric or special (#, @, or $) characters. The first character of the routine name cannot be a number.

Chapter 33. EXSPATxx (excessive spin condition actions)

The EXSPATxx member allows an installation to specify actions to be taken to recover from excessive spin conditions without operator involvement. A system routine might be spinning because it cannot obtain a resource held by another processor. EXSPATxx allows you to specify the action or actions to be taken to end the spin loop if the excessive spin is detected while spinning for one of the following:

- A RISGNL response
- A spin lock to be released
- A successful BIND BREAK completion
- The RESTART resource
- An address space to quiesce
- An INTSECT release
- A CPU in the stopped state.

EXSPATxx does not support actions for spin loops that are caused by SIGP failures.

The IBM-supplied defaults for the system are appropriate in a normal environment. In a test environment, however, you might want to specify particular actions to be taken.

In addition to recovery actions, EXSPATxx allows you to specify a timeout interval. If an excessive spin continues beyond the timeout interval, the system automatically issues a SPIN action before taking any recovery action specified in an EXSPATxx member. If the excessive spin continues until the timeout interval is reached again, then the recovery actions specified in the EXSPATxx member are taken.

For more information about handling excessive spin conditions, see *z/OS MVS Setting Up a Sysplex*

Parameter in IEASYSxx (or issued by the operator)

There is no IEASYSxx parameter to set EXS=xx. It can be set only by the SET command.

SET EXS=xx

The two alphanumeric characters (**xx**) indicate the EXSPATxx member that contains the excessive spin recovery actions and the excessive spin loop timeout interval.

Syntax rules for EXSPATxx

The following syntax rules apply to EXSPATxx:

- SPINRCVY or SPINTIME can start in any column between column 1 and 71.
- SPINRCVY or SPINTIME must be the first entry on a record.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- Any number of blanks can exist between SPINRCVY and the first action.

- Multiple actions on a SPINRCVY statement must be separated by commas. A blank is not a valid delimiter between two actions.
- SPINRCVY and multiple actions must appear on the same logical record.
- SPINTIME=*sss* cannot contain blanks.
- The system ignores columns 72 through 80.

Syntax example of EXSPATxx

An example of the syntax for an EXSPATxx member is:

```
SPINRCVY   action[,action]...
SPINTIME=sss
```

IBM-supplied default for EXSPATxx

None.

The EXSPATxx member is optional. At initialization time, the IBM-supplied default recovery actions are taken for excessive spin conditions. After initialization, you can issue the SET EXS=xx command to specify the EXSPATxx member. Two alphanumeric characters are appended to EXSPAT to form the name of the EXSPATxx member.

The IBM-supplied default recovery actions for excessive spin conditions are:

```
SPINRCVY ABEND,TERM,ACR (see note 1)
SPINTIME=10 or SPINTIME=40 (see note 2)
```

Note:

1. Since the system will automatically issue a SPIN action before taking any recovery action specified in an EXSPATxx member, this specification will actually result in actions of SPIN, ABEND, TERM, ACR, which are the same actions that would be taken by the system if SPINRCVY were not specified at all.
2. The default spin loop timeout interval depends on the system configuration. For MVS running in a native or basic mode, or in an LPAR with dedicated CPs, the default is 10 seconds. For MVS running on VM or in an LPAR with shared CPs, the default spin loop timeout interval is 40 seconds.

Statements and parameters for EXSPATxx

SPINRCVY

Specifies the actions that the system is to take to end a spin loop that is not caused by a SIGP failure. SPINRCVY must be the first entry on a record defining these actions. At least one action is required on a SPINRCVY statement. You can specify a maximum of eight parameters on a SPINRCVY statement.

ABEND

The system abnormally ends the current unit of work on the processor that is causing the excessive spin (abend code X'071') and takes a SLIP dump if the installation is running with the IBM-supplied default for IEASLPxx. Recovery routines for the unit of work are allowed to retry.

ACR

The system invokes alternate CPU recovery (ACR) processing for the processor causing the excessive spin. The processor is taken offline.

OPER
The system issues message IEE331A and processes the operator's reply. If the message could not be issued, or if the operator does not respond, the processor that is in an excessive spin is put into a restartable wait state (X'09n'). The operator can respond to the message with ABEND, ACR, TERM, SPIN, or U (equivalent to SPIN) to continue.

SPIN
Specifies that the spinning processor is to continue spinning. An informational message (IEE178I) is displayed.

TERM
The system abnormally ends the current unit of work on the processor that is causing the excessive spin (abend code X'071') and takes a SLIP dump if the installation is running with the IBM-supplied default for IEASLPxx. Recovery routines for the unit of work are not allowed to retry.

SPINTIME=*sss*
Specifies the excessive spin loop timeout interval, where *sss* is the number of seconds. *sss* can be one through three digits but must be at least 10 seconds. For MVS running in a native or basic mode, or in an LPAR with dedicated CPs, the default is 10 seconds. For MVS running on VM or in an LPAR with shared CPs, the default spin loop timeout interval is 40 seconds.

Default value: 10 seconds or 40 seconds

Notes:

1. In a PR/SM environment, system configuration and weighting affect the amount of processing (CPU cycles) that occurs in each partition during the timeout interval.
2. The specified SPINTIME value affects the default value of the INTERVAL parameter in the COUPLExx member of SYS1.PARMLIB. If the excessive spin parameters are not explicitly set in an EXSPATxx parmlib member, the IBM supplied default spin parameters are used: the derived spin failure detection interval can be 45 seconds or 165 seconds depending on the configuration. See the INTERVAL parameter in the COUPLExx member for more information.
3. For an MVS system that is running in a non-dedicated PR/SM environment or under VM, if you specify non-default values for SPINTIME, consider the following facts:
   - The more processing power (CPU cycles) available to the system, the less time it needs to resolve a spin loop.
   - The less processing power available to the system, the more time it needs to resolve a spin loop.
   - Specifying a SPINTIME that is too low might cause premature excessive spin conditions.

   If your system is running in a logical partition that is sharing central processing resources, and you want to use a SPINTIME value of less than the default time (40 seconds), the BLWSPINR member of SYS1.SAMPLIB contains an exec that can help you evaluate your spintime requirements.

Example of EXSPATxx

If you specify more than one action, the system executes the actions in the order of their specification. For example, if you code

```
SPINRCVY      ABEND,ACR
SPINTIME=30
```

the system allows the excessive spin to continue for about 30 seconds and then issues a SPIN action before your first recovery action, issuing an informational message and allowing the excessive spin to continue for another 30 seconds. The system then issues ABEND 071 to end the unit of work that is causing the spin and takes a SLIP dump if the installation is running with the IBM-supplied default for IEASLPxx. The recovery routines for that unit of work try to recover; if they fail, the spin is allowed to continue for 30 seconds. Finally, the system initiates ACR (alternate CPU recovery) to take the processor that is causing the spin offline.

Note: After the processor is taken offline, it can be brought back online by the operator through the CF CPU(x),ONLINE command.

Chapter 34. GRSCNFxx (global resource serialization configuration)

The contents of the GRSCNFxx parmlib member are used during system initialization to define the attributes of global resource serialization processing.

A *global resource serialization complex* consists of two or more systems connected by communication links. The systems in the complex use global resource serialization to serialize any shared resources such as controlling access to data sets on shared DASD volumes at the data set level rather than at the volume level. (For more information, see *z/OS MVS Planning: Global Resource Serialization*.)

The majority of GRSCNFxx parameters are specific to defining a global resource serialization ring complex. However, GRSQ is specific to star mode, and the following keywords are relevant for all modes, including GRS=NONE:

- SYNCHRES
- ENQMAXA
- ENQMAXU
- AUTHQLVL
- MONITOR

To define a global resource serialization ring complex, use the GRSCNFxx parmlib member. GRSCNFxx contains a GRSDEF statement for each system. The statement identifies:

- The name of the system in the complex, by means of the MATCHSYS parameter.
- The CTC links assigned to the system, by means of the CTC parameter.
- The minimum amount of time that the RSA-message is to spend in the system, by means of the RESMIL parameter.
- Whether or not the system identified by the MATCHSYS keyword has the automatic restart capability, by means of the RESTART parameter.
- Whether or not the system identified by the MATCHSYS keyword has the capability to automatically rejoin the complex, by means of the REJOIN parameter.
- The maximum amount of time (the tolerance time interval) before global resource serialization will detect a timeout, by means of the TOLINT parameter.
- The method to establish the global resource serialization ring acceleration function, by means of the ACCELSYS parameter.
- The member of parmlib that contains the default global resource serialization component tracing options, by means of the CTRACE parameter.

There are three basic ways to use GRSCNFxx to define your complex:

1. You can create one GRSCNFxx parmlib member that contains GRSDEF statements that define all of the systems in the complex. After creating the member, copy it to the parmlib on each system. As each system is initialized, it reads the GRSCNFxx member, locates its own GRSDEF statement, and uses the information in the statement during initialization.
2. You can create a unique GRSCNFxx member for each system in the complex. The member consists of a GRSDEF statement for that particular system. You

place the unique member in parmlib on that particular system, and that system uses the information it contains during initialization.

3. If all systems in the global resource serialization complex will belong to the same sysplex, you can create one GRSCNFxx member that contains a single GRSDEF statement that defines consistent information for all systems. The IBM-supplied default member, GRSCNF00, can be used for this purpose.

Variations or combinations of these methods are, of course, possible. Which ever method you choose, GRSCNFxx provides information about the configuration of the complex that global resource serialization requires during initialization. Other information that global resource serialization requires comes from the following system parameters (in IEASYSxx):

- GRS=
- GRSCNF=
- GRSRNL=
- SYSNAME=

The GRS, GRSCNF, GRSRNL, and SYSNAME parameters can only be specified at IPL time, either in IEASYSxx or by the operator. They all remain in effect for the duration of the IPL. For specific details about how to specify any one of these parameters, see the description of the parameter provided in Chapter 54, "IEASYSxx (system parameter list)," on page 431.

During initialization processing for a system that is to join an existing complex, global resource serialization verifies that the information in GRSCNFxx is consistent with the existing complex. Even if a system is defined in a GRSCNFxx parmlib member and initialized with GRS=JOIN, it cannot join the complex unless either:

- at least one CTC link associated with the system in GRSCNFxx is attached to another active system in the complex, or
- the system is joining a sysplex (see Chapter 24, "COUPLExx (cross-system coupling facility (XCF) parameters)," on page 263)

Parameters in IEASYSxx

```
GRS=            {JOIN      }
                {START     }
                {TRYJOIN   }
                {NONE      }
                {STAR      }
GRSCNF=xx
GRSRNL=         {xx        }
                {(xx,yy... }
                {EXCLUDE   }
SYSNAME=name
```

Syntax rules for GRSCNFxx

The following rules apply to the creation of GRSCNFxx:

- Use columns 1 through 71; columns 72 through 80 are ignored.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". A comment can span lines and can appear anywhere except within a keyword or a specified value.

- Each GRSDEF statement is defined as beginning with the characters "GRSDEF" and ending with the character immediately preceding the next GRSDEF statement.
- The GRSDEF statement format is:

```
GRSDEF     [MATCHSYS     {(name)                       }]
           [             {(*)                          }]
           [RESMIL       {(1-8 decimal digit)          }]
           [             {(OFF)                        }]
           [CTC (3-4 hexadecimal digit device number)...]
           [RESTART      {(YES)                        }]
           [             {(NO)                         }]
           [REJOIN       {(YES)                        }]
           [             {(NO)                         }]
           [TOLINT (1-8 decimal digit)                  ]
           [ACCELSYS (1-2 decimal digit)                ]
           [CTRACE (parmlib member name)                ]
           [SYNCHRES     {(YES)                        }]
           [             {(NO)                         }]
           [GRSQ         {(ALL)                        }]
           [             {(CONTENTION)                 }]
           [             {(LOCAL)                      }]
           [ ENQMAXA(1-8 decimal digit)]
           [ ENQMAXU(1-8 decimal digit)]
           [ AUTHQLVL(decimal digit 1 or 2)]
           [MONITOR     {(YES)                       }]
           [             {(NO)                        }]
```

Note:

1. You can put a blank anywhere except within a keyword or a specified value.
2. You can use as many lines as you need for one GRSDEF statement.
3. You can specify multiple parameters on the same line.
4. No delimiters or blanks are needed between parameters. For example, the following GRSDEF statement is valid:

   ```
   GRSDEF MATCHSYS(*) CTC(9A0)CTC(8B0)CTC(7D0)RESMIL(10)
   ```

5. Duplicate GRSDEF statements cause a syntax error. A syntax error also occurs if you specify more than one GRSDEF statement with MATCHSYS(*), either explicitly or by default, or more than one GRSDEF statement with the same MATCHSYS name.
6. If this system is not a member of a sysplex, each GRSDEF statement must include at least one CTC parameter. Each GRSDEF statement can include up to a maximum of 64 CTC parameters.

IBM-supplied default for GRSCNFxx

Default member GRSCNF00 contains:

```
GRSDEF
       RESMIL(10)
       TOLINT(180)
       ACCELSYS(99)
```

Statements and parameters for GRSCNFxx

ACCELSYS(num)

Specifies the threshold for global resource serialization ring acceleration and the number of systems that must see a resource request before it is granted. To use ring acceleration, every system must have a link to every other system. Ring acceleration requires a fully connected complex, and it also

requires an alternate link for each connection not in the sysplex. When the ACCELSYS keyword is not specified, ring acceleration is turned off. The maximum performance benefit is provided when the ACCELSYS threshold value is 2.

Specifying a different ACCELSYS value for different systems in a complex is allowed, but the complex uses the highest ACCELSYS value specified to determine when to send the shoulder-tap acknowledgment. If the highest value is greater than the number of systems in the ring, the shoulder-tap acknowledgment does not take place. To ensure that shoulder-tap processing occurs, specify ACCELSYS(num) on all systems in the ring, making sure that the value of *num* is less than the number of systems in the ring.

Value range: The ACCELSYS threshold value range is 2 - 99.

Default: 99

CTC(device number)
Identifies the device number of a CTC link that is attached to this system that is dedicated to the use of global resource serialization. Only one device number can be specified on each CTC parameter, but up to 64 different CTC parameters can be specified. If this system is a member of a sysplex, it is not necessary to specify the CTC parameter unless this system must have a link to a system outside of the sysplex.

Value range: 3 or 4 hexadecimal digits. The device number cannot be preceded by a slash.

Default: None

CTRACE(member-name)
Specifies the member of SYS1.PARMLIB that contains the default global resource serialization component tracing options.

Value range: The 8-character member name must be in the format CTnGRSxx, where:

n An alphanumeric character to specify the source of the member. IBM-supplied members use I.

xx Any two characters to identify the member.

Default: CTIGRS00

The contents of CTIGRS00 are as follows:

```
TRACEOPTS
      OFF
      OPTIONS('CONTROL','MONITOR')
      BUFSIZE(16M)
```

For more information about specifying options for the global resource serialization component trace, see *z/OS MVS Diagnosis: Tools and Service Aids* . For information about specifying the component trace member of SYS1.PARMLIB, see Chapter 26, “CTncccxx (component trace parameters),” on page 283.

GRSQ{(ALL)}
{(CONTENTION)}
{(LOCAL)}
The GRSQ keyword is only relevant for star mode. It is ignored for ring and

none mode. When the SDATA=GRSQ option is specified, the GRSQ keyword collects global resource serialization information during SYSMDUMP and SVC dump processing. ENQ resource information from all systems in the global resource serialization complex is gathered and placed into the dump. This can be time consuming when running in star mode because global resource serialization needs to collect and merge information from all systems in the sysplex. The time it takes increases as more systems are added to the sysplex or the number of concurrent ENQs increase, or both. The GRSQ keyword allows the installation to limit the data collected and therefore lessen the time that is needed to complete the dump.

- GRSQ(CONTENTION) causes GRSQ processing to collect all ENQ resources on the local system, and only collect information about global resources in contention in the rest of the GRS complex.
- GRSQ(LOCAL) causes GRSQ processing to collect all ENQ resources on the local system only. This is not a recommended value and should only be used as a circumvention. The GQSCAN that is issued does not collect data from the other systems.
- GRSQ(ALL) causes GRSQ processing to collect all ENQ resources from all systems in the entire GRS complex. Specifying GRSQ(CONTENTION) adds WAITCNT=1.

Default: CONTENTION

MATCHSYS{(name)}
{(*) }

Identifies the system that is defined in the GRSDEF statement. A system that is being initialized compares the value that is specified on the SYSNAME system parameter to the value specified for MATCHSYS to locate its GRSDEF statement in GRSCNFxx. Omitting MATCHSYS or specifying MATCHSYS(*) is regarded as a match unless a GRSDEF statement exists with an explicit name that matches the SYSNAME value. Including both a GRSDEF statement with an explicit MATCHSYS name and a GRSDEF statement with MATCHSYS(*), either explicitly or by default, thus does not cause a syntax error; the system ignores the GRSDEF statement with MATCHSYS(*).

Global resource serialization initialization processing searches the statements in GRSCNFxx until it finds a GRSDEF statement that matches the SYSNAME value, then uses the information in the statement to continue initialization. It issues an error message if it finds no match.

Value range: 1 - 8 alphanumeric characters, including the @,#, and $ characters.

Default: *

MONITOR{(YES)|(NO)}

Indicates whether the system defined in the GRSCNFxx MATCHSYS parameter is to have GRS SMF 87 monitoring activated.

Default: NO

Note: See in *z/OS MVS Planning: Global Resource Serialization* for more information.

REJOIN{(YES)|(NO)}

Indicates whether the system defined in the MATCHSYS parameter can automatically rejoin the active ring, after that system was stopped, then

started. Specifying REJOIN(YES) eliminates the need for the operator to issue a VARY GRS, RESTART command to have the system join the active ring.

Note: If this system is a member of a sysplex, regardless of what you specify, REJOIN is set to REJOIN(YES).

Default: YES

RESMIL(number|OFF)
Specifies the RSA-message residency time.

A value, if specified, indicates the minimum RSA-message residency time in milliseconds, (that is, the least amount of time that the RSA-message is to spend in this system). The actual amount of time that the RSA-message will spend in this system will vary between the time you specify in milliseconds and a maximum value calculated by global resource serialization. In this way, global resource serialization balances CPU utilization and ENQ response time. If you omit RESMIL, then the default for the RSA-message residency time is 10 milliseconds.

OFF, if specified, indicates that the RSA-message residency time is zero and that global resource serialization does no tuning. Note that RESMIL(OFF) differs from RESMIL(0). When you specify RESMIL(0), the system tunes the residency time in a range with a minimum of zero.

Value range: 0 to 99999999 milliseconds

Default: 10 milliseconds

RESTART{(YES)|(NO)}
Indicates whether the system defined in the GRSCNFxx MATCHSYS parameter can automatically rebuild a disrupted ring. A disrupted ring occurs when the RSA-message stops moving because of a system failure or a link failure.

Note: If this system is a member of a sysplex, regardless of what you specify, RESTART is set to RESTART(YES).

Default: YES

TOLINT(num)
Specifies, in seconds, the maximum tolerance time interval global resource serialization allows the RSA-message to return to that system, before it considers the RSA-message overdue.

When the TOLINT value specified is not valid, then message ISG008E is issued, the TOLINT keyword is ignored, a default of 3 minutes (180 seconds) is used, and the IPL continues.

Value range: The TOLINT value that is specified must be greater than zero (0), and less than 86000 seconds (24 hours).

Default: 180 seconds (3 minutes)

SYNCHRES{(YES)|(NO)}
Indicates whether the system defined in the GRSCNFxx MATCHSYS parameter is to have synchronous reserve processing activated.

Default: YES

Note: If SYNCHRES=YES, but GRS cannot complete the I/O to reserve a device, message ISG363I is issued. The request continues as if SYNCHRES=NO.

[ENQMAXA(value)]
[ENQMAXU(value)]
Identifies the system-wide maximum of concurrent ENQ requests for authorized (ENQMAXA) or unauthorized (ENQMAXU) requesters.

The ENQMAXA range is 250,000 to 99,999,999. The default MAXVALUE is 250,000.

The ENQMAXU range is 16,384 to 99,999,999. The default MAXVALUE is 16,384.See *z/OS MVS Planning: Global Resource Serialization* for information about how concurrent ENQ request counts are updated.

[AUTHQLVL(value)]
Indicates the authorized qname level. GRS manages a list of authorized qnames for processing ENQ and DEQ names. If an unauthorized requester attempts to issue an ENQ or DEQ for an authorized qname, GRS fails the request. Although authorized requesters are not required to use qnames from the authorized list, IBM strongly suggests that they use these qnames to provide protection against "denial-of-service attacks", where an unauthorized requester might obtain a critical ENQ resource and not release it.

As of z/OS V1R13 IBM recognizes two authorized qname lists. The default list (AUTHQLVL=1) contains the qnames for many IBM products or services that uses them. IBM recognizes a second authorized qname list (AUTHQLVL=2) that includes additional qnames.

For complete information about the contents of these qname lists and ENQ and DEQ processing for authorized qnamess, see "Authorized qnames" in *z/OS MVS Planning: Global Resource Serialization*.

Use the SETGRS AUTHQLVL to dynamically change the value of the level from 2 to 1.

Use the DISPLAY command to return information about the AUTHQLVL value in effect.

For command syntax information, see *z/OS MVS System Commands*.

Value range: Either 1 or 2.

Default: 1

Chapter 35. GRSRNLxx (global resource serialization resource name lists)

GRSRNLxx consists of three resource name lists (RNLs). When a global resource serialization complex is active, the system uses these RNLs to determine how to treat the resources defined in the RNLs. The RNLs are:

- The **SYSTEM inclusion RNL**. The system treats each resource named in this RNL as a global resource if an ENQ or a DEQ macro instruction for the resource specifies a scope of SYSTEM.
- The **SYSTEMS exclusion RNL**. The system treats each resource named in this RNL as a local resource if an ENQ, DEQ, or RESERVE macro instruction for the resource specifies a scope of SYSTEMS.
- The **RESERVE conversion RNL**. The system suppresses a hardware reserve for each resource named in this RNL if a RESERVE macro instruction requests the use of the resource.

Note: Specifying RNL=NO on the ENQ macro will bypass all RNL processing. See *z/OS MVS Programming: Assembler Services Guide* for information about the ENQ macro.

Use the SET GRSRNL command to change the RNL list while the system is active. For more information, see *z/OS MVS System Commands*.

See *z/OS MVS Planning: Global Resource Serialization* for detailed information about global resource serialization, RNLs, and the RNL syntax checker. The RNL syntax checker lets you check the syntax of GRSRNL parameters specified on the JCL and resource names specified in GRSRNLxx.

Parameter in IEASYSxx (or supplied by the operator)

```
GRSRNL=  {aa        }
         {(aa,bb...)}
         {EXCLUDE   }
```

The two alphanumeric characters, represented by aa (or bb, and so forth), are appended to GRSRNL to form the name of the GRSRNLxx member(s). If GRSRNL=EXCLUDE is specified, no RNLs are to be used in the complex and all resources are treated as local.

Support for system symbols

Any static symbols used must be consistent across the GRS complex. If the symbols used in the RNL members resolve to different values, the RNL specifications for each system will differ. GRS requires that the RNL specification used for system initialization matches the current specification for the rest of the complex. If the RNL specifications do not match, GRS puts the initializing system into a non-restartable wait state.

Syntax rules for GRSRNLxx

The following rules apply to the creation of GRSRNLxx:

- Use columns 1 through 71; columns 72 through 80 are ignored. Note that if you need to continue to another line when specifying a value (such as the name on the RNAME parameter), you must start the continuation in column 1.
- Comments may appear in columns 1-71 and must begin with /* and end with = */. A comment can span lines and can appear anywhere except within a keyword or a specified value.
- Each RNLDEF statement is defined as beginning with the characters RNLDEF" nd ending with the character immediately preceding the next RNLDEF statement.
- Each RNLDEF statement must contain, in any order, the RNL, TYPE, QNAME, and RNAME parameters. For example:

```
RNLDEF RNL(INCL) TYPE(SPECIFIC) QNAME(SYSDTNM)
       RNAME(SYS1.USR)

RNLDEF RNL(INCL) TYPE(GENERIC) QNAME(SYSDTNM)
       RNAME(SYS1.U)

RNLDEF RNAME(SYS1.U) TYPE(GENERIC) RNL(INCL)
       QNAME(SYSDTNM)
```

  Exceptions to this are:

  – The RNAME parameter can be omitted if TYPE(GENERIC) is specified for a generic QNAME resource. For example:

    ```
    RNLDEF TYPE(GENERIC) QNAME(X'1A4A783F2B') RNL(EXCL)
    ```

  – A null GRSRNLxx parmlib member is valid. The system does not consider a member in error if it is empty; that is, if it contains neither RNLDEF statements nor comments.

- A specific resource name entry matches a search argument only when they are the same.

  A match occurs only when the length of the RNAME value on the RNLDEF statement is the same as the length of the RNAME value that is coded on the ISGENQ, ENQ, or RESERVE macro, and only when the RNAME value on the RNLDEF statement is the same as RNAME value coded on the ISGENQ, ENQ, or RESERVE macro. For example:

  Assume that an RNLDEF statement is coded as follows to add an entry in the Systems Exclusion List:

  ```
  RNLDEF RNL(EXCL) TYPE(SPECIFIC) QNAME(MYQNAME) RNAME(ABC)
  ```

  Further assume that an application program is coded to issues an ENQ and defines the length of the RNAME as 44 bytes with blanks filled to the right of 'ABC':

  ```
  ENQ (QNAM,RNAM,E,L'RNAM),MF=(E,XENQ)
  .
  .
  .
  RNAM   DC  CL44'ABC'
  ```

  In this case, the length of the RNAME value on the RNLDEF statement does not match the length of the RNAME value that is coded on ENQ macro. You can correct this by taking one of the following actions:

  – Change the RNLDEF statement to specify the correct RNAME length and value:

```
RNLDEF RNL(EXCL)TYPE(SPECIFIC)QNAME(MYQNAME)RNAME(X'C1C2C34040404040....40')
[not all blanks are shown]
```

– Change the RNLDEF statement from a specific to a generic type:

```
RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(MYQNAME) RNAME(ABC)
```

– Change the RNLDEF statement from a specific to a pattern type using any one of the following variations:

```
RNLDEF RNL(EXCL) TYPE(PATTERN) QNAME(MYQNAME) RNAME(*)
RNLDEF RNL(EXCL) TYPE(PATTERN) QNAME(MYQNAME) RNAME(ABC*)
```

– Change the application program to eliminate the blank fill:

```
ENQ   (QNAM,RNAM,E,L'RNAM),MF=(E,XENQ)
.
.
.
RNAM     DC    CL3'ABC'
```

Change the length of the RNAME, if the length is hardcoded on the ENQ macro.

Note:

1. You can put a blank anywhere except within a keyword or a specified value.
2. You can use as many lines as you need for one RNLDEF statement.
3. You can specify multiple parameters on the same line.
4. You do not need to put a blank (or other delimiter) between parameters. For example, the following RNLDEF statement is valid:

   ```
   RNLDEF RNL(EXCL)TYPE(SPECIFIC)QNAME(SYSDSN)RNAME(SYS1.BRODCAST)
   ```

5. You can use from 1 to 8 characters for the name that you specify on the QNAME parameter. You must enclose the name in parentheses.
6. You can use from 1 to 255 characters for the name that you specify on the RNAME parameter. You must enclose the name in parentheses. Any complete line of blanks (columns 1-71) will be ignored when specifying a value for the RNAME.
7. You can specify the names on the QNAME and RNAME parameters in any of the following formats:
   - If the name contains characters that cannot display, you must use two hexadecimal digits to specify each character of the name. For example:

     ```
     RNLDEF RNL(INCL) TYPE(GENERIC) QNAME(X'18')
            RNAME(X'19')
     ```

   - If the name contains displayable characters that are alphanumeric (A-Z and 0-9), *, ?, #, @, and $, or a period (.), enter the name as is. or example:

     ```
     RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(STW@7)
            RNAME(REW.20)
     ```

   - If the name contains displayable characters other than those already described (including a blank, but excluding a single quotation mark), enclose the name in single quotation marks. For example:

     ```
     RNLDEF RNL(CON) TYPE(SPECIFIC) QNAME('$ ( )*')
            RNAME('A B')
     ```

IBM-supplied default for GRSRNLxx

IBM provides default RNL statements in GRSRNL00 (the default member). See *z/OS MVS Planning: Global Resource Serialization* for the default statements and their meaning.

Statements/parameters for GRSRNLxx

RNL {(INCL)}
{(EXCL)}
{(CON) }

Specifies the RNL in which the resource name entry is to be placed, where INCL indicates the SYSTEM inclusion RNL, EXCL indicates the SYSTEMS exclusion RNL, and CON indicates the RESERVE conversion RNL.

The RNLs on all systems in the complex must be the same. That means that every RNL on each system must contain the same resource name entries, and that these resource name entries must appear in the same order.

TYPE {(GENERIC) }
{(SPECIFIC)}
{(PATTERN)}

Specifies the type of resource name entry being defined in the RNL. For a SPECIFIC entry, you must specify a QNAME (major name) and an RNAME (minor name) for the resource.

PATTERN indicates that the resource name in the RNL entry is a pattern that must fit the resource name specified on the ENQ request. You can use wildcard characters (* or ?) in either part of the resource name.

***** Allows matching for a substring of any characters for any length, including zero.

? Allows matching for any single character.

For example:

```
RNLDEF RNL(EXCL) TYPE(PATTERN)
QNAME(SYSDSN)
RNAME(SYS1.*.PARMLIB)
```

You can convert all hardware RESERVEs to global ENQ with the following specification:

```
RNLDEF RNL(CON) TYPE(PATTERN)
QNAME(*)
```

Note: This is recommended only on systems using GRS Star.

`QNAME(name)`

Specifies the major name of the resource.

`RNAME(name)`

Specifies the minor name of the resource.

Chapter 36. GTFPARM (generalized trace facility parameters)

GTFPARM provides default or installation-defined trace options to control the generalized trace facility (GTF). The member is read only when the operator (or an automatic command) issues START GTF. GTFPARM is not used during system initialization.

The member name on the START command can be the same as the IBM-supplied cataloged procedure, GTF. The PROC statement of that procedure identifies GTFPARM as the member from which GTF will get its trace parameters. If the installation wants to place the GTFPARM member in a data set other than SYS1.PARMLIB, specify the alternate data set in the SYSLIB DD statement and then specify a member from that PDS using the MEMBER keyword. If the installation wants to substitute another member in place of GTFPARM, the operator may enter the replacement member name on the START command with the MEMBER keyword.

The IBM procedure, GTF, as supplied in SYS1.PROCLIB, contains these statements:

```
//GTFNEW PROC        MEMBER=GTFPARM
//IEFPROC EXEC       PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,
                     TIME=YES',TIME=1440,REGION=2880K
//IEFRDER DD         DSNAME=SYS1.TRACE,UNIT=SYSDA,
//                   SPACE=(TRK,20),DISP=(NEW,KEEP)
//SYSLIB DD          DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

For further analysis of this procedure and additional information about GTF trace options, see *z/OS MVS Diagnosis: Tools and Service Aids* or the GTF topics in *z/OS MVS Diagnosis: Tools and Service Aids*.

Because default options in GTFPARM specify minimal data for only a limited number of traced events, you may wish to tailor GTF to trace specific events for problem determination purposes through one of the following methods:

- Specify another SYS1.PARMLIB member name, using the MEMBER keyword on the START command.
- Change the trace options in GTFPARM.
- Change the SYSLIB DD statement of the IBM procedure to specify a parmlib member that has the options you want.
- Retain the IBM procedure to handle default options, and write one or more alternate procedures, each specifying a different alternate parmlib member. You could design each member to contain GTF options useful under particular circumstances. Instruct the operator when to issue the START command for each member name.

GTF tries to read parameters from the specified parmlib member. If an error occurs in opening or reading the member, or if GTF detects a syntax error, it writes a diagnostic message to the operator, and requesting that SPECIFY TRACE OPTIONS be used, as if no GTF parmlib member were available. The operator therefore must have a complete list of desired GTF parameters available when he starts GTF.

Parameter in IEASYSxx (or issued by the operator)

None.

Syntax rules for GTFPARM

The following rules apply to the creation of a GTF parmlib member:

- Specify the TRACE keyword and its main options only on the first record. Do not place them on subsequent records. For example,

  ```
  Record #1:      TRACE=IOP,SVCP,SSCH
  ```

  This example requests the tracing of specific I/O interrupts, specific SVC interrupts, and all start subchannel and resume subchannel operations.
- The second and subsequent records should contain only "prompting" keywords, such as IO= or SVC=. These keywords provide for detailed operands that indicate which I/O interrupts or which SVC interrupts should be traced. For example, the IOP and SVCP keywords in the Record #1 example (above) must be followed by prompting records that name specific device numbers and specific SVC numbers for which interrupts should be traced. As an example,

  ```
  Record #2:     IO=(191,192,102A),SVC=(1,2,3)
  ```

  If the specific operands of any prompting keyword are missing, GTF does not prompt the operator. It accepts a general specification. For example, if IOP is specified in Record #1, and Record #2 specifies only SVC=(1,2,3), and no particular device numbers are specified for I/O tracing, GTF assumes that tracing of I/O interrupts is desired for all devices.

  When all of the records have been read, GTF issues message AHL103I TRACE OPTIONS SELECTED -- IO,SVC=(1,2,3). The operator can then respond to message AHL125A RESPECIFY TRACE OPTIONS OR REPLY U by either entering all the desired options or accepting the input that was specified in the parmlib member.
- An END keyword or an end-of-file must follow all prompting keywords. If the END keyword is used, it must appear either on the last record supplying prompting keywords or on its own record.
- Lines that begin with an asterisk in column 1 are comments.
- If you need to specify additional operands for the same keyword, restate the keyword and the additional operands in a subsequent prompting record. The previous examples, expanded to include additional SVC numbers and an END keyword, would appear like this:

  ```
  Record #1:      TRACE=IOP,SVCP,SSCH

  Record #2:      IO=(191,192,102A),SVC=(1,2,3)

  Record #3:      SVC=(4,5,6,7,8,9,10),END
  ```
- Certain trace options do not work in combination with others. Table 20 shows those trace options that should not be specified together. If you specify two or more options from the same horizontal row, GTF uses the option in the lowest numbered column and ignores the other options. For example, if you specify both SYSP and SVC (row C), GTF uses SYSP and ignores SVC.

Table 20. Combining certain GTFPARM options

	1	2	3	4	5
A	SYSM	SYSP	SYS	SSCHP	SSCH
B	SYSM	SYSP	SYS	IOP	IO

Table 20. Combining certain GTFPARM options (continued)

	1	2	3	4	5
C	SYSM	SYSP	SYS	SVCP	SVC
D	SYSM	SYSP	SYS	PIP	PI
E	SYSM	SYSP	SYS	EXT	
F	SYSM	SYSP	SYS	RR	
G	SYSM	SYSP	SYS	CSCH	
H	SYSM	SYSP	SYS	HSCH	
I	SYSM	SYSP	SYS	MSCH	
J	SYSM	SYSP	SYS	XSCH	
K	URSP	USR			
L	CCWP	CCW			

IBM-supplied default for GTFPARM

When GTF is started by specifying the IBM-supplied cataloged procedure, the following options exist in GTFPARM:

```
TRACE=SYSM,USR,TRC,DSP,PCI,SRM
```

These keywords cause the following events to be recorded: SVC interruptions, I/O interruptions, program-controlled I/O (PCI) interruptions, program interruptions, external interruptions, dispatcher executions, start subchannel and resume subchannel operations, clear subchannel operations, halt subchannel operations, modify subchannel operations, entries to the system resource manager, entries to recovery routines, events associated with GTF (TRC), and data passed to GTF via the GTRACE macro (USR). All keywords except USR result in minimal format trace entries. USR entries are the length specified by the user in the GTRACE macro. Such entries may optionally be a maximum of 8192 bytes, excluding the prefix.

Note: If you use the default options provided, be aware that some data will be duplicated by the system trace if it is executing concurrently with GTF.

Statements/parameters for GTFPARM

Only the options provided in the IBM default member, GTFPARM, are described here. For a complete list and description of GTF options, see *z/OS MVS Diagnosis: Tools and Service Aids*.

DSP

This parameter requests recording for all dispatchable units of work (SRB, LSR, TCB, and SVC prologue dispatch events). The option is not included in the specification of SYS or SYSM. It must be specified in addition to other parameters. The parameter produces comprehensive format except when SYSM is also specified. With SYSM, data is in minimal format.

END

This parameter indicates the end of the prompting records. In the parmlib member, an end of file serves the same purpose. Never include the END parameter in the first record (the record that contains the TRACE parameters) because GTF regards such an occurrence as an error.

PCI
This parameter requests that program-controlled I/O interruptions (PCIs) be recorded in the same format as other requested I/O trace records. If specific device numbers are specified through prompting records, program controlled I/O interruptions are recorded for the specified devices. I/O tracing must be requested because PCI must be specified with a GTF parameter that causes IO, such as SYS or SYSM.

SRM
This parameter requests a trace entry each time that the system resource manager (SRM) is invoked. The option is not included in the specification of SYS or SYSM. It must be specified in addition to other parameters. Data is in comprehensive format except when SYSM is also specified. Comprehensive format includes the jobname.

SYSM
This parameter requests recording of minimal trace data for all external interruptions (EXT), program interruptions (PI), recovery routines (RR), and supervisor call interruptions (SVC). SYSM causes recording of all I/O interruption (IO), start subchannel and resume subchannel operations (SSCH), clear subchannel operations (CSCH), halt subchannel operations (HSCH), modify subchannel operations (MSCH), and cancel subchannel operations (XSCH). When you specify DSP, RNIO, or SRM, in addition to SYSM, GTF produces minimal trace data for those events.

Note: Specification of SYS, SYSM, or SYSP causes GTF to ignore the following trace options if you specify them in any form: CSCH, HSCH, MSCH, SSCH, XSCH, IO, SVC, PI, EXT, RR.

TRC
This parameter requests that traced events include those related to GTF processing itself. If this parameter is not specified, GTF-related events are excluded from the trace output.

USR
This parameter requests that user data passed to GTF through the GTRACE macro be recorded with the system data.

Chapter 37. GTZPRMxx (Generic Tracker parameters)

GTZPRMxx parmlib members contain statements that control the behavior of Generic Tracker. The statements are:

- EXCLUDE - add an EXCLUDE statement.
- TRACKING - enable/disable tracking.
- CLEAR - reset/clear settings, statistics, and data.
- DEBUG - add a DEBUG statement.
- DIAGNOSE - for debug/service.

To specify which list of GTZPRMxx members the tracker should consider, use one of the following:

- The system parameter GTZ on the reply to message IEA101A at IPL time, via GTZ={xx|(xx,...,zz)}.
- The system parameter GTZ in IEASYSxx, via GTZ={xx|(xx,...,zz)}.
- The operator command SET GTZ={xx|(xx,...,zz)}.

Additional references and an overview for the tracking facility can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. The overview describes how to create GTZPRMxx parmlib members from existing CNIDTRxx parmlib members and how you can use utility program GTZCNIDT to convert CNIDTRxx (console tracker) parmlib members to GTZPRMxx parmlib members.

The syntax of GTZPRMxx parmlib members is a sequence of zero or more statements. The statements have the same syntax and semantics as SETGTZ subcommands. Both the SETGTZ command and the parmlib statements support keyword=value and keyword(value) syntax, but the preferred syntax for each is "keyword=value" for the SETGTZ command and "keyword(value)" for the parmlib syntax. The SETGTZ subcommands are:

- TRACKING
- CLEAR
- EXCLUDE
- DEBUG
- DIAGNOSE.

For more information on the SETGTZ subcommands, see the SETGTZ Command in *z/OS MVS System Commands*.

When multiple statements are specified in a GTZPRMxx parmlib member, the parser will attempt to recover from erroneous statements and will try to continue processing any remaining statements.

GTZPRMxx parmlib members and the contained statements are processed in the order specified.

Enter data only in columns 1 through 71. Do not enter data in columns 72 through 80 as the system ignores these columns. Comments may appear in columns 1-71 and must begin with "/*" (slash asterisk) and end with "*/" (asterisk slash).

See also SYS1.PARMLIB(GTZPRM00) for a working GTZPRMxx parmlib member that contains recommended statements (mainly exclusions in the form of EXCLUDE statements) as known to IBM at the time when a new release becomes available.

For a more current version of the GTZPRM00 parmlib member, check the web page http://www-03.ibm.com/systems/z/os/zos/downloads/.

Chapter 38. HZSPRMxx (manage IBM Health Checker for z/OS checks)

Use an HZSPRMxx parmlib member to make permanent updates to IBM Health Checker for z/OS checks and their parameters. The policy statements in the HZSPRMxx member or members currently in use for a system make up the IBM Health Checker for z/OS **policies** . The information in the current IBM Health Checker for z/OS policy will be applied to all existing checks and to any new checks you add. IBM Health Checker for z/OS processes policy information every time checks are refreshed, added, or when there is an IPL, so your policy is the place to put any check changes you want to make permanent and to have applied to any checks you add in the future.

For complete information about the HZSPRMxx parmlib member, including syntax and parameters, see Creating HZSPRMxx IBM Health Checker for z/OS policies in *IBM Health Checker for z/OS User's Guide*.

Chapter 39. IDAVDTxx (VSAM Dynamic Trace parameters)

Use the IDAVDTxx parmlib member to enable VSAM record management trace and define the trace parameter for any VSAM data set dynamically when diagnostic data is required.

You can predefine up to the maximum of 8 trace entries with different trace parameters. Each trace entry will consist of the target data set name, the target job name, and the usual trace parameters. When the trace entries are defined, you can use the START console command to call IDAVDT to enable VSAM record management trace. After you start IDAVDT, you can interact with IDAVDT by the MODIFY command to perform the following functions:

READIN
Parse the trace entries in the specified IDAVDTxx parmlib member and insert it into the dynamic trace save area located in common storage.

ENABLE
Enable a specific trace entry that is stored in the dynamic trace area.

DISABLE
Disable a specific trace entry that is stored in the dynamic trace area.

DISPLAY
Display a specific trace entry that is stored in the dynamic trace area.

VALIDATE
Validate a specific trace entry, or all trace entries, that are stored in the dynamic trace area.

INVALIDATE
Invalidate a specific trace entry, or all trace entries, that are stored in the dynamic trace area.

Syntax rules for IDAVDTxx

Follow the rules in "General syntax rules for the creation of members" on page 9. The following rule also applies to the creation of IDAVDTxx parmlib members:

- All text should be in uppercase.

Syntax format for IDAVDTxx

```
VTRACE   DSNAME(data_set_name)
           JOBNAME(job_name)
           HOOK(hook_id [,hook_id])
           PARM1(trace_parm_1)
           [PARM2(trace_parm_2)]
           [ECODE(ANY|rsncode)]
           [KEYVALUE(target_key_value|x'target_key_hex_value')]
           END
```

IBM-supplied defaults for IDAVDTxx

The default member is IDAVDT00.

Statements/parameters for IDAVDTxx

VTRACE
Specifies the current entry is to define a VSAM record management trace entry.

DSNAME(*data_set_name*)
Specifies which data set the user would like to enable VSAM record management trace.

JOBNAME(*job_name*)
Specifies the job where the targeted data set was opened or will be opened.

HOOK(*hook_id* [,*hook_id*])
Specifies the list of VSAM predefined Hook points where the trace should execute during VSAM record management processing. This statement consists of 1 to 3 other statements (SLIP, DUMP or ABEND).

PARM1(*trace_parm_1*)
Specifies which control blocks VSAM record management trace should capture.

PARM2(*trace_parm_2*)
Specifies which control blocks VSAM record management trace should capture when tracing the AIX®.

ECODE(ANY|*rsncode*)
Specifies that tracing should occur only if an error code is being returned to the caller.

ANY Tracing is performed for any nonzero return code.

rsncode
Tracing is preformed only if the RPLFDBK code matches the *rsncode*.

KEYVALUE(*target_key_value*|*x'target_key_hex_value'*)
Specifies tracing should occur only if the current processing key matched the *target_key_value*. The input value can be either HEX or EBCDIC value.

Examples of IDAVDTxx parmlib member

Example 1

```
VTRACE   DSNAME(SAMPLE.KSDS.DATA)
           JOBNAME(VRMTRA01)
           HOOK(0,1)
           PARM1(100101000000)
           KEYVALUE(x'C1C2C3C4C5C6')
           END
```

Chapter 40. IEAABD00 (ABDUMP written to a SYSABEND data set)

IEAABD00 contains IBM defaults and/or installation assigned parameters for ABDUMP, for use when an ABEND dump is written to a SYSABEND data set.

You may specify ABDUMP parameters for a dump to be taken to a SYSABEND data set as follows:

- The dump request parameter list pointed to by the DUMPOPT keyword of the ABEND macro. (An ABEND dump can also be requested through the CALLRTM or SETRP macros. See *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* or *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.) You can use the list form of the SNAP macro to build the list. (For details regarding the ABEND and SNAP macros, see *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.)
- The initial system dump options specified in IEAABD00. These options are added to the options on the dump request parameter list.
- The system dump options as altered by the CHNGDUMP command. With the CHNGDUMP command, options can be added to or deleted from the system dump options list. The CHNGDUMP command can also cause the dump request parameters to be ignored. (For a description of the CHNGDUMP command, refer to *z/OS MVS System Commands*.)

The ABDUMP initialization routine reads IEAABD00 to get ABDUMP parameters. If during initialization, IEAABD00 is invalid or cannot be located, the operator is notified. No prompting occurs. If both valid and invalid options are included in the member, or a syntax error is encountered, a message lists the valid options that were accepted before the error occurred.

Parameter in IEASYSxx (or specified by the operator)

None.

Syntax rules for IEAABD00

The following rules apply to the replacement of IEAABD00:

- There are two keywords, SDATA and PDATA. Each keyword is followed by a string of operands that are separated by commas and enclosed in parentheses. A single operand does not need parentheses.

```
SDATA=(SQA,CB,ENQ,TRT)          or  SDATA=ALLSDATA

PDATA=(PSW,REGS,SA,ALLPA,SPLS)  or  PDATA=ALLPDATA
```

- Normally both parameters (that is, SDATA=operands and PDATA=operands) can fit on one line. If, however, continuation is needed, use a comma followed by a blank.

```
SDATA=(SQA,CB,ENQ,TRT),
PDATA=(PSW,REGS,
SA,ALLPA,SPLS)
```

- Lines that begin with an asterisk in column 1 are comments.

- Alternatively, to include a comment, place the comment on the same line as the data. Use at least one blank to separate the data from the comment. The following example shows a continuation and the related comments.

```
SDATA=(SQA,CB,  SQA and CB data
ENQ,TRT)        ENQ and trace data
```

IBM-supplied default for IEAABD00

IBM places the following defaults in IEAABD00:

```
SDATA=(LSQA,CB,ENQ,TRT,ERR,DM,IO,SUM),
PDATA=(PSW,REGS,SPLS,ALLPA,SA)
```

These options request a dump of the following areas:

- LSQA, including subpools 229, 230, and 249
- formatted control blocks for the task
- formatted global resource serialization control blocks for the task
- GTF trace and/or system trace
- recovery termination control blocks for the task
- data management control blocks for the task
- IOS control blocks for the task
- summary data (see explanation under SUM parameter)
- PSW at entry to ABEND
- contents of general registers at entry to ABEND
- save area linkage information and a backward trace of save areas
- modules listed on the link pack area queue for the task and the job pack area queue for the task, and active SVC modules related to the failing task
- user storage allocated for the task.

Statements/parameters for IEAABD00

SDATA=

Request application related system storage, including key 0 areas, that the application programmer can see.

ALLSDATA

Except ALLVNUC and NOSYM, all the following options are automatically specified.

The following parameters request dump of specific SDATA areas, as indicated:

ALLVNUC

Entire virtual nucleus. SQA, LSQA, and the PSA are included.

NOSYM

No symptom dump is to be produced.

SUM

Requests that the dump contain summary data, which includes the following:

- Dump title.
- Abend code and PSW at the time of the error.

- If the PSW at the time of the error points to an active load module: (1) the name and address of the load module, (2) the offset into the load module indicating where the error occurred, and (3) the contents of the load module.
- Control blocks related to the failing task.
- Recovery termination control blocks.
- Save areas.
- Registers at the time of the error.
- Storage summary consisting of 1K (1024) bytes of storage before and 1K bytes of storage after the addresses pointed to by the registers and the PSW. The storage will be printed only if the user is authorized to obtain it, and, when printed, duplicate addresses will be removed.
- System trace table entries for the dumped address space.

NUC
Read/write portion of the control program nucleus. SQA, LSQA and the PSA are included.

PCDATA
Program call information for the task being dumped.

SQA
The system queue area.

LSQA
Local system queue area for the address space. If storage is allocated for subpools 229, 230 and 249, they will be dumped for the current task.

SWA
Scheduler work area used for the failing task.

CB Control blocks related to the failing task.

ENQ
Global resource serialization control blocks for the task.

TRT
System trace table and GTF trace, as available.

DM Data management control blocks (DEB, DCB, IOB) for the task.

IO IOS control blocks (UCB, EXCPD) for the task.

ERR
Recovery termination control blocks (RTM2WA, registers from the SDWA, SCB, EED) for the task.

PDATA=
Request application related problem program key storage areas.

ALLPDATA
All the following options are automatically specified.

The following parameters request dump of specific PDATA areas, as indicated:

SUBTASKS
Problem data (PDATA) options requested for the designated task will also be in effect for its subtasks.

PSW
Program status word at entry at ABEND.

REGS
Contents of general registers at entry to ABEND.

SA or SAH
SA requests save area linkage information and a backward trace of save areas. This option is automatically selected if ALLPDATA is specified. SAH requests only save area linkage information.

JPA
Contents of the job pack area (module names and contents) that relate to the failing task.

LPA
Contents of the LPA (module names and contents) related to the failing task. Includes active SVCs related to the failing task.

ALLPA
Contents of both the job pack area and the LPA, as they relate to the failing task, plus SVCs related to the failing task.

SPLS
User storage subpools (0-127, 129–132, 244, 251, and 252) related to the failing task.

Chapter 41. IEAAPFxx (authorized program facility list)

Use the IEAAPFxx member to specify program libraries that are to receive authorized program facility (APF) authorization. List the names (dsnames) of the libraries, along with one of the following to indicate where the library resides:

- The volume serial number of the volume on which the library resides.
- Six asterisks (******) to indicate that the library resides on the current system residence (sysres). volume.
- ***MCAT*** to indicate the library resides on the volume on which the master catalog resides.
- Nothing after the library name, to indicate that the storage management subsystem (SMS) manages the library.

If an installation wants IEAAPFxx, it must explicitly create the member. The member IEAAPF00 must be explicitly created by the installation also.

Defining aliases in the APF List: You should not define aliases in the APF list because IBM's data management services (for example, OPEN processing) map an alias to its actual library name and query the APF list by the actual library name. An alias in the APF list does not authorize anything.

You can use IEAAPFxx to create an APF list in a static format. A static list can be updated only at IPL and can contain a maximum of 255 entries (SYS1.LINKLIB, SYS1.SVCLIB, and 253 entries specified by your installation).

Note: IBM provides the PROGxx parmlib member as an alternative to IEAAPFxx, which allows you to update the APF list dynamically and specify an unlimited number of APF-authorized libraries. IBM suggests that you use PROGxx to specify the APF list (regardless of whether you plan to take advantage of the dynamic update capability). The system will process IEAAPFxx and PROGxx if both parameters are specified. If you decide to use PROGxx **only**, remove APF=xx system parameters from IEASYSxx and IEASYS00.

For information about how to use the IEAAPFPR REXX exec to convert the APF definitions in IEAAPFxx to equivalent definitions in PROGxx, see “Specifying the APF list” on page 29.

For information about how to use PROGxx to specify the format and contents of the APF list, see PROG.

For more information about using authorized libraries, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Note:

1. Except for concatenations opened during NIP, any unauthorized library that is concatenated to authorized libraries will cause all of the concatenated libraries to be considered unauthorized.
2. You can specify a maximum of 253 library names in an IEAAPFxx member.

3. Allowing SMS to manage a data set means that the data set might be moved to a different volume during normal SMS processing. To ensure the data set retains APF authorization, specify nothing after the library name, to indicate that the library is managed by SMS.
4. As of MVS 4.3, it is no longer necessary for the data sets in the LPALST to be APF-authorized. However, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library in the LPALST to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Parameter in IEASYSxx (or supplied by the operator)

APF=xx
The two characters (A-Z, 0-9, @, #, or $), represented by xx, are appended to IEAAPF to identify the IEAAPFxx member. If the APF list is not explicitly defined (via the APF system parameter, the default IEAAPF00 parmlib member, or APF statements within PROGxx via the PROG system parameter), then only SYS1.LINKLIB and SYS1.SVCLIB will be APF-authorized during the IPL.

Note: In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Syntax rules for IEAAPFxx

The following rules apply to the creation of IEAAPFxx:

- Enter data only in columns 1 through 71. Do not enter data in column 72 through 80; the system ignores these columns.
- Place only one library name and corresponding volume serial number on a line (record).
- Duplicate data set names are valid.
- On each record, first enter the library name, then one or more blanks, then one of the following:
  - The volume serial number of the volume on which the library resides.
  - ****** (six asterisks), to indicate that the library resides on the current SYSRES volume.
  - ***MCAT***, to indicate that the library resides on the volume that contains the master catalog.
  - Nothing, to indicate the library is SMS-managed.
- To continue to another record, place a comma after the volume serial number. Omit this comma on the last record.

```
first record:       SYS1.SUPER.UTILS       614703,

second record:      W12.PAYROLL.LOADLIB     705650
```

- On a line, data entered after the comma is treated as a comment and ignored.
- Data records entered after the last data line are treated as comments and ignored.

```
SYS1.SUPER.UTILS        614703,   Super utilities

SMS.MANAGED.LOADLIB           ,   No vol=>SMS managed

W12.PAYROLL.LOADLIB     705650,   /* Payroll programs */

X11.LOADLIB             345000    /* Last line */

This line and any following lines are ignored.
```

- For an SMS-managed volume, use only standard comments (delimited by /* and */) on the last line.

```
SYS1.SUPER.UTILS        614703,   Super utilities

SMS.MANAGED.LOADLIB1          ,   No vol=>SMS managed

W12.PAYROLL.LOADLIB     705650,   Payroll programs

X11.LOADLIB             345000,   /* X11 programs */

SMS.MANAGED.LOADLIB2              /* Last line */
```

IBM-supplied default for IEAAPFxx

If the APF list is not explicitly defined (via the APF system parameter, the default IEAAPF00 parmlib member, or APF statements within PROGxx via the PROG system parameter), then only SYS1.LINKLIB and SYS1.SVCLIB is APF-authorized during the IPL. Additionally, if the default for the LNKAUTH system parameter is taken (LNKAUTH=LNKLST) or is specified in IEASYSxx or by the operator, libraries in the LNKLST concatenation are also authorized when accessed as part of the LNKLST concatenation.

Note:

1. When LNKAUTH=APFTAB is specified, the system considers SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE and SYS1.SIEAMIGE to be APF-authorized when they are accessed as part of the concatenation (even when they are not included in the APF list).
2. In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) is treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Statements/parameters for IEAAPFxx

Not applicable.

Chapter 42. IEAAPP00 (authorized I/O appendage routines)

IEAAPP00 contains the names of authorized installation-written I/O appendage routines. These appendages, when listed, can be used by any unauthorized user program. Otherwise, only programs authorized under APF or running under system protection key (0-7) may use the EXCP appendages. If your installation does not use EXCP appendages, you need not create IEAAPP00.

If the EXCP caller is not in system protection key or the job step is not authorized, the system verifies that the caller's appendage names are listed. If the names cannot be found, the system issues a 913 ABEND. If, however, the caller is authorized, the system loads the appendages without inspecting the list.

Syntax rules for IEAAPP00

The following rules apply to the creation of IEAAPP00:

- IEAAPP00 can contain up to five entries, with each entry specifying names of a particular type of appendage. For example:

  ```
  SIOAPP        Start I/O
  PCIAPP        Program-Controlled Interrupt
  EOEAPP        End of extent
  ABEAPP        Abnormal end
  CHEAPP        Channel end
  ```

  You need not necessarily use all five types of entries.
- Each entry consists of an appendage-type name, followed by a list of suffixes of that type, separated by commas. The suffixes can range from WA to Z9. The appendage-type name can start in any column. For example, this entry specifies two Start I/O appendages.

  ```
  SIOAPP        WA,X4
  ```
- Continuation is indicated by a comma followed by one or more blanks. The next record can start in any column.
- Comments are indicated by an asterisk in the first column or a combination of /* and */ between columns 1 and 71. Comments are allowed at the start of a member or interspersed throughout the member.

Here is an example of a complete IEAAPP00 member. In this example, note that there are no channel-end appendages and none for abnormal end. Routine IGG019X3 is used as both end-of extent and PCI appendage.

```
SIOAPP        Y1,Y2,
EOEAPP        X1,Z2,X3,X4,X5,X6,
PCIAPP        X3
```

IBM-supplied default for IEAAPP00

None.

Statements/parameters for IEAAPP00

Not applicable.

Chapter 43. IEACMD00 (IBM-supplied commands)

IEACMD00 contains IBM-supplied commands, as follows:

- A CHNGDUMP command to add trace table, LSQA and XES information to SVC dumps.
- A SET command (SET SLIP=00) to indicate that the system is to use the IEASLP00 parmlib member to issue IBM-supplied SLIP commands.
- A SET command (SET DAE=00) to indicate that the system is to use the ADYSET00 parmlib member to start DAE processing.
- A START command (START LLA, SUB=MSTR) to start the library lookaside (LLA) procedure, which resides in SYS1.PROCLIB. The LLA procedure then starts the library lookaside function.
- A START command (START BLSJPRMI, SUB=MSTR) creates IPCS tables which allows SNAP ABDUMP and IPCS to print formatted control blocks in dumps.
- A START command (START IFGEDI,SUB=MSTR) enables enhanced data integrity for physical sequential data sets.

An installation that uses the optional COMMNDxx parmlib member should check for commands in COMMNDxx that conflict with commands in IEACMD00, and resolve any conflicts. It is also recommended that an installation place its commands in the COMMNDxx member, leaving the IEACMD00 member for IBM-supplied commands only.

Place all SLIP commands in IEASLPxx.

Note:

1. The order in which task-creating commands appear in IEACMD00 does NOT guarantee the order in which they are executed. Thus, IEACMD00 should not be used for commands that must be executed in a specific order. Commands are issued in the order that they appear in IEACMD00, but they are executed as follows:
   - Immediate commands, such as DISPLAY T, are executed sequentially as they are issued from IEACMD00.
   - Execution of task-creating commands, such as DISPLAY A, are deferred until system initialization is complete. Then, factors such as multitasking, multiprocessing, and competition for resources influence the order in which these commands are executed.

     Thus, IEACMD00 should *not* be used to issue task-creating commands that must be executed in a specific order, because the execution order of these commands can vary.
2. The SET SLIP=00 command causes the IBM-supplied IEASLP00 parmlib member to execute. As a result, the SLIP commands specified in IEASLP00 are in effect at IPL-time. These commands suppress dumps that are considered to be unnecessary. For more information about IEASLP00, see Chapter 51, "IEASLPxx (SLIP commands)," on page 415.

   To activate an IEASLPxx member instead of the IBM-supplied member (IEASLP00), change the SET SLIP=00 command to SET SLIP=xx (where xx is the two character suffix of the alternate IEASLPxx member to be used).

3. The SET DAE=00 command causes the ADYSET00 parmlib member to execute. As a result, dump analysis and elimination (DAE) processing as specified in ADYSET00 is in effect.

   To prevent the system from activating DAE automatically, change ADYSET00 to contain DAE=STOP.

   For more information about DAE and its parmlib members, see Chapter 3, "ADYSETxx (dump suppression)," on page 65.
4. For descriptions of the CHNGDUMP, SET, START, and SLIP commands, see *z/OS MVS System Commands*.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for IEACMD00

The following rules apply to the modification of IEACMD00:

- Enter only one command for each line. To do so, specify the COM= keyword, followed by the command enclosed in single quotation marks.
- Do not specify continuation on any line.
- Lines that begin with an asterisk in column 1 are comments.

IBM-supplied default for IEACMD00

The following commands are placed in IEACMD00 by IBM:

```
COM='CHNGDUMP SET,SDUMP=(LSQA,TRT,XESDATA),ADD'
COM='SET SLIP=00'
COM='SET DAE=00'
COM='START LLA,SUB=MSTR'
COM='START BLSJPRMI,SUB=MSTR'
COM='START IFGEDI,SUB=MSTR'
```

Statements/parameters for IEACMD00

Not applicable.

Chapter 44. IEADMCxx (DUMP command parmlib)

IEADMCxx enables you to supply DUMP command parameters through a parmlib member. IEADMCxx enables the operator to specify the collection of dump data without having to remember and identify all the systems, address spaces and data spaces involved.

This parmlib enables you to specify lengthy DUMP commands without having to reply to multiple writes to operator with reply (WTORs). Any errors in an original specification may be corrected and the DUMP command re-specified.

IEADMCxx is an installation-supplied member of SYS1.PARMLIB that can contain any valid DUMP command. A dump command may span multiple lines and contain system static and (DUMP command SYMDEF defined) symbols and comments.

Dump commands using parmlib members are specified by member suffix using the PARMLIB=keyword. Multiple members may be specified within parentheses separated by commas. A dump command specified using the parmlib keyword are taken immediately without prompting for further parameters through WTOR. Since no WTOR replies are required, the ROUTE command can be used when issuing a DUMP command specifying parmlib members for increased functionality.

Note: Do not use the ROUTE command with the DUMP command when specifying parmlib members that use the REMOTE keyword.

Performance implications

None.

Syntax rules for IEADMCxx

The following syntax rules apply to IEADMCxx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- Blank lines are permitted.
- Comments may appear in columns 1-71, can span lines, and must begin with "/*" and end with "*/".
- Do not use nested comments in this parmlib member.
- A statement can be continued even though there is no explicit continuation character.
- You can use substrings of previously defined system symbols. See “Step 1. Know the rules for using system symbols in parmlib” on page 51 for more information about using substrings.

Syntax format of IEADMCxx

The syntax of the DUMP command specified within the IEADMCxx members of SYS1.PARMLIB is identical to that specified on the DUMP command through replies. See *z/OS MVS System Commands* for more information.

IBM-supplied default for IEADMCxx

None.

Statements/parameters for IEADMCxx

DUMP
The DUMP command.

```
COMM={(title)}
     {'title'}
     {"title"}

TITLE={(title)}
      {'title'}
      {"title"}
```

The title (1 to 100 characters) that you assign to the dump. The title is the first record in the dump data set.

PARMLIB=xx |PARMLIB=(xx[,xx]...)
The two alphanumeric characters indicating the IEADMCxx member of SYS1.PARMLIB that contains the DUMP command specification.

The syntax of the DUMP command specified within the IEADMCxx members of SYS1.PARMLIB is identical to that specified on the DUMP command through replies. See *z/OS MVS System Commands* for more information.

Note: A DUMP's title is determined as follows:

- Titles specified in the DUMP command (for example, DUMP TITLE="DUMP Specified through WTOR") take precedence over titles specified within PARMLIBs.
- When a title is not specified in the DUMP command, the title specified within a PARMLIB takes precedence.

  If titles are specified in multiple PARMLIBs, the first PARMLIB's title takes precedence. For example, if all of the PARMLIBS in **PARMLIB=(RA,XC,CF)** are titled, the dump title is the one specified in the **RA** PARMLIB.
- If a title is not specified in the DUMP command or PARMLIB, the title is as follows: DUMP FOR PARMLIB=(*xx,yy,zz*), where *xx,yy,zz* are the PARMLIBs.

The following are examples of IEADMCxx usage:

Example 1: To request a dump through WTOR, enter:

```
DUMP COMM=(Dump specified via WTOR)

or

DUMP TITLE=(Dump specified via WTOR)
```

In response to this command, the system issues the following response:

```
 * id IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

Replying to this command determines the contents of the dump.

Example 2: To request a dump of the RACF address space, including all of its private area and a summary dump using a DUMP command parmlib member:

1. Create member IEADMCRA

```
JOBNAME=RASP,SDATA=(SUM,RGN)
```

2. Issue the following on the MVS console:

```
DUMP TITLE=(RACF dump),PARMLIB=RA
```

Example 3: To request a dump of the XCF address space, including all of its data spaces on all systems within the sysplex using a DUMP command parmlib member:

1. Create member IEADMCXC

```
JOBNAME=XCFAS,DSPNAME=('XCFAS'.*),SDATA=COUPLE,
REMOTE=(SYSLIST=*('XCFAS'),DSPNAME,SDATA)
```

2. Issue the following on the MVS console:

```
DUMP TITLE=(Dump all XCF data),PARMLIB=XC
```

Example 4: To request a dump that includes structure control data for CACHESTRUCTURE, directory information and entry data for storage classes 3-8 and 10 with entry data written with serialization, and directory information for all entries grouped by cast-out class using a DUMP command parmlib member:

1. Create member IEADMCCF

```
STRLIST=(STRNAME=CACHESTRUCTURE,
ACCESSTIME=NOLIMIT,
(STGCLASS=(3-8,10),ENTRYDATA=SERIALIZE),(COCLASS=ALL))
```

2. Issue the following on the MVS console:

```
DUMP TITLE=(CACHESTRUCTURE dump),PARMLIB=CF
```

Example 5: To request a dump of the RACF address space on this system, the XCF address spaces and data spaces on all systems in the sysplex, and the CACHESTRUCTURE structure from the coupling facility, issue the following command on the MVS console:

```
DUMP TITLE=(RACF, XCFASs and CACHESTRUCTURE),PARMLIB=(RA,XC,CF)
```

Example 6: To show the use of variable substitution where &REGIONAM is a new symbol and &SYSCLONE is a static symbol:

1. Where a PARMLIB member, like IEADMCTC, could contain:

```
TSONAME=(TCPIP&SYSCLONE.,CICS&REGIONAM.),SDATA=(SWA,RGN)
```

2. The MVS console DUMP command includes the SYMDEF for &REGIONAM:

```
DUMP TITLE=(Dump TCPIP and CICS),PARMLIB=TC,
SYMDEF=&REGIONAM.='TOR7')
```

Example 7: To request the dump of TCP/IP on this system, including only the high virtual CSA that is owned by the ASID and a summary dump, using a DUMP command parmlib member:

1. Run the following command to create member IEADMCTP:

```
JOBNAME=TCPIP, SDATA=(SUM,HCSAByASID)
```

2. Run the following command on the MVS console:

```
DUMP TITLE=(TCPIP dump),PARMLIB=TP
```

Chapter 45. IEADMP00 (ABDUMP written to a SYSUDUMP data set)

IEADMP00 contains IBM defaults and installation parameters for ABDUMP for use when an ABEND dump is written to a SYSUDUMP data set. ABDUMP parameters for a SYSUDUMP data set may be specified as follows:

- The dump request parameter list pointed to by the DUMPOPT keyword of an ABEND macro. (An ABEND dump can also be requested through the CALLRTM and SETRP macros. For details, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.) The list can be built by using the list form of the SNAP macro. (For details, see ABEND and SNAP macros in *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.)
- The initial system dump option specified in IEADMP00. This option is added to the options on the dump request parameter list.
- The system dump options as altered by the CHNGDUMP command. With the CHNGDUMP command, options can be added to or deleted from the system dump options list. The CHNGDUMP command can also cause the dump request parameters to be ignored. (For a description of the CHNGDUMP command, refer to *z/OS MVS System Commands*.)

During IPL an information message will notify the operator if IEADMP00 is invalid or cannot be found. No prompting of the operator will occur. If the member contains both valid and invalid parameters, an information message will indicate the valid options that were accepted before the error occurred.

Parameter in IEASYSxx (or specified by the operator)

None.

Syntax rules for IEADMP00

The following rules apply to the replacement of IEADMP00:

- There are two keywords, SDATA and PDATA. Each keyword is followed by a string of operands separated by commas and enclosed in parentheses. A single operand does not need parentheses.

```
SDATA=(SQA,CB,ENQ,TRT)            or    SDATA=ALLSDATA

PDATA=(PSW,REGS,SA,ALLPA,SPLS)    or    PDATA=ALLPDATA
```

- Normally both parameters (SDATA=operands and PDATA=operands) can fit on one line. If, however, continuation is needed, use a comma followed by a blank.

```
SDATA=(SQA,CB,ENQ,TRT),
PDATA=(PSW,REGS,
SA,ALLPA,SPLS)
```

- Lines that begin with an asterisk in column 1 are comments.
- Alternatively, to include a comment, place the comment on the same line as the data. Use at least one blank to separate the data from the comment. The following example shows a continuation and related comments.

```
SDATA=(SQA,CB,  SQA and CB data
ENQ,TRT)        ENQ and trace data
```

IBM-supplied default for IEADMP00

The IBM-supplied default parmlib member is IEADMP00, which contains SDATA=SUM.

Statements/parameters for IEADMP00

SDATA=

Request application related system storage, including key 0 areas, that the application programmer can see.

ALLSDATA

Except ALLVNUC and NOSYM, all the following options are automatically specified.

The following parameters request dump of specific SDATA areas, as indicated:

ALLVNUC

Entire virtual nucleus. SQA, LSQA, and the PSA are included.

NOSYM

No symptom dump is to be produced.

SUM

Requests that the dump contain summary data, which includes the following:

- Dump title.
- Abend code and PSW at the time of the error.
- If the PSW at the time of the error points to an active load module: (1) the name and address of the load module, (2) the offset into the load module indicating where the error occurred, and (3) the contents of the load module.
- Control blocks related to the failing task.
- Recovery termination control blocks.
- Save areas.
- Registers at the time of the error.
- Storage summary consisting of 1K (1024) bytes of storage before and 1K bytes of storage after the addresses pointed to by the registers and the PSW. The storage will be printed only if the user is authorized to obtain it, and, when printed, duplicate addresses will be removed.
- System trace table entries for the dumped address space.

NUC

Read/write portion of the control program nucleus. SQA, LSQA, and the PSA are included.

PCDATA

Program call information for the task being dumped.

SQA

The system queue area.

LSQA

Local system queue area for the address space. If storage is allocated for subpools 229, 230 and 249, they will be dumped for the current task.

SWA

Scheduler work area used for the failing task.

CB Control blocks related to the failing task.

ENQ
Global resource serialization control blocks for the task.

TRT
System trace table and GTF trace, as available.

DM Data management control blocks (DEB, DCB, IOB) for the task.

IO IOS control blocks (UCB, EXCPD) for the task.

ERR
Recovery termination control blocks (RTM2WA, registers from the SDWA, SCB, EED) for the task.

PDATA=
Request application related problem program key storage areas.

ALLPDATA
All the following options are automatically specified.

The following parameters request dump of specific PDATA areas, as indicated:

PSW
Program status word at entry to ABEND.

REGS
Contents of general registers at entry to ABEND.

SA or SAH
SA requests save area linkage information and a backward trace of save areas. This option is automatically selected if ALLPDATA is specified. SAH requests only save area linkage information.

JPA
Contents of the job pack area that relate to the failing task. These include module names and contents.

LPA
Contents of the LPA related to the failing task. These include module names and contents. Also includes active SVCs related to the failing task.

ALLPA
Contents of both the job pack area and the LPA, as they relate to the failing task, plus SVCs related to the failing task.

SPLS
User storage subpools (0-127, 129–132, 244, 251, and 252) related to the failing task.

SUBTASKS
Problem data (PDATA) options requested for the designated task will also be in effect for its subtasks.

Chapter 46. IEADMR00 (ABDUMP written to a SYSMDUMP data set)

IEADMR00 contains IBM defaults and/or installation parameters for ABDUMP, for use when an ABEND dump is written to a SYSMDUMP data set. ABDUMP parameters for a SYSMDUMP data set may be specified as follows:

- The dump request parameter list pointed to by the DUMPOPT keyword of an ABEND macro. (An ABEND dump can also be requested through the CALLRTM and SETRP macros. For details, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* or *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.) The list can be built by using the list form of the SNAP macro. (For details, see the descriptions of the ABEND and SNAP macros in *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.)
- The initial system dump options specified in IEADMR00. These options are added to the options on the dump request parameter list.
- The system dump options as altered by the CHNGDUMP command. With the CHNGDUMP command, options can be added to or deleted from the system dump options list. The CHNGDUMP command can also cause the dump request parameters to be ignored. (For a description of the CHNGDUMP command, see *z/OS MVS System Commands*.)

During IPL, an informational message will notify the operator if IEADMR00 is invalid or cannot be found. No prompting of the operator will occur. If the member contains both valid and invalid parameters, an informational message will indicate the valid options that were accepted before the error occurred.

Recommendation for IEADMR00 with z/OS UNIX

The system writes a SYSMDUMP as the *core dump* of a forked address space that runs a z/OS UNIX process. A core dump is written to an HFS file on behalf of the user experiencing the error. To obtain sufficient diagnostic data without consuming excessive storage in the file system, request the following options in IEADMR00:

```
SDATA=(RGN,TRT,SUM)
```

Parameter in IEASYSxx (or specified by the operator)

None.

Syntax rules for IEADMR00

The following rule applies to the replacement of IEADMR00:

- There is one keyword, SDATA. The keyword is followed by a string of operands separated by commas and enclosed in parentheses. A single operand does not need parentheses.

  ```
  SDATA=(SQA,TRT)            or    SDATA=ALLSDATA
  ```

- Lines that begin with an asterisk in column 1 are comments.
- Alternatively, to include a comment, place the comment on the same line as the data. Use at least one blank to separate the data from the comment. The following example shows a continuation and the related comments.

```
SDATA=(SQA,LSQA,  SQA and LSQA data
SWA,TRT)          SWA and trace data
```

IBM-supplied default for IEADMR00

The IBM-supplied default parmlib member is IEADMR00, which contains SDATA=(NUC,SQA,LSQA,SWA,TRT,RGN,SUM).

Statements/parameters for IEADMR00

SDATA=

ALLSDATA
Except ALLNUC and NOSYM, all the following options are automatically specified.

The following parameters request dump of specific SDATA areas, as indicated:

NUC
Read/write portion of the control program nucleus. The PSA is included.

SQA
The system queue area.

LSQA
Local system queue area for the address space. If storage is allocated for subpools 229, 230 and 249, they will also be dumped.

SWA
Scheduler work area used for the failing task.

TRT
System trace table and GTF trace, as available.

LPA
Contents of the LPA related to the failing task. These include module names and contents. Also includes SVCs related to the failing task.

CSA
Common service area subpools that are not fetch protected.

RGN
Private area for the address space region.

GRSQ
All global resource serialization control blocks and other global resource serialization storage.

ALLNUC
Entire control program nucleus. The PSA is included.

NOSYM
No symptom dump is to be produced.

SUM
Summary dump information as provided in the SVC dump. (See the description of the SUM parameter for the SDUMP macro in *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.)

Chapter 47. IEAFIXxx (fixed LPA list)

Use the IEAFIXxx member to specify the names of modules that are to be fixed in storage for the duration of an IPL. (The libraries, which contain the modules, must be cataloged in the system master catalog, unless the VOLUME parameter is specified.) You can use IEAFIXxx to temporarily add or replace SVC or ERP routines that already exist in the pageable LPA (PLPA), or to page fix such routines to improve system performance.

Like the temporary modules chosen through the MLPA option, fixed link pack area (FLPA) modules are not automatically reactivated by a quick start or a warm start IPL. That is, the FLPA can be reestablished only by re-specification of the FIX parameter at the quick start or warm start IPL.

Note:

1. Any library that includes modules for the FLPA must be a PDS. You cannot use PDSEs in the LPALST concatenation.
2. As of MVS Version 4.3, it is no longer necessary for the data sets in the LPALST to be APF-authorized. However, any module in the fixed link pack area will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected any data set named in IEAFIXxx to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Because fixed modules are not paged, you can save I/O time and paging overhead by placing moderately used modules into the FLPA. This can shorten the LPALST concatenation. When a module is requested, the program manager searches the list of fixed routines before it examines the pageable LPA directory. The price for this performance improvement is the reduction in storage available for paging old jobs and starting new jobs. Remember that pages referenced frequently tend to remain in storage even when they are not fixed.

You can use the FLPA to reduce page-fault overhead at the expense of some storage. This trade-off is desirable with a system that tends to be CPU-bound but has sufficient storage.

You carry out the trade-off by placing moderately used modules into the FLPA (through IEAFIXxx). High usage PLPA modules probably need not be fixed because they are referenced frequently enough to remain in storage. Because less storage will be available for pageable programs, the system will swap out address spaces that would otherwise occupy storage, awaiting CPU availability.

Modules specified in IEAFIXxx are loaded and fixed in the order in which they are specified in the member (to keep search time within reasonable limits, do not allow the FLPA to become excessively large). If the first load module of a type 3 or 4 SVC routine is specified, the SVC table is updated as required.

Modules specified in IEAFIXxx are placed in the FLPA or in the extended FLPA, depending on the RMODE of the modules. Modules with an RMODE of 24 are placed in the FLPA, while those with an RMODE of ANY are placed in the extended FLPA.

The FLPA and the extended FLPA are page-protected by default. A protection exception occurs if there is an attempt to store data into either area.

To override page protection, use the NOPROT option on the FIX system parameter.

Note: The FLPA and the extended FLPA are not mapped V=R. Modules in either area cannot be assumed to be backed by contiguous real frames.

Parameter in IEASYSxx (or specified by the operator)

```
FIX=   {aa                      }
       {(aa[,L][,NOPROT])       }
       {(aa,bb,...[,L][,NOPROT])}
```

The two characters (A-Z, 0-9, @, #, or $) represented by aa (or bb, etc.), are appended to IEAFIX to form the name of one or more IEAFIXxx members of SYS1.PARMLIB. The system defaults to no FLPA, if you fail to specify the option in one of the following ways:

- FIX keyword included in the IEASYSxx member.
- FIX keyword entered by the operator at IPL.

The LPA modules that are fixed in storage are also page protected, by default. If an attempt is made to store into a page-protected module, a protection exception occurs. However, an installation can use the NOPROT option to override the page protection default. When NOPROT is specified, the LPA modules in the IEAFIXxx parmlib member(s) are not page protected in storage.

If the L option is specified, the system displays the contents of the IEAFIXxx parmlib member(s) at the operator's console as the system processes the member(s).

Syntax rules for IEAFIXxx

The following rules apply to the creation of IEAFIXxx:

- Each statement must begin with the INCLUDE keyword.
- The library name follows the LIBRARY keyword and is enclosed in parentheses.
- The list of modules follows the MODULES keyword and is enclosed in parentheses. Any number of module names can be specified.

  Use major names or alias names, or both. Any alias names by which a module is to be accessed must be included. If an alias name is included and the associated major name is omitted, the system will locate the major name.
- The statement is assumed to be all information from one INCLUDE keyword to the next INCLUDE keyword or until end-of-file.
- Use all columns except 72 through 80.
- Blanks or commas can be used as delimiters. Multiple occurrences of a delimiter are interpreted as a single delimiter.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". They are allowed anywhere a delimiter is allowed.

Syntax format of IEAFIXxx

```
INCLUDE    LIBRARY(data-set-name)

           VOLUME(volser)

           MODULES(module-name, module-name,...)
```

Syntax example of IEAFIXxx

```
INCLUDE   LIBRARY(SYS1.LINKLIB)  MODULES( IKJPARS IKJPARS2 IKJSCAN
                                          IKJEFD00 IKJDAIR)

INCLUDE  LIBRARY(SYS1.SVCLIB)  MODULES(IGC000RC,IGC09302,IGC09303)

INCLUDE  LIBRARY(SYS1.AUTHLIB)   MODULES(IEAV0021,IEAV0032,IEAV0033,
                                         IEAV0041,IEAV0042,IEAV0043,
                                         IEAV0053,IEAV0054)
```

IBM-supplied default for IEAFIXxx

None.

Statements/parameters for IEAFIXxx

INCLUDE
Specifies modules to be loaded as a temporary extension to the existing pageable link pack area (PLPA).

Both LIBRARY and MODULES must be specified. VOLUME is optional.

LIBRARY
Specifies a data set name. The specified library, which must be a PDS, must be cataloged in the system master catalog unless you specify a volser using VOLUME. In this case, the data set must be on the specified volume.

Note: If you specify SYS1.LINKLIB or SYS1.LPALIB, the system uses only those libraries, and not their concatenations.

MODULES
Specifies a list of 1 to 8 character module names.

Chapter 48. IEALPAxx (modified LPA list)

Use the IEALPAxx member to specify the reenterable modules that are to be added as a temporary extension to the pageable link pack area (PLPA). (The libraries, which contain the modules, must be cataloged in the system master catalog, unless the VOLUME parameter is specified.) This extension is temporary; the modules will remain in paging data sets and will be listed on the active LPA queue only for the duration of the IPL.

The system will not automatically quick-start or warm-start these modules (that is, reinstate the modules without re-specification of the MLPA parameter). Both the modified LPA (MLPA) and the fixed LPA (FLPA) modules (those named in IEAFIXxx) do not require the system to search the LPA directory when one of the modules is requested. The modified LPA, unlike the fixed LPA, however, contains pageable modules that behave in most respects like PLPA modules.

Note:

1. A library that includes modules for the MLPA must be a PDS. You cannot use PDSEs in the LPALST concatenation.
2. The data sets in the LPALST can be a mixture of APF-authorized and non-APF-authorized data sets. However, any module in the modified link pack area will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected any library that contributes modules to the modified link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.
3. If duplicate entries are found for a module or alias name, the first instance is used and subsequent instances are ignored.

You can use IEALPAxx to temporarily add or replace SVC or ERP routines. Another possible application would be the testing of replacement LPA modules that have been altered by PTFs.

LPA modules specified in IEALPAxx are placed in the MLPA or in the extended MLPA, depending on the RMODE of the modules. Modules with an RMODE of 24 are placed in the MLPA, while those with an RMODE of ANY are placed in the extended MLPA. By default, the MLPA and the extended MLPA are page protected, which means that a protection exception will occur if there is an attempt to store data into either area. To override page protection, use the NOPROT option on the MLPA system parameter.

LPA modules that have been replaced through IEALPAxx are not physically removed from the PLPA or from the LPA directory. They are, however, logically replaced because, when one of them is requested, MLPA is searched first and the system does not examine the LPA directory which contains the name of the replaced module.

The system searches the fixed LPA before the modified LPA for a particular module and selects the module from the modified LPA only if the module is not also in the fixed LPA.

If the first load module of a type 3 or 4 SVC routine is added or replaced, the SVC table is updated as required.

Parameter in IEASYSxx (or supplied by the operator)

```
MLPA= {aa                     }
      {(aa[,L][,NOPROT])      }
      {(aa,bb,...[,L][,NOPROT])}
```

The two characters (A-Z, 0-9, @, #, or $) represented by aa (or bb, and so forth), are appended to IEALPA to form the name of the IEALPAxx parmlib members. If the L option is specified, the system displays the contents of the IEALPAxx parmlib members at the operator's console as the system processes the members.

Because the modified LPA is not a permanent addition to the LPA, you should specify MLPA in an alternate system parameter list (IEASYSxx) and not in IEASYS00. Or, you can have the operator enter the parameter from the console.

The LPA modules in the IEALPAxx parmlib members are page-protected in storage, by default. If an attempt is made to store into a page-protected module, a protection exception occurs. However, the NOPROT option allows an installation to override the page protection default. When NOPROT is specified, the LPA modules in the IEALPAxx parmlib members are not page-protected.

Syntax rules for IEALPAxx

The following rules apply to the creation of IEALPAxx:

- Each statement must begin with the INCLUDE keyword.
- The library name follows the LIBRARY keyword and is enclosed in parentheses.
- The list of modules follows the MODULES keyword and is enclosed in parentheses. Any number of module names can be specified.

  Use major names or alias names, or both. Any alias names by which a module is to be accessed must be included. If an alias name is included and the associated major name is omitted, the system will locate the major name.
- The statement is assumed to be all information from one INCLUDE keyword to the next INCLUDE keyword or until end-of-file.
- Use all columns except 72 through 80.
- Blanks or commas can be used as delimiter. Multiple occurrences of a delimiter are interpreted as a single delimiter.
- Lines that begin with an asterisk in column 1 are comments.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". They are allowed anywhere a delimiter is allowed.

Syntax format of IEALPAxx

```
INCLUDE    LIBRARY(data-set-name) [VOLUME(volser)]
           MODULES(module-name, module-name,...)
```

Syntax example of IEALPAxx

```
INCLUDE   LIBRARY(SYS1.LINKLIB)  MODULES( IKJPARS IKJPARS2 IKJSCAN
                                          IKJEFD00 IKJDAIR)

INCLUDE  LIBRARY(SYS1.SVCLIB)  MODULES(IGC000RC,IGC09302,IGC09303)

INCLUDE  LIBRARY(SYS1.U30LIB)   MODULES(IEAU0021,IEAU3002,IEAU3003)

INCLUDE  LIBRARY(SYS1.AUTHLIB)   MODULES(IEAV0021,IEAV0032,IEAV0033,
                                         IEAV0041,IEAV0042,IEAV0043,
                                         IEAV0053,IEAV0054)
INCLUDE LIBRARY(MY.U90LIB) VOLUME(U90PAK) MODULES(MYMOD1,MYMOD2)
```

IBM-supplied default for IEALPAxx

None.

Statements/parameters for IEALPAxx

INCLUDE
Specifies modules to be loaded as a temporary extension to the existing PLPA. Both **LIBRARY** and **MODULES** *must* be specified. VOLUME is optional.

LIBRARY
Specifies a data set name. The specified library, which must be a PDS, must be cataloged in the system master catalog unless you specify a volser using VOLUME in which case the data set must be on the specified volume. (If you specify SYS1.LINKLIB or SYS1.LPALIB, the system uses only those libraries, and not their concatenations.)

MODULES
Specifies a list of 1 to 8 character module names.

Chapter 49. IEAOPTxx (OPT parameters)

The OPT parameters allow the installation to change many system resources manager (SRM) constants. The following list categorizes the options as special including, for example, abnormal termination or whether non-enclave transaction work of queue servers and enclave servers is to be managed or not, adjusting options for constants, or options for special assist processors.

The OPT contains several categories of information. The keywords need not be grouped by category and can appear in any order. The system ignores both repetition of a keyword and the use of keywords that were used in prior releases and are no longer supported.

Special options

- [,ABNORMALTERM=*option*]
- [,CNTCLIST=*option*]
- [,DVIO=*option*]
- [FULLPRESYSTEM=YES | NO]
- [HIPERDISPATCH=YES | NO]
- [,INITIMP=option]
- [,MANAGENONENCLAVEWORK=YES | **NO**]
- [,VARYCPU=*option*]
- [,VARYCPUMIN=*nn*]
- [,STORAGESERVERMGT=YES | **NO**]
- [REPORTCOMPLETIONS=YES | **NO**]

Adjusting constants options

- Enqueue residence constants
  - [,ERV=*xxxxxx*]
- SRM invocation interval constant
  - [,RMPTTOM=*xxxxxx*]
- MPL adjustment constants
  - [,RCCFXET=(*xxxxx*,*yyyyy*)]
  - [,RCCFXTT=(*xxxxx*,*yyyyy*)]
- CPU management constants
  - [,BLWLINTHD=*option*]
  - [,BLWLTRPCT=*option*]
  - [,CCCAWMT=*xxxxxxx*]
  - [,CCCSIGUR=*option*]
  - [,TIMESLICES=*option*]
- Pageable storage and real storage constants
  - [,IRA405I(n)=*xx*]
  - [,MCCAFCTH=*xxxxx*,*yyyyy*]
  - [,MCCFXEPR=*xxx*]
  - [,MCCFXTPR=*xxx*]
  - [,STORAGENSWDP=*xxx*]

  - [,STORAGEWTOR=*xxx*]
- Selective enablement for I/O constants
  - [,CPENABLE=(*xxx*,*yyy*)]
- Routing options
  - [,WASROUTINGLEVEL=*x*]
  - [,RTPIFACTOR=*xxx*]
- Resource contention constants
  - [,MAXPROMOTETIME=*xxxx*]

Special assist processor options

- [IFAHONORPRIORITY=**YES** | NO]
- [IIPHONORPRIORITY= **YES** | NO]
- [PROJECTCPU=YES | **NO**]
- [ZAAPAWMT=*xxxxx*]
- [ZIIPAWMT=*xxxxx*]

Multi-threading controls

- MT_CP_MODE=*x*
- MT_ZIIP_MODE=*x*

Syntax rules for IEAOPTxx

Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

IBM-supplied default for IEAOPTxx

The IBM-supplied default parmlib member is IEAOPT00 with ERV=500.

Statements/parameters for IEAOPTxx

ABNORMALTERM=option

Specifies if the abnormal terminations, that are reported by the WLM report service IWMRPT for a server, are included when computing the weight of the routing service IWM4SRSC that is called for the server. The syntax is:

ABNORMALTERM=NO means that the abnormal terminations, that are reported by the IWMRPT service, are not included when computing the weight of the IWM4SRSC routing service. ABNORMALTERM=YES means that the abnormal terminations are included when computing the weight of the IWM4SRSC routing service.

Values: YES or NO

Default Value: YES

BLWLINTHD=*option*

Specifies the threshold time interval for which a blocked address space or enclave must wait before being considered for promotion.

If the CPU utilization of a system is at 100%, workloads with low importance (low dispatch priority) might not get dispatched anymore. This can lead to problems if the low priority work holds a resource that is required by high priority workloads. Therefore, if an address space or enclave has ready-to-run work units (TCBs or SRBs) but does not get CPU service for the specified time

interval because of its low dispatch priority, it will be temporarily promoted to a higher dispatch priority. Address spaces that are swapped out are not considered for promotion.

Value range: 1-65535 seconds

Default Value: 20 seconds

BLWLTRPCT=*option*

Specifies how much of the CPU capacity is to be used to promote blocked workloads. This parameter does not influence the amount of CPU service that a single blocked address space or enclave is given. Instead, this parameter influences how many different address spaces or enclaves can be promoted at the same point in time. If the value specified with this parameter is not large enough, blocked workloads might need to wait longer than the time interval defined by BLWLINTHD.

Value Range: 0-200 (up to 20%. 0% indicates blocked workload support is not enabled.)

Default Value: 5 (0.5%)

CCCAWMT=*xxxxxxx*

Specifies whether to activate or deactivate Alternate Wait Management (AWM). If AWM is active, SRM and LPAR cooperate to reduce low utilization effects and overhead. In an LPAR, some n-way environments with little work appear to require more capacity than expected because of the time spent waking up idle logical and physical processors to compete for individual pieces of work. If AWM is active, SRM and LPAR will reduce this unproductive use of processor so that capacity planning is more accurate and CPU overhead is reduced.

Value Range: For HIPERDISPATCH=NO, any value between 1 and 499999 makes AWM active, and any value between 500000 and 1000000 makes AWM inactive. AWM is active or inactive only for general CPs, zAAPs, and zIIPs.

For HIPERDISPATCH=YES, the valid range is 1600 to 3200. Any other value will be reset to the default value of 3200. AWM is always active and cannot be turned off.

Default Value: AWM is active. For HIPERDISPATCH=NO, the default is 12000 (12 ms). For HIPERDISPATCH=YES, the default is 3200 (3.2 ms).

- There can be a very significant effect on performance by using large values (such as 490,000) to activate AWM. IBM suggests accepting the default value. Specify a value for CCCAWMT in order to override the default after consulting with the IBM Support Center. Do not specify what is thought to be the default value as a means of disabling any OPT control. Default values sometimes change over time. Always convert a control to a comment (see the following) if the objective is to use the default value while retaining a reminder of the last value used.

```
/* CCCAWMT=500000     now a comment - used to turn off AWM   */
```

- Specify a value greater than or equal to 500,000 to deactivate AWM. AWM should be active when special assist processors, such as System z Application Assist Processors (zAAPs) and IBM System z9® Integrated Information Processors and IBM System z10® Integrated Information Processors (zIIPs) are present and IFAHONORPRIORITY=YES or IIPHONORPRIORITY=YES is specified. See the description of IFAHONORPRIORITY and IIPHONORPRIORITY for more information. You should accomplish this by removing the CCCAWMT parameter.

CCCSIGUR=*option*

Specifies the minimum mean-time-to-wait threshold value for heavy CPU users. This constant is used to determine the range of mean-time-to-wait values which are assigned to each of the ten mean-time-to-wait dispatching priorities. The specified real time value is adjusted by relative processor speed to become SRM time to insure consistent SRM control across various processors.

Value Range: 0-32767 milliseconds

Default Value: 45

CNTCLIST=*option*

Specifies if the individual commands in a TSO/E CLIST are treated as separate commands for transaction control.

In the syntax, option is either YES or NO. CNTCLIST=NO specifies that the CLIST is treated as a single transaction. CNTCLIST=YES specifies that each command is to be treated as an individual transaction. By specifying CNTCLIST=YES, SRM control of a TSO/E command becomes the same whether the command is executed explicitly or as part of a CLIST.

Values: YES or NO

Default Value: NO

CPENABLE=(*xxx***,***yyy***)**

Specifies the low (ICCTPILO) and high (ICCTPIHI) threshold values for the percentage of I/O interruptions to be processed through the test pending interrupt (TPI) instruction path in IOS. SRM uses the following thresholds to control the number of processors enabled for I/O interruptions.

Value Range	Basic Mode Default	LPAR Mode Default
a=0-100%	10	0
b=0-100%	30	0

Note: For MVS running in LPAR mode with dedicated CPUs, specify the basic mode default values for CPENABLE. For recommendations on setting CPENABLE parameters, please see ATS Flash1033 at the following URL:

`http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10337`

DVIO=*option*

Specifies whether directed VIO is to be active in the system or not.

DVIO=YES, the default, specifies that directed VIO is to be active in the system; that is, the NONVIO keyword of the IEASYSxx parmlib member is honored. DVIO=NO specifies that directed VIO is not to be active in the system; the NONVIO parameter of the IEASYSxx parmlib member is ignored.

Values: YES or NO

Default Value: YES

FULLPRESYSTEM= [YES | NO]

Specifies the SRM to allow full preemption for the system address spaces.

In FULLPRESYSTEM=YES mode, SRM allows full preemption handling for system address spaces. To free up a processor immediately when such an address space becomes ready, the system issues signal processor (SIGP) instructions. The SIGP instructions also run in HiperDispatch mode, so when HIPERDISPATCH=YES and FULLPRESYSTEM=YES, too much LPAR overhead can result. Therefore:

- Use FULLPRESYSTEM=NO when HIPERDISPATCH=YES
- Use FULLPRESYSTEM=YES when HIPERDISPATCH=NO.

FULLPRESYSTEM=**NO** Specifies the SRM to turn off full preemption mode for the system address spaces.

Value Range: YES or NO

Default Value:

- YES when HiperDispatch is not in effect.
- NO when HiperDispatch is in effect.

ERV=*xxxxxx*

Specifies the number of CPU service units that an address space or enclave is allowed to absorb when it is possibly causing enqueue contention.

During this "enqueue residency" time, the address space (including the address space associated with an enclave) is not considered for swap-out based on recommendation value analysis. The address space or enclave runs with a high enough priority to guarantee the needed CPU time.

ERV is in effect for an address space or enclave that meets one of the following criteria:

- The address space or enclave is enqueued on a system resource needed by another address space.
- An authorized program in the address space or enclave obtains control of the resource (even if another address space does not need that resource) as a result of issuing a reserve for a DASD device which is SHARED.

Note: SRM determines the execution time equivalent to the specified ERV by multiplying the ERV by the model-dependent time needed to accumulate 1 CPU service unit.

Example: ERV=2

In the example above, if an address space consumes 1 service unit in 10 milliseconds, it will be allowed to execute for 20 milliseconds before it will be eligible for swap-out while enqueued on a resource requested by other address spaces.

Value Range: 0-999999

Default Value: 500

HIPERDISPATCH=YES|NO

YES
Specifies that SRM should turn on HiperDispatch mode.

NO Specifies that SRM should turn off HiperDispatch mode.

Note: All partitions with greater than 64 logical processors defined at IPL are forced to run with HIPERDISPATCH=YES. After IPL, LPARs with greater than 64 logical processors defined are unable to switch into HIPERDISPATCH=NO. Also, when HIPERDISPATCH=YES (when it is supported by the processor), VARY CPU management is automatically turned off no matter what is specified for VARYCPU.

Note: On IBM z13 and follow-on processors that support multi threading mode, when partitions specify LOADxx PROCVIEW=CORE, they are forced to run with HIPERDISPATCH=YES. After IPL, these partitions are

unable to switch into HIPERDISPATCH=NO. Also, when HIPERDISPATCH=YES (when it is supported by the processor), VARY CPU management is automatically turned off no matter what is specified for VARYCPU.

Default Value:

For IBM zEnterprise® 196 (and follow-on processors): YES

For IBM System z10 processors: NO

IFAHONORPRIORITY=YES|NO

YES
Specifies that standard processors run both zAAP processor eligible and non-zAAP processor eligible work in priority order when the zAAP processors indicate the need for help from standard processors. The need for help is determined by the alternate wait management (AWM) function of SRM for both standard and zAAP processors. Standard processors help each other and standard processors can also help zAAP processors if YES is in effect. Specifying YES does not mean the priorities will always be honored because the system manages dispatching priorities based on the goals provided in the WLM service definition. AWM should not be disabled when IFAHONORPRIORITY=YES is in effect. See the description for parameter CCCAWMT for a description of AWM.

If zAAP processors are defined to the LPAR but are not online, the zAAP processor eligible work units are processed by standard processors in priority order. The system ignores the IFAHONORPRIORITY parameter in this case and handles the work as if it had no eligibility to zAAP processors. The zAAP processor eligible processor times are reported in RMF and SMF for planning purposes.

IBM suggests that you specify or default to IFAHONORPRIORITY=YES.

NO Specifies that standard processors will never examine zAAP processor eligible work. Note that standard processors also run zAAP processor eligible work, if it's necessary to resolve contention for resources with non-zAAP processor eligible work.

Default Value: YES

IIPHONORPRIORITY=YES|NO

YES
Specifies that standard processors run both zIIP processor eligible and non-zIIP processor eligible work in priority order when the zIIP processors indicate the need for help from standard processors. The need for help is determined by the alternate wait management (AWM) function of SRM for both standard and zIIP processors. Standard processors help each other and standard processors can also help zIIP processors if YES is in effect, which is the default. Specifying YES does not mean the priorities will always be honored because the system manages dispatching priorities based on the goals provided in the WLM service definition. AWM should not be disabled when IIPHONORPRIORITY=YES is in effect. See the description for parameter CCCAWMT for a description of AWM.

If zIIP processors are defined to the LPAR but are not online, the zIIP processor eligible work units are processed by standard processors in priority order. The system ignores the IIPHONORPRIORITY parameter in

this case and handles the work as if it had no eligibility to zIIP processors. The zIIP processor eligible processor times are reported in RMF and SMF for planning purposes.

IBM suggests that you specify or default to IIPHONORPRIORITY=YES.

NO Specifies that standard processors will never examine zIIP processor eligible work. Note that standard processors also run zIIP processor eligible work, if it's necessary to resolve contention for resources with non-zIIP processor eligible work.

Default Value: YES

INITIMP=*option*

Specifies the dispatching priority for JES, APPC, and OMVS initiators. The option is specified as one of the following values:

- 0 — The initiator dispatching priority is set to 254.
- 1,2, or 3 — Defines that the initiator dispatching priority has to be lower than the dispatching priority for CPU critical work with the same or a higher importance level.

  If no service class with the CPU critical attribute and a corresponding or higher importance level is defined in the WLM policy, the dispatching priority is calculated in the same way as for parameter INITIMP=E.
- E — Defines that the initiator dispatching priority will be calculated in the same way as the enqueue promotion dispatching priority. The dispatching priority is calculated dynamically to ensure access to the processor. It should not impact high importance work; however, there is no guarantee that CPU critical work will always have a higher dispatching priority.

An INTIMP value affects only the system on which it has been set.

Example: INITIMP=2

With INITIMP=2, if there are service classes with an importance of 2 and the CPU critical attribute, the dispatching priority of initiator address spaces will always be lower than the dispatching priority of the work running in those service classes. Because CPU critical work with a higher importance level will always have a higher dispatching priority, the dispatching priority of the initiator will also be lower than any CPU critical work with an importance of 1.

With INITIMP=2 and CPU critical work with an importance of 1 and 3, the initiator will always have a lower dispatching priority than the work running in the service class with the CPU critical attribute and an importance of 1 but not for the work running in the CPU critical service class with an importance of 3.

With INITIMP=2 and no service classes with the CPU critical attribute and an importance of 1 or 2, the initiator dispatching priority is set to the enqueue promotion dispatching priority which is dynamically calculated. Please note that the initiators will not run at a dispatching priority of 254 as that is only the case if you do not specify the INITIMP parameter or if you have set it to 0.

Default Value: 0

IRA405I=*n*

Specifies the percentage of the fixed storage that causes the system to issue message IRA405I. Specify a value for *n* to indicate the storage area, where *n* can be:

0 Real storage area below 16M

1 Real storage area between 16M and 2G

2 Total real storage

Note that when you specify the percentage value, you must specify the value without the percent symbol (%), as shown in the following example:

```
IRA405I(1)=60
```

When each value of the *n* is specified, the value ranges and the default values for the IRA405I parameter can be:

IRA405I(0)
Value Range: 0 to 100.

Default Value: 70.

IRA405I(1)
Value Range: 0 to 100.

Default Value: 50.

IRA405I(2)
Value Range: 0 to 100.

Default Value: 50.

MANAGENONENCLAVEWORK=*option*

Specifies whether non-enclave transaction work of queue servers and enclave servers is to be managed or not, where *option* is YES or NO.

MANAGENONENCLAVEWORK=YES indicates that SRM is to manage the non-enclave transaction work towards the first service class period of the address space performance goal. Based on this expanded performance management, be sure to verify the performance goals for the service class of the address spaces that process the enclave work. For a detailed description, see the chapter "Performance Management of Address Spaces with Enclaves" in *z/OS MVS Programming: Workload Management Services*.

MANAGENONENCLAVEWORK=NO indicates that you expect no work consuming significant CPU service to be running in the address space outside of an enclave. The CPU consumption of work running outside of enclaves is not included when workload management assesses the impact of CPU adjustments for the enclave work.

This performance management is available for z/OS V1R12 and later releases.

Note: To complete a MANAGENONENCLAVEWORK status change from YES to NO or NO to YES for an active system, SRM issues a policy refresh for that active system. Workload Manager indicates the policy refresh for this status change with message IWM065I.

Values: YES or NO

Default Value: NO

Example: MANAGENONENCLAVEWORK=YES

In the example, SRM is to manage the non-enclave work of enclave servers and queue servers

MAXPROMOTETIME=*xxxx*

Specifies the time during which a resource holder is allowed to run promoted when it is possibly causing a resource contention. A resource holder is either

an address space or enclave. During this "resource contention residency" time, the resource holder runs with the highest priority of all resource waiters to guarantee the needed importance. Also during this interval, the address space (including the address space associated with an enclave) is not considered for swap-out based on recommendation value analysis. If MAXPROMOTETIME=0 is specified, SRM does not promote any resource holder.

MAXPROMOTETIME is in effect if the resource manager has notified the address space or enclave under contention by using the WLM IWMCNTN notification service. See *z/OS MVS Programming: Workload Management Services* for a description of the WLM IWMCNTN service.

The value specified for MAXPROMOTETIME is multiplied by 10. This is the time span in seconds during which the address space or enclave under contention is allowed to run promoted.

Example: MAXPROMOTETIME=12

With MAXPROMOTETIME=12, SRM allows an address space or enclave to run 12 × 10 seconds (=2 minutes) promoted. When the time is exceeded, the promotion is canceled.

Value Range: 0-1000

Default Value: 6

MCCAFCTH=(*lowvalue***,***okvalue***)**
Specifies the low and the OK threshold values for storage. The *lowvalue* indicates the number of frames on the available frame queue when stealing begins. The *okvalue* indicates the number of frames on the available frame queue when stealing ends. SRM will automatically adjust the actual threshold values based on measurements of storage usage, but threshold values should never fall below the values specified in MCCAFCTH. You do not have to specify an MCCAFCTH value.

If the *lowvalue* or *okvalue* is below the default value, the default value is enforced.

Value lowvalue=1-32767 frames

Default
The maximum of 400 and 0.2% of the pageable storage in the system

Value okvalue=1-32767 frames

Default
The maximum of 400 and 0.2% of the pageable storage in the system

MCCFXEPR=*xxx*
Specifies the percentage of storage that is fixed within the first 16 megabytes. SRM uses this threshold to determine when a shortage of pageable storage exists because there are too many fixed pages.

Value Range: 0-100 percent

Default Value: 92 percent

MCCFXTPR=*xxx*
Specifies the percentage of online storage that might be fixed. SRM uses this threshold to determine when a shortage of pageable storage exists.

Note: SRM uses the lesser of the values, (MCCFXTPR × amount of online storage) and (MCCFXEPR × amount of storage that is fixed within the first 16 megabytes) to set the threshold frame count so that it can detect a shortage of

pageable storage that is caused by too much page fixing. In this way, SRM can detect a shortage of pageable storage that is caused by too much page fixing before or at the same time as a shortage caused by too much paging.

Value range: 0-100 percent

Default value: On small systems (less than 320 GB), the target is 80 percent. On large systems (more than 320 GB), the target is total storage minus 64 GB.

Note: If 100% - MCCFXTPR * the total amount of online frames is greater than 64 gigabytes, the MCCFXTPR keyword is not used in determining the threshold at which a shortage of pageable storage exists.

Instead, on systems with more than 320 gigabytes of storage, a pageable storage shortage is detected when less than 64 gigabytes of online storage is pageable.

When calculating the number of frames that can be page fixed before a pageable storage shortage is detected, SRM uses the maximum of MCCFXTPR * the total online frames and total online storage minus 64 gigabytes.

MT_CP_MODE=1
MT_ZIIP_MODE=1|2

Specifies the multithreading (MT) mode, which is the number of active threads per core for all online cores with a core type of standard CP, or System z Integrated Information Processor (zIIP).

Default Value: 1

MT_CP_MODE=
The number of active threads for each online CP core. For CPs only a value of 1 is valid.

MT_ZIIP_MODE
The number of active threads for each online zIIP core. The value is limited to what the underlying hardware and operating system support.

A value of 1 is always accepted.

Activating an MT mode greater than 1, requires the following:
- The hardware must support multithreading for the respective processor class, namely zIIPs.
- On the HMC Customize/Delete Activation Profiles task, "Do not end the time slice if a partition enters a wait state" must not be checked. This is the recommended default setting.
- The system must be IPLed with LOADxx PROCVIEW CORE in effect.
- HIPERDISPATCH=YES must be in effect.

If these requirements are met, the MT mode for all cores of the respective processor class is changed to the specified value/mode if the current MT mode is different from that value. In case of a successful change of the MT Mode, message IWM066 is issued. For an unsuccessful request, message IWM067 is issued.

Changing the MT Mode for all cores of the same type can take multiple seconds to complete.

PROJECTCPU=[YES | NO]

Specifies whether to activate or deactivate the projection of how work could be

offloaded from regular CPs to special assist processors like the System z Application Assist Processor (zAAP) and the System z Integrated Information Processor (zIIP).

Note:

1. The PROJECTCPU parameter is not necessary if you can define a zIIP or zAAP as a reserved processor in the LPAR configuration. The PROJECTCPU parameter is provided for users of earlier processors on which defining a zIIP as a reserved processor is not supported when no zIIP is purchased for planning purposes.
2. If the installation requires ZIIP statistics but there is no ZIIP defined (reserved or actual) on the LPAR, then PROJECTCPU=YES is required regardless of hardware model.
3. Any work that is eligible for being offloaded to a special assist processor will be reported as Special_Processor_on_CP work. This information can be used to understand the benefit of adding a special processor into the configuration.

Values: YES or NO

Default Value: NO - projection will not be done, unless a special assist processor is configured to the system. In that case, the system will collect projection values.

RCCFXET=*option*

SRM uses these thresholds to determine whether the system MPL needs to be increased or decreased based on the first 16 MB. The values that you can specify for *option* are as follows:

(*aaa***,***bbb***)**

Specifies the low (RCCFXETL) and the high (RCCFXETH) percentages of storage that is fixed within the first 16 megabytes.

Value range: 0-100 percent

Default values:

aaa = 82

bbb = 88

RCCFXTT=*option*

SRM uses these thresholds to determine if the system MPL needs to be increased or decreased. The values that you can specify for *option* are as follows:

(*aaa***,***bbb***)**

Specifies the low (RCCFXTTL) and the high (RCCFXTTH) percentages of online storage that is fixed.

Value range: 0-100 percent

Default values:

aaa = 66

bbb = 72

RMPTTOM=*xxxxxx*

Specifies the SRM invocation interval. The specified real-time interval is adjusted by relative processor speed to become SRM time in order to ensure consistent SRM control across various processors. The relationship of real time

to SRM time for each processor is described in the "Advanced SRM Parameter Concepts" section of *z/OS MVS Initialization and Tuning Guide*.

Value Range: 1000-999999 msec

Default Value:

- 3000 (for systems with a uni-processor speed of more than 100 MIPS)
- 1000 (for systems with a uni-processor speed of 100 MIPS or less)

RTPIFACTOR=*xxx*

Specifies how much the server performance index (PI) should affect the server routing weights returned by WLM services IWM4SRSC and IWMSRSRS with FUNCTION=SPECIFIC.

- When RTPIFACTOR is 0, the server weight is independent from the server PI.
- When RTPIFACTOR is 100 and the server PI is bigger than 1, the server weight is divided by the server PI.
- When RTPIFACTOR is between 1 and 99, it results in a corresponding intermediate influence of the server PI on the server weight returned by WLM.

Value Range: 0-100

Default Value: 100

STORAGENSWDP=*option*

Specifies if the system should select non-swappable address spaces to resolve a storage shortage.

STORAGENSWDP=YES specifies that the system should also select non-swappable address spaces to resolve the storage shortage. The system then sets non-swappable address spaces non-dispachable and issues message IRA210E or IRA410E.

Note: The system does not set address spaces in service class SYSTEM non-dispatchable.

Values: YES or NO

Default Value: YES

STORAGESERVERMGT=*option*

Specifies whether SRM should pass service class importance and goal information to the storage I/O priority manager in the IBM System Storage® DS8000® series. The passed information enables the storage I/O priority manager to provide favored processing for I/O requests of important z/OS workloads that are missing their goals.

STORAGESERVERMGT=YES specifies that SRM should pass service class importance and goal information to the storage I/O priority manager. Before specifying STORAGESERVERMGT=YES, verify that your IBM System Storage DS8000 model incorporates the storage I/O priority manager feature.

If STORAGESERVERMGT=YES is specified, the storage I/O priority manager may throttle I/O requests to facilitate a favored access to storage server resources for other I/O requests. The storage I/O priority manager considers the properties of the service class period associated with an I/O request to determine whether the I/O request should receive a favored processing or may be throttled:

- For service class periods with a response time goal, the goal achievement and specified importance are considered.
- For service class periods with a velocity goal, the specified velocity goal and importance are considered.
- I/O requests associated with the system-provided service classes SYSTEM, SYSSTC, or SYSSTC1 - SYSSTC5 are not managed by the I/O priority manager.
- I/O requests associated with service class periods that have a discretionary goal will always be throttled.

Throttle delays introduced by the storage IO priority manager are reflected in control unit queue delays. Therefore, if STORAGESERVERMGT=YES is specified, control unit queue delays are not considered when the achieved velocity and performance index are calculated for service class periods with a velocity goal.

If you have significant control unit queue delays in your installation, you may have to adjust the velocity goal of service class periods when you specify STORAGESERVERMGT=YES.

Values: YES or NO

Default Value: NO

STORAGEWTOR=*option*

Specifies how the system handles address spaces in a critical storage shortage.

STORAGEWTOR=YES specifies that the system presents a list of address spaces on the console. The operator can reply to a WTOR (IRA221D or IRA421D) request and select which address space to cancel. STORAGEWTOR=AUTO is similar to STORAGEWTOR=YES, except that the IRA210I or IRA420I message presents up to 20 address spaces at once. This option is useful, when an automation product needs to answer the WTOR request.

Values: YES, AUTO, or NO

Default Value: YES

TIMESLICES=*option*

Specifies the number of timeslices that a CPU-intensive address space or enclave with a discretionary goal should be given before a dispatchable unit of equal importance is dispatched.

Increasing this parameter might increase the processor delay for some CPU-intensive work, but decrease the number of context switches between equal priority work, and therefore increase the throughput of the system. This parameter only affects discretionary work that is CPU-intensive as determined by significant mean time to wait (MTTW) (see parameter CCCSIGUR).

Value Range: 1-255

Default Value: 1

VARYCPU=*option*

Specifies whether LPAR Vary CPU management is available or not available. Note that Vary CPU management is available only in a partition that is enabled for LPAR weight management. Also, Vary CPU management is automatically turned off no matter what is specified for VARYCPU, if HIPERDISPATCH is on.

VARYCPU=YES, the default, specifies that LPAR Vary CPU Management is available for this system. VARYCPU=NO specifies that LPAR Vary CPU Management is *not* available.

Value Range: YES or NO

Default Value: YES

VARYCPUMIN=*nn*
Specifies the minimum number of CPs (*nn*) that must stay online during WLM LPAR management. WLM LPAR management will not take CPs offline below this threshold.

Value Range: 1 - 64

Default Value: 1

WASROUTINGLEVEL=0|1
Specifies the routing algorithm used by the WLM Websphere routing services.

0 WLM uses the most advanced routing algorithm that is supported by all systems in the sysplex.
- If the release level of the lowest system is z/OS R9 (or above), routing decisions are based on available or dispatchable capacity of standard and assist processors.
- If the release level of the lowest system is below z/OS R9, but has the APAR OA16486 installed, routing decisions are based on available or dispatchable capacity of standard processors only.
- If the release level of the lowest system is below z/OS R9 and runs without the APAR OA16486 installed, the routing algorithm round robin is used.

1 WLM uses the routing algorithm round robin. If this option is used, set it on all systems of the sysplex so that all are using the same algorithm; otherwise, the WebSphere® routing service on each system uses the specific algorithm setting for the system, which can lead to inconsistent results.

Default Value: 0

ZAAPAWMT=*xxxxxx*
Specifies an Alternate Wait Management (AWM) value for IBM System z Application Assist Processors (zAAPs) to minimize SRM and LPAR low utilization effects and overhead. In an LPAR, some n-way environments with a small workload may appear to have little capacity remaining because of the time spent waking up idle zAAPs to compete for individual pieces of work. The ZAAPAWMT parameter allows you to reduce this time so that capacity planning is more accurate and CPU overhead is reduced, even though it might take longer until arriving work gets dispatched.

Value Range: For HIPERDISPATCH=NO, the valid range is 1 to 499999 microseconds. Any value between 1 and 499999 makes AWM active. To inactivate AWM, set CCCAWMT to any value between 500000 and 1000000.

For HIPERDISPATCH=YES, the valid range is 1600 to 499999 microseconds. Any other value will be reset to the default value of 3200. AWM is always active and cannot be turned off.

Default Value: AWM is active. For HIPERDISPATCH=NO, the default is 12000 (12 ms), and for HIPERDISPATCH=YES, the default is 3200 (3.2 ms).

ZIIPAWMT=*xxxxxx*
Specifies an Alternate Wait Management (AWM) value for the IBM System z9

Integrated Information Processor (zIIP) to minimize SRM and LPAR low utilization effects and overhead. In an LPAR, some n-way environments with a small workload may appear to have little capacity remaining because of the time spent waking up idle zIIPs to compete for individual pieces of work. The ZIIPAWMT parameter allows you to reduce this time so that capacity planning is more accurate and CPU overhead is reduced, even though it might take longer until arriving work gets dispatched.

Value Range: For HIPERDISPATCH=NO, the valid range is between 1 and 499999 microseconds. Any value between 1 to 499999 makes AWM active. To inactivate AWM, set CCCAWMT to any value between 500000 and 1000000.

For HIPERDISPATCH=YES, the valid range is 1600 to 499999 microseconds. Any other value will be reset to the default value of 3200. AWM is always active and cannot be turned off.

Default Value: AWM is active. For HIPERDISPATCH=NO, the default is 12000 (12 ms), and for HIPERDISPATCH=YES, the default is 3200 (3.2 ms).

Note: The ZAAPAWMT and ZIIPAWMT parameters internally affect the frequency with which the specialty engines will check the need for help. If help is required, the zAAP or zIIP processor signals a waiting zAAP or zIIP to help. When all zAAP or zIIP processors are busy and IFAHONORPRIORITY/IIPHONORPRIORITY=YES, the zAAP or zIIP processors ask for help from the standard processors. All available speciality engines (that is, all zAAPs or all zIIPs) must be busy before help is asked of the standard processors. Even with IFAHONORPRORITY=YES or IIPHONORPRIORITY=YES parameters set, the general CP will not help the specialty engine for work in a discretionary service class.

Reducing the value specified for ZAAPAWMT or ZIIPAWMT causes the specialty engines to request help after being busy for a shorter period of time. If IFAHONORPRIORITY or IIPHONORPRIORITY is set to YES, help is provided to one CP at a time, in the priority order of zAAP or zIIP processor eligible work, non-zAAP or non-zIIP processor eligible work. Reducing the ZAAPAWMT or ZIIPAWMT value too low can cause the standard processors to run an excessive amount of zAAP or zIIP processor eligible workload, which might result in lower priority non-zAAP/zIIP processor eligible work to be delayed. Conversely, increasing the value specified for ZAAPAWMT or ZIIPAWMT causes the specialty engines to request help only after being busy for a longer period of time, which might delay the standard processors from providing help when it is necessary.

Chapter 50. IEAPAKxx (LPA pack list)

IEAPAKxx is an installation-supplied member that contains groups of names of modules in the LPALST concatenation that are executed together or in sequence. (As an example, the load modules of a type-4 SVC routine are called sequentially and executed as a group.) The member is used only during a "cold" start (CLPA specified), when the PLPA is loaded from the LPALST concatenation.

The system uses the specified IEAPAKxx member(s) to determine the order in which modules are to be loaded from the LPALST concatenation into the pageable LPA. These modules are packed together, if possible, on a single page. The purpose is to reduce page faults. The LPA can greatly contribute to page faults because it is highly used. The IEAPAKxx list can significantly reduce page faults.

Each group ideally should not exceed 4K bytes in size. If a group exceeds 4K, the module in the group that causes 4K to be exceeded, and all later modules in the same group, will be loaded at the next page boundary in the LPA. In contrast, the system loads other modules (those not listed in IEAPAKxx) in size order, the largest modules first, then the smaller modules. Unused spaces within page boundaries are filled, if possible, with modules smaller than 4K.

You should select link pack area programs for inclusion in the system pack list to reduce the number of page faults from the pageable link pack area and thereby enhance system performance. The affinity of programs for each other and the size of the programs determine which should be selected for a pack list entry. Program affinity means that one program will usually refer to another program when the first program is invoked. By putting programs that refer to each other into the same pack list entry, and thereby on the same page, extra page faults are avoided, because the programs are always in storage together.

Very large programs that are to be put into the link pack area should be link edited so that CSECTs that have affinity or other CSECTs are placed within the same page. The linkedit ORDER statement can be used to group CSECTs into pages. This process will accomplish the same goal as the pack list entries, because page faults during execution will be reduced. The modules either execute concurrently or call each other.

Note: An installation can create one or more IEAPAKxx parmlib members, as needed. The system uses the IEAPAK00 member (if it exists) if PAK=xx is not included in the IEASYSxx parmlib member or is not specified by the operator. However, initialization continues whether or not there is an IEAPAK00 or IEAPAKxx member.

Parameter in IEASYSxx (or supplied by the operator)

```
PAK={aa             }
    {(aa,bb,...[,L])}
    {(,L)           }
```

The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, etc.), are appended to IEAPAK to form the name of the IEAPAKxx parmlib member(s). If the 'L' option is specified, the system displays the contents of the IEAPAKxx member(s) at the operator's console when the system processes the member(s).

Syntax rules for IEAPAKxx

The following rules apply to the creation of IEAPAKxx:

- The member consists of "groups" or entries containing load module names. Each group is enclosed in parentheses. For example: (IGG019CM,IGG019CN,IGG019CO,IGG019CP,IGG019CR).
- All modules in a group must have the same residence mode (RMODE).
- Separate modules within each group by commas.
- Separate each group from the next by a comma after the closing parenthesis.
- Do not use alias module names. The system processes only major names.
- All named modules must be refreshable because LPA modules must have this attribute.
- Lines that begin with an asterisk in column 1 are comments.
- On a line, data entered after the last group or load module name and after the optional comma continuation character is treated as a comment and ignored.
- Data records entered after the last data line are treated as comments and ignored.

IBM-supplied default for IEAPAKxx

The IBM-supplied default parmlib member is IEAPAK00, which contains (IEFBR14).

Statements/parameters for IEAPAKxx

Not applicable.

Chapter 51. IEASLPxx (SLIP commands)

Use IEASLPxx to contain SLIP commands. The commands can span multiple lines, and the system processes the commands in order.

It is recommended that you move any SLIP commands in the COMMNDxx and IEACMDxx parmlib members into a IEASLPxx parmlib member. By using IEASLPxx to contain your SLIP commands, you avoid restrictions found in other parmlib members. For example:

- IEASLPxx supports multiple-line commands; IEACMD00 does not.
- IEASLPxx does not require any special command syntax; IEACMD00 does.

If you move SLIP commands from COMMNDxx and IEACMDxx to IEASLPxx, you must replace the commands in COMMNDxx and IEACMDxx with the following line, where xx is the ID of the IEASLPxx member to be used.

```
COM='SET SLIP=xx'
```

See the descriptions of the COMMNDxx and IEACMDxx parmlib members for related information.

Note:

1. For an SVC dump suppressed by a SLIP command with the action of NODUMP or NOSVCD, a LOGREC data set entry will contain a code indicating the reason for the suppression. For an ABEND dump suppressed by a SLIP command with the action of NODUMP, NOSYSU, NOSYSA, or NOSYSM, message IEA848I will indicate the reason for the suppression.
2. If a program loop causes an “out-of-space” abnormal termination (as in abend code B37) and you want a dump, specify a SYSABEND DD statement or a SYSMDUMP DD statement.
3. If you enter multiple non-ignore SLIP PER traps in IEASLPxx, only the first non-ignore SLIP PER trap will be enabled (if there were no enabled SLIP PER traps set prior to setting the traps out of IEASLPxx). The remaining non-ignore SLIP PER traps will be set, but will be disabled. The first non-ignore SLIP PER trap will be set but disabled if there was an enabled SLIP PER trap set prior to setting the traps out of IEASLPxx.
4. For a description of the SLIP command, see *z/OS MVS System Commands*.

Parameter in IEASYSxx (or supplied by the operator)

None

Syntax rules for IEASLPxx

The following rules apply to the creation of IEASLPxx:

- Begin each command on a new line.
- Data, including comments, must be contained in columns 1-71; the system ignores columns 72-80.
- Comments must begin with an asterisk "*" in column one.
- Enter the SLIP commands as you would enter them on the operator's console.

- Do not enclose the SLIP commands with double or single quotation marks or extra keywords.
- You can code multiple lines to specify a SLIP command.
- After the SET, MOD, or DEL keyword, the first blank ends the SLIP processing for that line. The system ignores all subsequent characters on that line.
- It does not matter whether the options end with a comma. If the END option has not yet been reached, the options continue with the first non-blank character on the next line.

The following is an example of a SLIP command within an IEASLPxx member:

```
SLIP SET,C=C06,
     A=WAIT,END
```

IBM-supplied default for IEASLPxx

IBM supplies member IEASLP00 in SYS1.PARMLIB. IEASLP00 contains IBM-supplied SLIP commands to suppress dumps that are considered unneeded because the system provides information (such as a message) that is normally sufficient for problem determination. For selected abend codes, the SLIP commands in IEASLP00 suppress either all dumps or specific types of dumps.

IEASLP00 contains the following commands:

```
SLIP SET,C=013,ID=X013,A=NOSVCD,J=JES2,END
SLIP SET,C=028,ID=X028,A=NOSVCD,END
SLIP SET,C=058,ID=X058,A=NODUMP,DATA=(15R,EQ,4,OR,15R,EQ,8,OR,
15R,EQ,C,OR,15R,EQ,10,OR,15R,EQ,2C,OR,15R,EQ,30,OR,
15R,EQ,3C),END
SLIP SET,C=071,SDATA=(ALLNUC,SQA,CSA,LPA,LSQA,ALLPSA,RGN,SUM,TRT),
REASON=30,ID=S071,A=SVCD,END
SLIP SET,C=071,SDATA=(ALLNUC,SQA,CSA,LPA,LSQA,ALLPSA,RGN,SUM,TRT),
REASON=20,ID=SS71,A=SVCD,END
SLIP SET,C=0E7,ID=X0E7,A=NOSVCD,END
SLIP SET,C=0F3,ID=X0F3,A=NODUMP,END
SLIP SET,C=13E,ID=X13E,A=NODUMP,END
SLIP SET,C=1C5,RE=00090004,ID=X1C5,A=NODUMP,END
SLIP SET,C=222,ID=X222,A=NODUMP,END
SLIP SET,C=322,ID=X322,A=NODUMP,END
SLIP SET,C=33E,ID=X33E,A=NODUMP,END
SLIP SET,C=3C4,REASON=1A,ID=S3C4,A=SVCD,END
SLIP SET,C=422,ID=X422,A=NODUMP,END
SLIP SET,C=422,RE=XXXX01A5,ID=X42X,A=IGNORE,END
SLIP SET,C=47B,DATA=(15R,EQ,0,OR,15R,EQ,8),ID=X47B,A=NODUMP,END
SLIP SET,C=622,ID=X622,A=NODUMP,END
SLIP SET,C=71A,ID=X71A,A=NODUMP,END
SLIP SET,C=804,ID=X804,A=(NOSVCD,NOSYSU),END
SLIP SET,C=806,ID=X806,A=(NOSVCD,NOSYSU),END
SLIP SET,C=80A,ID=X80A,A=(NOSVCD,NOSYSU),END
SLIP SET,C=81A,REASON=5,ID=X81A,A=NODUMP,END
SLIP SET,C=91A,ID=X91A,A=NODUMP,END
SLIP SET,C=9FB,ID=X9FB,A=NOSVCD,J=JES3,END
SLIP SET,C=B37,ID=XB37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=C1A,ID=XC1A,A=NODUMP,END
SLIP SET,C=D37,ID=XD37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=E37,ID=XE37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=EC6,RE=0000FFXX,ID=XEC6,A=NODUMP,END
SLIP SET,C=EC6,RE=0000FDXX,ID=XXC6,A=NOSVCD,END
```

Note:

1. IEASLP00 is read during system initialization because the IEACMD00 parmlib member contains the command SET SLIP=00. As a result, the SLIP traps specified in IEASLP00 member are in effect at IPL-time.
2. An installation that does not currently use SLIP commands should understand that the execution of IEASLP00 causes the system to allocate fixed storage for the SLIP processing modules and for the control blocks that define the SLIP traps supplied by IBM in IEASLP00. Approximately 30K bytes of storage will be fixed in the extended LPA for the SLIP processing modules and approximately 1K bytes of storage will be fixed in the extended SQA for the control blocks needed by the SLIP traps.
3. If you use a name other than JES2 or JES3 to start the job entry subsystem, the SLIP commands for abend codes X'013' and X'9FB' have no effect. Therefore, to suppress SVC dumps for these abend codes when you are not using JES2 or JES3 as the name of your job entry subsystem, place your own SLIP command in IEASLPxx. Specify NOSVCD on the ACTION=keyword (A=) and the name of your job entry subsystem on the JOBNAME=keyword (J=).

Using system commands

When you issue SET SLIP=xx, the system processes all of the commands in IEASLPxx.

To view the status of the SLIP traps, use the DISPLAY SLIP command. For more information about the SET and DISPLAY commands, see *z/OS MVS System Commands*.

Chapter 52. IEASVCxx (installation-defined SVCs)

Use IEASVCxx to define your installation's own SVCs, which can be numbered from 200 through 255.

During system initialization, the system reads the IEASVCxx members, and places any SVCs you specified in the SVC table.

To dynamically replace or delete SVC table entries use the SVCUPDTE macro. For more information, see *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.

Parameter in IEASYSxx (or entered by the operator)

```
SVC= {aa            }
     {(aa,bb,...[,L])}
```

The two alphanumeric characters (aa, bb, and so on) are appended to IEASVC to form the name of the parmlib member.

The contents of IEASVCxx appears on the operator's console if the (,L) option is specified with either the SVC= keyword or in response to the 'SPECIFY SYSTEM PARAMETERS' prompt.

Syntax rules for IEASVCxx

The following rules apply to the creation of IEASVCxx.

- Data, including comments, must appear in columns 1 through 71. Do not use columns 72-80 for data; these columns are ignored.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- One or more blanks may precede the SVCPARM statement.
- One or more blanks may follow the SVCPARM statement.
- The system processes each statement in a member sequentially.
- If you use multiple SVCPARM statements for the same SVC number, the first valid statement is processed. The rest are ignored.
- SVCNUM, REPLACE, and TYPE are required parameters for every statement.
- SVCNUM, REPLACE, and TYPE are positional parameters. SVCNUM must be coded as the first parameter with REPLACE following as the second parameter, and TYPE as the third parameter.

Syntax examples of IEASVCxx

The following example shows the coding for a type 1 and type 3 SVC:

```
SVCPARM 245,REPLACE,TYPE(1),EPNAME(SVC245),LOCKS(CMS),APF(YES)
SVCPARM 244,REPLACE,TYPE(3),EPNAME(IGC0024D),LOCKS(LOCAL)
```

IBM-supplied default for IEASVCxx

None.

Statements and parameters for IEASVCxx

SVCPARM(*svcnum***)**
Specifies the number of the installation-supplied SVC routine (a decimal number from 200 to 255). When a program issues an SVC instruction that contains this number, the system invokes the corresponding SVC routine.

,REPLACE
Specifies that an SVC table entry is to be updated.

,TYPE(*typenum***)**
Specifies the SVC type (1, 2, 3, 4, or 6) being defined.

,EPNAME(*epname***)**
Specifies the entry point name of the SVC routine. The *epname* value, if specified, must be one of the following:

- IGCERROR. Identifies an SVC routine that will result in abend F*xx* when SVC *xx* is issued, where *xx* matches *svcnum*
- Not IGCERROR.
  - For SVC types 3 and 4, *epname* must be the load module name (or alias) of a load module in the LPA.
  - For SVC types 1, 2, and 6, *epname* must be the entry point name (not the load module name, unless it is the same name) of a module in the nucleus region. The module must have been link-edited directly into the nucleus region, or added to the nucleus region through the NMLDEF macro or NUCLSTx x parmlib member.

For information about the NMLDEF macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For information about the NUCLSTxx parmlib member, see Chapter 74, "NUCLSTxx (customizing the nucleus region)," on page 689.

The EPNAME parameter is optional, unless you want to specify an EPNAME value other than the IBM default SVC naming convention.

The default naming convention for routines for SVC types 1, 2, and 6 is IGCnnn, where *nnn* is the decimal number of the SVC routine.

The default naming convention for SVC routines for SVC types 3 and 4 is IGC00nnn, where *nnn* is the signed decimal number of the SVC routine. Here, a *signed decimal* is a number that ends in either of the following ways:

- When the last digit of the SVC routine's load module name is a number from 1 - 9, specify an *epname* that ends with the EBCDIC character (A-I) that corresponds with the last digit. For example, the *epname* for a type 4 SVC 255 is IGC0025E.
- When the last digit of the SVC routine's load module name is zero, specify for the last character of *epname*, the display representation of hexadecimal C0; in EBCDIC, this is the left brace ({) character. For example, the EPNAME for a type 4 SVC 250 is IGC0025{.

,LOCKS(*lockname***,***lockname***,...)**
Specifies the name of the system locks this SVC requires. *lockname* can be either LOCAL or CMS.

Observe the following conventions for the LOCKS parameter:

- You do not need to specify the LOCAL lock for SVC type 1; the LOCAL lock is automatically obtained for SVC type 1.
- If you specify the CMS lock for SVC types 2, 3, or 4, you must also specify the LOCAL lock.
- You cannot specify a global spin lock for SVC types 3 or 4.
- You cannot specify locks for SVC type 6.

,APF{YES}|{NO }
Specifies whether the program invoking the SVC must be authorized. A program is considered to be *authorized* if it meets at least one of the following conditions:

- Runs in supervisor state
- Holds PSW key 0-7
- Is running with APF authorization.

When you specify YES, the program issuing the SVC must be authorized. When you specify NO, the program issuing the SVC can be unauthorized.

Note: NO

NPRMPT{YES} | {NO }
Specifies whether the SVC is non-preemptable (YES) or can be preempted (NO).

Note: NO

,AR{YES} | {NO }
Specifies whether the SVC can be issued in access register ASC mode.

Note: NO

Chapter 53. IEASYMxx (symbol definitions and IEASYSxx members)

IEASYMxx contains statements that:

- Define static system symbols
- Specify the IEASYSxx parmlib members that the system is to use.

You can use the HWNAME, LPARNAME, and VMUSERID parameters to limit the scope of statements in IEASYMxx. When you specify one or more of those parameters, the parameters on that statement apply only to the system that HWNAME, LPARNAME, or VMUSERID identify.

When the system finishes processing IEASYMxx, the system displays the system symbols that are defined in IEASYMxx and their associated substitution texts in message IEA009I.

Before you code IEASYMxx: Read Chapter 2, “Sharing parmlib definitions,” on page 33

Parameter in LOADxx:

To use IEASYMxx, append the 2-character identifier (aa, bb, and so forth) to the IEASYM parameter in the LOADxx parmlib member. Multiple members can be specified. If you specify a parameter in more than one IEASYMxx member, the system uses the value in the last member to be processed and ignores the values in preceding members.

See Chapter 68, “LOADxx (system configuration data sets),” on page 619 for information about specifying the IEASYM parameter in LOADxx.

Performance implications

None.

Syntax rules for IEASYMxx

The following syntax rules apply to IEASYMxx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- Blank lines are permitted.
- Delimiters are not required; the end of a physical record is considered to be a valid delimiter. However, you may optionally place a delimiter (space or comma) between a statement and a parameter.
- Multiple occurrences of a delimiter are accepted but treated as one.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- A statement can be continued even though there is no explicit continuation character.
- Do not use nested comments in this parmlib member.
- A statement type consists of 1-10 characters.

- A statement type must be the first data item on a record.
- A statement must begin with a valid statement type followed by at least one blank.
- Each parameter must consist of 1-10 characters (parameters cannot span more than one record).
- A logical record (a record that has the statement type and comments removed) cannot exceed 4 kilobytes (which is 56 - 72 character physical records).
- You can use substrings of previously defined system symbols in definitions for other system symbols. See "Step 1. Know the rules for using system symbols in parmlib" on page 51 for more information about substringing system symbols.

Syntax format of IEASYMxx

```
SYSDEF          [HWNAME(processor-name)]
                [LPARNAME(lpar-name)]
                [VMUSERID(vm-userid)]
                [SYSPARM(aa[,bb...][,L])]
                [SYSNAME(system-name)      ]
                [SYSCLONE(system-clone)    ]
                [SYMDEF(&symbol='sub-text')]
```

IBM-supplied default for IEASYMxx

None.

Statements/parameters for IEASYMxx

SYSDEF

Defines the following for one or more MVS systems in a multisystem environment:

- System symbols and their associated substitution texts
- The suffixes of the IEASYSxx parmlib members to be used.

Each SYSDEF statement can contain:

Filter Parameters

The HWNAME, LPARNAME, and VMUSERID parameters specify the system to which one or more *value parameters* (see description below) apply:

- A *local* SYSDEF statement contains at least one filter parameter; in other words, HWNAME, LPARNAME, or VMUSERID are present on the statement. The value parameters on a local SYSDEF statement apply only to the system that HWNAME, LPARNAME, or VMUSERID identifies.
- A *global* SYSDEF statement contains *no* filter parameters; in other words, HWNAME, LPARNAME, or VMUSERID are not present on the statement. The value parameters on a global SYSDEF statement apply to all systems that use IEASYMxx.

Value Parameters

The SYSPARM, SYSNAME, SYSCLONE, and SYMDEF parameters specify the static system symbols and IEASYSxx members that a system is to use. If one or more *filter parameters* precedes a value parameter, value defined on that parameter applies only to the system identified by the filter

parameters. If a filter parameter *does not* precede a value parameter, the value parameter applies to all systems that use IEASYMxx.

If two SYSDEF statements define the same symbol for the same system, the last statement processed overrides any previous definitions of that symbol. For example, suppose you code a SYSDEF statement at the beginning of IEASYMxx that defines a system symbol for all systems. Later in IEASYMxx, you code another SYSDEF statement that defines the same system symbol for a specific system identified on the HWNAME parameter. The second definition, identified by HWNAME, overrides the first.

HWNAME(processor-name)
The name (identifier) of a central processor complex (CPC), as defined to hardware configuration definition (HCD).

If a processor does not have a hardware name, and you define the processor in the same IEASYMxx member that you define other processors *with* hardware names, specify HWNAME() to indicate that a set of definitions is to apply only to the processor with no hardware name.

Value Range: The processor-name is the 1- through 8-character name of the processor on which the system is running. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$). The first character must be alphabetic (A-Z) or national (@,#,$).

Default: Match on any hardware name.

Example:

```
HWNAME(C1)              /* First test processor */

HWNAME()            /* Definitions apply only to processors */
                    /* to which no hardware name is defined */
```

Note: To use this parameter, or accept its default value, ensure that the processor on which MVS is running supports dynamic I/O configuration, and that the I/O configuration data set (IOCDS) was built using hardware configuration definition (HCD). See the section on dynamic I/O configuration in *z/OS Planning for Installation* for a list of processors that support dynamic I/O configuration.

LPARNAME(lpar-name)
The name of a *logical partition* that is defined to a processor, which is one of the following:

- The partition name specified on the "Add Partition" panel in HCD (see *z/OS HCD User's Guide* for more information).
- The partition name specified on the RESOURCE or CHPID statement that is input to the I/O configuration program (IOCP).

A value of LPARNAME() indicates that a set of definitions applies to a processor that is not initialized in LPAR mode. For example, if you run a processor sometimes in LPAR mode and sometimes in basic mode, and you define separate symbols for each mode in the same IEASYMxx parmlib member, specify LPARNAME() to indicate that the system is to process a set of definitions only when running in basic mode.

Note: If running under VM, you cannot filter using LPARNAME.

Value Range: The lpar-name is the 1- through 8-character name of a valid logical partition. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$). The first character must be alphabetic (A-Z) or national (@,#,$). Do not specify an lpar-name of all blanks.

Default: Match on any LPAR name.

Examples:

```
LPARNAME(TEST1)                      /* First test system */

LPARNAME()                         /* Processor is running */
                                 /* in non-LPAR (basic) mode */
```

VMUSERID(vm-userid)

The user ID of a Virtual Machine/Enterprise Systems Architecture (VM/ESA) system under which an MVS image is running as a guest. For information about running MVS as a VM guest, see *z/VM Running Guest Operating Systems*.

If you run a processor sometimes as a guest under VM and sometimes not, and you define both instances in the same IEASYMxx parmlib member, specify VMUSERID() to indicate that the system is to process a set of definitions only when it is not running as a guest under VM.

Note: If a system is identified by a VM user ID that is not a valid system name, the system prompts the operator to specify a valid name for the system.

Value Range: The vm-userid is a 1- through 8-character name of a valid VM system. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$).

Default: Match on any VM user ID.

Example:

```
VMUSERID(AUTOLOG1)      /* VM system of which MVS is a guest */

VMUSERID()          /* Process a set a definitions only when */
                          /* not running as a guest under VM */
```

SYSPARM(aa[,bb...][,L])

The suffixes of the IEASYSxx parmlib members to be used when this SYSDEF statement is selected.

To display the contents of IEASYSxx at the operator console when the system processes each member, specify L anywhere after the first suffix and enclose the values in parentheses. For example, specify (01,L) on SYSPARM to tell the system to process IEASYS01 and display the contents of that member at the operator console. The system ignores the SYSPARM statement if the operator specifies on the LOAD parameter that the system should prompt for system information. The operator can accomplish this by specifying an A, P, S, or T IMSI character on the LOAD parameter on the system console. For details about IMSI characters, see the section on loading the system software in *z/OS MVS System Commands*.

Note: The suffixes of IEASYSxx members can also be specified:

- On the SYSPARM parameter in the LOADxx parmlib member.
- By the operator, in response to message IEA101A SPECIFY SYSTEM PARAMETERS.

See “Step 2. Determine where to specify system parameters” on page 42 for more information.

Value Range: aa and bb are 1- through 2-character suffixes of a valid IEASYSxx parmlib members. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$). **Note**: Neither static nor dynamic system symbols are accepted.

Default: None. If you omit the SYSPARM parameter, the system uses the default member IEASYS00, the suffixes specified on the SYSPARM parameter in the LOADxx parmlib member, or the suffixes specified by the operator in response to the SPECIFY SYSTEM PARAMETERS prompt.

Example:

```
SYSPARM(00,01)          /* IEASYSxx parmlib members 00 and 01 */
```

SYSNAME(system-name)

The name to be assigned to this MVS system. The system name is also the substitution text for the &SYSNAME system symbol. The value on this parameter can be overridden by alternate parameters specified by the operator in response to the SPECIFY SYSTEMS PARAMETERS prompt.

Value Range: The system-name is a 1- through 8-character name for an MVS system. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$). Static system symbols can be used as long as the values substituted for the symbols are in the value range.

Default: For information about where to specify the system name, how the system determines which name to use, and how the default value is chosen, see "Step 3. Determine where to specify the system name" on page 44.

Note: MVS does not prevent two systems from having the same &SYSNAME value. If system resources on multiple systems are defined using &SYSNAME (such as page data sets), the results are unpredictable.

SYSCLONE(system-clone)

The value to be assigned to MVS static system system symbol &SYSCLONE; SYSCLONE is a 1 to 2 character shorthand notation for the system name.

Each system in a sysplex must specify a unique SYSCLONE value. Message IXC215I is issued if the substitution text for the &SYSCLONE symbol is not unique in a sysplex.

Value Range: Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$). Static system symbols can be used as long as the values substituted for the symbols are in the value range.

Default: The last two characters of the value specified on the SYSNAME parameter. This default is equivalent to &SYSNAME(-2:2).

Examples: Using a hard-coded value:

```
SYSCLONE(01)              /* First test system */
```

Using another static system symbol:

```
SYSNAME(S1MVS)            /* Specify name for first test system */
SYSCLONE(&SYSNAME(1:2)) /* Resolves to first 2 chars in SYSNAME */
```

SYMDEF(&symbol='sub-text')

Defines a static system symbol and its substitution text. Your installation can define at least 800 static system symbols in addition to the system symbols that MVS provides.

Value Range: &symbol is the 1-8 character name of a system symbol that your installation defines to the system. You can optionally specify an ending period on &symbol.

'sub-text' is the substitution text for the system symbol to be defined. The rules for specifying *sub-text* are:

- Enclose the text in single quotation marks (as shown in the syntax diagram).
- For &symbol, names must start with an ampersand (&). The ampersand must be followed by an alphabetic (A-Z) or national (@,#,$) character. Subsequent characters of the symbol name can be any valid alphanumeric (A-Z, 0-9) or national (@,#,$) characters. The trailing period is optional, but recommended.

  For 'sub-text', there are no restrictions on the types of characters that can be used.
- The substitution text can contain other *static* system symbols (or substrings of another static system symbol) in any order and in any combination. However, **do not specify dynamic system symbols in the substitution text.** See "Using substrings of system symbols" on page 52 for the syntax of substrings.
- When concatenating a static system symbol and a literal, terminate the static system symbol with a period.
- The substitution text can be a null value (for example, *SYMDEF(&VAR1='')* and *SYMDEF(&VAR2='&VAR1')* are both valid.)
- The length of the resolved substitution text cannot exceed the length of &symbol, *including* the ampersand on &symbol and *excluding* the single quotation marks on 'sub-text'. For example, although the length of *sub-text* exceeds the length of the symbol &FRANKIE in the following example, the symbols contained in *sub-text* resolve to 1268ABC, which is a valid substitution text:

  ```
  SYMDEF(&MARYJOE.='1234568')
  SYMDEF(&FRANKIE.='&MARYJOE(1:2).&MARYJOE(-2:2).ABC')
  ```

  However, the definition &JOHN='JOHNDOE' is not valid because the resolved substitution text JOHNDOE contains more characters than the system symbol &JOHN.

Note: Do not specify blanks in SYMDEF statements, except in *sub-text*. For example, SYMDEF(&AB='AB') and SYMDEF(&A. ='A') are not valid. SYMDEF(&AB='A B') is valid.

Default: None. If you omit the SYMDEF parameter, the specified MVS image does not use installation-defined static system symbols.

The following are examples of IEASYMxx usage:

Example 1: Define a global system symbol, called &LOGSYM, to be used on all systems. The value for the symbol will be different on different systems. First the IEASYMxx member is shown, then the value for &LOGSYM on various systems is listed.

```
SYSDEF
     SYMDEF(&LOGSYM='LOG1')  /* Define &LOGSYM for all systems */
     SYMDEF(&ABCDEF='OWL')   /* Define &ABCDEF for all systems */
     SYSPARM(BB)               /* Define SYSPARM for all systems */

SYSDEF
   HWNAME(T0)                  /* Identify a test processor      */
     SYMDEF(&LOGSYM='LOGT')  /* Override global &LOGSYM only   */
                               /* for processor T0               */
SYSDEF
   LPARNAME(R1)                /* Identify a runtime processor   */
     SYMDEF(&LOGSYM='LOGR')  /* Override global &LOGSYM only   */
                               /* for LPAR R1                    */
```

```
SYSDEF
   HWNAME(T0) LPARNAME()       /* Identify a non-LPAR processor  */
     SYMDEF(&LOGSYM='LOGN')  /* Override global &LOGSYM only   */
                               /* for T0 in non-LPAR mode        */

For this system:      &LOGSYM will have this value:
HWNAME   LPARNAME
 D0     not LPAR mode      LOG1  (only first SYSDEF matched)
 T0       R1               LOGR  (1st, 2nd and 3rd SYSDEF matched)
 T0       R2               LOGT  (1st and 2nd SYSDEF matched)
 T0     not LPAR mode      LOGN  (1st and 4th SYSDEF matched)
none      R1               LOGR  (1st and 3rd SYSDEF matched)
```

Example 2: Define a symbol using a previously defined symbol and substringing. (See "Using substrings of system symbols" on page 52 for the syntax of subsrtings.)

```
   SYMDEF(&SYMBOL2='(( ')    /* &SYMBOL2 is assigned ((    */
   SYMDEF(&SYMBOL3='3,3,3 ') /* &SYMBOL3 is assigned 3,3,3 */
 .SYMDEF(&SYMBOL1='&SYMBOL2(1:2).&SYMBOL3(1:1).))')
         /* &SYMBOL1 resolves to the first two characters in   */
         /* &SYMBOL2, the first character in &SYMBOL3, and the */
         /* two hard-coded right parentheses.  &SYMBOL1=((3))  */
```

Example 3: Use the system-provided symbol to set the initial IPL VOLSER and also use substrings to define additional IPL volumes. For this example, &SYSR1 = RESA01.

```
SYSDEF
 SYMDEF(&SYSR2='&SYSR1(1:4).02') /*SYSR2 resolves to the first */
                                 /*four characters in SYSR1 and    */
                                 /*the hard-coded 02. SYSR2=RESA02 */
 SYMDEF(&SYSR3='&SYSR1(1:4).03') /*SYSR3 resolves to the first */
                                 /*four characters in SYSR1 and    */
                                 /*the hard-coded 03. SYSR3=RESA03 */
```

Example 4: Use the system-provided symbol to set the initial IPL VOLSER and also use substrings to define additional IPL volume names, using literal text. For this example, &SYSR1 = SRSAAA.

```
SYSDEF
SYMDEF(&SYSR2.='SRS2&SYSR1(-2:2)')  /* &SYSR2 resolves to SRS2AA */
SYMDEF(&SYSR3.='SRS3&SYSR1(-2:2)')  /* &SYSR3 resolves to SRS3AA */
```

Chapter 54. IEASYSxx (system parameter list)

You can specify system parameters using a combination of IEASYSxx parmlib members and operator responses to the SPECIFY SYSTEM PARAMETERS message. You can place system parameters in the IEASYS00 member or in one or more alternate system parameter lists (IEASYSxx) to provide a fast initialization that requires little or no operator intervention.

IEASYS00 is the most likely place to put installation defaults or parameters that will not change from IPL to IPL. The system programmer can add to or modify parameters in IEASYS00. The alternate IEASYSxx members, in contrast, should contain parameters that are subject to change, possibly from one work shift to another.

Use of the IEASYS00 or IEASYSxx members can minimize operator intervention at IPL. Because IEASYS00 is read automatically, the operator can respond to SPECIFY SYSTEM PARAMETERS with ENTER or "U" and need not enter parameters unless an error occurs and prompting ensues.

The use of system parameter lists in parmlib offers two main advantages:

- The parameter lists shorten and simplify the IPL process by allowing the installation to preselect system parameters.
- The parameter lists provide flexibility in the choice of system parameters.

You can do one of the following to specify a parameter list other than IEASYS00 for an IPL:

- Have the operator specify the suffix of an alternate IEASYSxx member by replying SYSP=xx in response to the SPECIFY SYSTEM PARAMETERS message (see SYSP for more information).
- Specify one or more suffixes of alternate IEASYSxx members on the SYSPARM parameter in the LOADxx or IEASYMxx parmlib member.

For further information about IEASYSxx parmlib members and system initialization, see "Step 2. Determine where to specify system parameters" on page 42.

Overview of IEASYSxx parameters

Table 21 on page 432 summarizes all system parameters that can be placed in an IEASYSxx or IEASYS00 member (or specified by the operator). Detailed discussions of these parameters are provided in later sections of the IEASYSxx topic.

Note: PAGE and GRS are the only mandatory parameters that have no default. They must be specified.

The GRSRNL parameter is mandatory when the GRS= parameter is specified as JOIN, TRYJOIN, START, or STAR. The GRSRNL parameter is ignored when GRS=NONE.

Table 21. Overview of IEASYSxx parameters

Parameter	Use of the parameter
ALLOC	Completes the names of one or more parmlib members (ALLOCxx) that describe installation defaults for allocation parameters.
APF	Names the parmlib member (IEAAPFxx) that contains authorized library names. IEAAPFxx can specify only a static APF list, which can only be updated at IPL and contain a maximum of 255 entries.
AUTOR	Enables an installation to specify its own auto-reply policy during IPL, or to request that auto-reply processing is not to be activated.
AXR	Completes the names of one or more parmlib members (AXRxx) that specify System REXX options.
CATALOG	Specifies which IGGCATxx members to use during the current IPL.
CEA	Completes the name of the parmlib member (CEAPRMxx) that contains common event adapter (CEA) and associated instrumentation parameters.
CEE	Completes the name of the parmlib member (CEEPRMxx) that provides another level of Language Environment runtime options.
CLOCK	Completes the name of the parmlib member (CLOCKxx) that prompts the operator to initialize the TOD clock during NIP, and specifies the difference between the local time and Coordinated Universal Time (UTC).
CLPA	Causes NIP to load the link pack area with the modules contained in the LPALST concatenation. Also, CLPA purges VIO data set pages that were used in the previously initialized system. Thus, CLPA implies CVIO.
CMB	Specifies the I/O device classes for which measurement data is to be collected, in addition to the DASD and tape device classes. If the system is running on a z990 or later model processor, the system ignores anything you have specified on this parameter and creates ECMBs as needed.
CMD	Completes the name of the parmlib member (COMMNDxx) that contains commands to be issued internally during master scheduler initialization.
CON	Completes the name of the parmlib member (CONSOLxx) that centralizes control of the console configuration for your installation. CONSOLxx also contains the initialization values for communication tasks, the characteristics for the hardcopy log, and default routing codes for messages that do not have routing information. The CON parameter is also used to allow a system with both JES2 and any level of JES3 installed to run with JES2. This allows function that is incompatible with JES3 to operate successfully. The CON parameter is also used to specify the Console Services mode of operation (Distributed or Shared).
COUPLE	Completes the name of the parmlib member (COUPLExx) that describes the sysplex environment for the initializing system.
CSA	Specifies the sizes of the virtual common service area and extended common service area.
CSCBLOC	Determines the location of the CSCB control block chain.
CVIO	Deletes previously used VIO data set pages from the paging space. This parameter is automatically included when CLPA is specified.
DEVSUP	Completes the name if the parmlib member (DEVSUPxx) that specifies installation defaults for device support options.
DIAG	Completes the names of one or more parmlib members (DIAGxx) that specify whether the CSA/ECSA or SQA/ESQA tracking functions are turned on or off, whether GETMAIN/ FREEMAIN/STORAGE (GFS) trace is turned on or off, and the trace records to be included in GFS trace output.
DRMODE	Specifies that an IPL of a recovery system occurs as part of a disaster recovery scenario and that special handling of certain resources is required. This option (when YES is specified) causes System Logger to include as part of its log data recovery processing, log data from log stream DRXRC-type staging data sets that were intended only for disaster recovery purposes.

Table 21. Overview of IEASYSxx parameters (continued)

Parameter	Use of the parameter
DUMP	Specifies whether SYS1.DUMP data sets for SVC dump are to be on direct access devices. This parameter can also indicate that no SYS1.DUMP data sets are to be made available for SVC dumps.
EXIT	Completes the name of one or more parmlib members (EXITxx) that contains the entry points and names of allocation installation exits.
FIX	Completes the name of one or more parmlib members (IEAFIXxx) that contain names of modules that are to be placed in a fixed LPA that lasts for the duration of the IPL.
GRS	Specifies whether the system is to participate in a global resource serialization complex.
GRSCNF	Completes the name of the parmlib member (GRSCNFxx) that contains the information that is needed to initialize a system that is to be part of a global resource serialization complex.
GRSRNL	Completes the name of one or more parmlib members (GRSRNLxx) that contain resource name lists (RNLs) or specifies that no RNLs are to be used in the complex.
GTZ	Specifies one or more suffixes of the IBM Generic Tracker for z/OS parmlib member, GTZPRMxx, to be used by the system.
HVCOMMON	Specifies the size of the 64-bit common area.
HVSHARE	Specifies the size of the high virtual shared area.
HZS	Specifies one or more suffixes of the IBM Health Checker for z/OS parmlib member, HZSPRMxx, to be used by the system.
HZSPROC	Specifies the name of the procedure the system uses to automatically start IBM Health Checker for z/OS at IPL time.
IKJTSO	Completes the name of the parmlib member (IKJTSOxx) that contains the TSO/E settings for the system.
IOS	Completes the name of the parmlib member (IECIOSxx) that contains missing interrupt handler (MIH) statements used to modify MIH intervals and HOTIO statements used to modify recovery actions specified in the hot I/O detection table (HIDT).
IQP	Completes the names of one or more parmlib members (IQPPRMxx) that contain options to be used for managing PCIE-related devices.
IXGCNF	Completes the names of one or more parmlib members (IXGCNFxx) that specify the system logger parameters.
LFAREA	Specifies the amount of real storage to be made available for 1 MB and 2 GB pages.
LICENSE	Specifies whether the system is running with a z/OS or zNALC license.
LNK	Completes the name of one or more parmlib members (LNKLSTxx) that contain names of data sets that are to be concatenated to SYS1.LINKLIB to form the LNKLST concatenation. You can also use PROG to specify the PROGxx member that defines the LNKLST concatenation.
LNKAUTH	Specifies whether all data sets in the LNKLST concatenation are to be treated as APF-authorized or whether only those that are named in the APF table are to be treated as APF-authorized.
LOGCLS	Specifies the JES output class for the log data sets.
LOGLMT	Specifies the maximum number of WTLs (messages) for a log data set. When the limit is reached, the data set is scheduled for sysout processing.
LOGREC	Specifies the logrec recording medium for error recording.
LPA	Completes the name of one or more parmlib members (LPALSTxx) that contain names of data sets that are to be concatenated to SYS1.LPALIB for building the pageable LPA (PLPA and extended PLPA).
MAXCAD	Specifies the maximum number of SCOPE=COMMON data spaces to be allowed during an IPL.

Table 21. Overview of IEASYSxx parameters (continued)

Parameter	Use of the parameter
MAXUSER	Specifies a value that the system uses (along with the RSVSTRT and RSVNONR parameter values) to limit the number of jobs and started tasks that the system can run concurrently during a given IPL. **Note:** Specifying a higher number can cause unintended resource shortages.
MLPA	Completes the name of one or more parmlib members (IEALPAxx) that names modules that are to be placed in a modified LPA that lasts for the duration of the IPL.
MSTRJCL	Completes the name of the MSTJCLxx data set that contains the JCL used to start the master scheduler address space.
NONVIO	Designates one or more local page data sets that are not to be used for VIO paging.
NSYSLX	Specifies the number of linkage indexes (LXs), in addition to those in the system function table, to be reserved for system linkage indexes (LXs).
OMVS	Specifies the parmlib member or members (BPXPRMxx) to use to locate the parmlib statements to configure the z/OS UNIX kernel.
OPI	Indicates whether the operator is to be allowed to override particular parameters, or all parameters, contained in IEASYSxx.
OPT	Completes the name of a parmlib member (IEAOPTxx) that contains parameters to be used by various algorithms of the system resources manager.
PAGE	Gives the names of new page data sets to be used as additions to or replacements for existing page data sets. The first-named data set is used for the PLPA and extended PLPA pages. The second-named data set is used for MLPA and CSA. Alternatively, specify ***NONE*** for the first or second parameter (or both) to use Storage Class Memory (SCM) instead of a data set. The third and all subsequently named data sets are used as local page data sets. Replacement is possible only if the parameter is placed in IEASYSxx, and the operator selects this member by entering SYSP=xx. The PAGE parameter, when specified by the operator, can only add temporarily (until the next cold or quick start) to a parmlib page data set list.
PAGESCM	Specifies the minimum amount of Storage Class Memory (SCM) that should be made available for use as auxiliary storage. The system reserves the amount of SCM specified during IPL for subsequent use as auxiliary storage. Additional memory is allocated on an as-needed basis if use of the amount that is specified by the PAGESCM parameter is exceeded.
PAGTOTL	Specifies the total number of page data sets that can be allocated for the life of the IPL.
PAK	Completes the name of one or more parmlib members (IEAPAKxx) that contain groups of names of modules in the LPALST concatenation that are processed together or in sequence.
PLEXCFG	Specifies the sysplex configuration into which the system is allowed to load the initial program.
PRESCPU	Causes system initialization processing to bring logically online the CPUs that appear to MVS to be physically online, without regard to the number of CPUs defined to be initially online in the logical partition profile.
PROD	Completes the name of one or more parmlib members (IFAPRDxx) that define the enablement policy for products or product features that can be dynamically enabled under OS/390.
PROG	Completes the name of one or more parmlib members (PROGxx) that specify the format and contents of the APF-authorized library list. PROGxx can specify either a static or dynamic APF list. A dynamic format allows users to update the APF list at any time during normal processing or at IPL. You can specify as many APF-authorized libraries as you need in a dynamic APF list; there is no system-imposed maximum number. PROGxx also contains statements that control the use of installation exits and installation exit routines. You can also use PROGxx instead of LNKLSTxx to define the LNKLST concatenation and activate it at IPL.
RDE	Specifies that the reliability data extractor feature is included. For information about RDE, see *z/OS MVS Diagnosis: Tools and Service Aids*.

Table 21. Overview of IEASYSxx parameters (continued)

Parameter	Use of the parameter
REAL	Specifies the maximum amount of central storage, in 1 KB blocks, that can be allocated for concurrent ADDRSPC=REAL jobs.
RER	Specifies that the reduced error recovery procedures for magnetic tapes are in effect if they are included on the OPTCD parameter of a data definition (DD) statement or on the DCB macro instruction. If the DD statement or the DCB macro does not specify the reduced error recovery procedures, all requests for them are ignored.
RSU	Specifies the number of central storage units to be made available for storage reconfiguration (dividing storage into logical partitions under PR/SM).
RSVNONR	Specifies the number of ASVT entries to be reserved for replacing those entries marked non-reusable for the duration of an IPL.
RSVSTRT	Specifies the number of address space vector table (ASVT) entries to be reserved for address spaces that are created in response to a START command.
SCH	Specifies a parmlib member (SCHEDxx) from which the master scheduler obtains its parameters. This member centralizes control over the size of the master trace table, the program properties table (PPT), and the completion codes that are eligible for automatic restart.
SMF	Specifies a parmlib member (SMFPRMxx) from which SMF obtains its parameters.
SMS	Specifies a parmlib member (IGDSMSxx) from which SMS obtains its parameters when the system is initialized with partitioned data set extended (PDSE) support.
SQA	Specifies the size of the virtual system queue area to be created at IPL (in addition to the system's minimum virtual SQA and extended SQA).
SSN	Completes the name of the parmlib member IEFSSNxx, which contains the information to be used in identifying subsystems that are to be initialized.
SVC	Completes the name of the parmlib member IEASVCxx, which contains the information that an installation supplies to define its own SVCs. NIP processing places these SVCs in the SVC table.
SYSNAME	Specifies the name of the system being initialized. The name that is specified is used by the system in various ways. For example, • In a multi-system global resource serialization complex, SYSNAME identifies each system in the complex. • The system also uses this value to uniquely identify the originating system in messages in the multiple console support (MCS) hardcopy log and in the display created by the DISPLAY R command.
SYSP	Specifies one or more alternate system parameter lists (IEASYSxx) that are to be read by NIP in addition to IEASYS00. SYSP may be specified only by the operator.
UNI	Completes the name of the parmlib member (CUNUNIxx) that contains the Unicode statements that are required to enable Unicode Conversion services such as ADD, DELETE, REPLACE, IMAGE, and REALSTORAGE.
VAL	Names one or more parmlib members (VATLSTxx) that contain "mount" and "use" attributes of direct access devices.
VIODSN	Specifies the VSAM data set name for storing information about journaled VIO data sets.
VRREGN	Gives the default real-storage region size for an ADDRSPC=REAL job step that does not have a REGION parameter in its JCL.
WARNUND	Asks that the system warn when finding an undefined system parameter rather than prompt for new system parameters.
ZAAPZIIP (Alias: ZZ)	Determines whether the system can run zAAP processor eligible work on zIIP processors when no zAAP processors are installed on the machine.

Changes to initialization parameters

For a list of parameters that have changed or that are no longer supported by the current level of MVS, see *z/OS Migration*.

Support for system symbols

You can specify system symbols in all parameter values in IEASYSxx *except* in the values for the SYSP and OPI parameters and in specifying CLPA and OPI.

If you intend to use system symbols to represent parmlib member suffixes in IEASYSxx, be careful when defining, in the IEASYMxx parmlib member, parentheses (such as in the case of list notation) or commas as part of the substitution text:

- Specify system symbols only to the right of the equals sign and before the comma in the IEASYSxx notation.
- Specify only "balanced" parentheses in either the defined substitution text or the hard-coded values.

Table 22 shows examples of IEASYMxx and IEASYSxx notations that are valid and not valid. For more information about defining system symbols in IEASYMxx, see "Step 4. Create an IEASYMxx parmlib member" on page 44.

Table 22. Example: IEASYMxx and IEASYSxx notation

Notation	IEASYMxx	IEASYSxx
Valid; the left and right parentheses both appear in the system symbol definition.	`SYMDEF(&PAGTOTL='(10)')`	`PAGTOTL=&PAGTOTL.`
Not valid; the parentheses are split between the system symbol definition and the hard-coded definition in IEASYSxx.	`SYMDEF(&PAGTOTL='10)')`	`PAGTOTL=(&PAGTOTL.,`

Example of using system symbols in IEASYSxx: Suppose the following system symbols have the values:

- &SYSNAME = SYSA
- &SYSCLONE = SA
- &SYSPLEX = PX01

Then, assume that you want to take the following steps in IEASYSxx:

1. Specify the LNKLSTxx member identified by the last two letters in the sysplex name and also LNKLSTxx members 03 and 00.
2. Specify the CLOCKxx member identified by &SYSCLONE.
3. Specify the PROGxx member identified by the first two letters in the system name.
4. Specify a data set name for error recording that has the system name as a high-level qualifier.

Code IEASYSxx as follows:

```
LNK=(&SYSPLEX(-2:2).,03,00,L),
CLOCK=&SYSCLONE.,
PROG=&SYSNAME(1:2),
LOGREC=&SYSNAME..LOGREC
```

The values of the parameters resolve to:

```
LNK=(01,03,00,L),
CLOCK=SA,
PROG=SY,
LOGREC=SYSA.LOGREC
```

For details about how to use system symbols in parmlib, see Chapter 2, "Sharing parmlib definitions," on page 33.

Parameter specified by the operator

```
SYSP=xx
```

The operator specifies this parameter to specify an alternate system parameter list in addition to IEASYS00. (See SYSP in "Statements/parameters for IEASYSxx" on page 438.)

Syntax rules for IEASYSxx

The following rules apply to the creation of IEASYSxx:

- Use columns 1 through 71 to specify parameters. The system ignores columns 72 through 80.
- A minimum IEASYSxx member can be created by specifying a blank line as the only record. Blank lines can also be used in comments, after the last statement in the record. Otherwise, do not use blank lines within record specifications.
- Leading blanks in records are acceptable. Therefore, a parameter need not start at column 1.
- The system considers the first record that does not end in a comma to be the end of the member and ignores subsequent lines. You can use the remainder of the record, which contains the last parameter, for comments, providing there is at least one blank between the last parameter and the comments. You can also use additional lines after the last parameter for comments. Comments can only be specified in IEASYSxx as described above.
- Enter data in uppercase characters only; the system does not recognize lowercase characters.
- Use commas to separate multiple parameters in a record, but do not leave blanks between commas and subsequent parameters.
- Enclose multiple subparameters in parentheses. The number of subparameters is not limited.
- Indicate record continuation with a comma followed by at least one blank.
- Lines that begin with an asterisk in column 1 are comments.
- The system ignores anything after a comma followed by one or more blanks. You can use the remainder of the line for comments, as Figure 18 on page 438 shows.

```
    CON=01,MLPA=(00,01,02,03,L),   USE CONSOL01, IEALPA00-03
    COUPLE=&SYSCLONE,              XCF SERIAL CTCS ARE DEFINED
    PLEXCFG=MULTISYSTEM,           TURN SYSPLEX ON
    LICENSE=Z/OS,                  THE SYSTEM IS RUNNING WITH A Z/OS LICENSE
    DIAG=01                        USE DIAG01.  LAST STMT; LIST ENDS HERE
/*
/*
/*
```

Figure 18. Example of how to specify comments

IBM-supplied default for IEASYSxx

None.

Specifying the list option for IEASYSxx parameters

Certain parameters in IEASYSxx (such as CLOCK, CON, and MLPA) allow you to specify the list option (L). If you specify the L option, and message suppression is not active, the system writes all of the statements read from the associated parmlib member to the operator's console during system initialization. If message suppression is active (the default), the system writes the list to the console log only. While the L option is generally allowed, unless otherwise documented, you should not expect that any action is taken as a result of specifying it.

To ensure that messages are not suppressed, specify an appropriate initialization message suppression indicator (IMSI character) on the LOAD parameter. The IMSI characters that do not suppress messages are A, C, D, M, and T.

For more information about the LOAD parameter, see the topic on loading the system software in *z/OS MVS System Commands*.

Statements/parameters for IEASYSxx

The IEASYSxx parameters are listed alphabetically and are individually described. These parameters may optionally be issued by the operator, although such manual issuance would slow the IPL.

Note: "Value range", if applicable, means the syntactically acceptable range of values, not necessarily a range of values reasonable for function or performance. The "associated parmlib member" refers to the parmlib member that is named by the parameter. For example, IEAAPF01 is named by the APF=01 parameter in IEASYSxx, or entered by the operator.

ALLOC

ALLOC={aa} {(aa,bb,...)}

This parameter specifies the ALLOCxx members of parmlib. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to ALLOC to form the names of the ALLOCxx members. The ALLOCxx parmlib members specify allocation values for the initializing system.

Value Range: Any two alphanumeric characters.

Default Value: None.

Associated Parmlib Member: ALLOCxx

APF

APF=xx

The two characters (A-Z, 0-9, @, #, or $), represented by xx, are appended to IEAAPF to form the name of parmlib member IEAAPFxx. This member lists the data set names and volume serial numbers of authorized data sets. SYS1.LINKLIB and SYS1.SVCLIB are automatically included as authorized data sets. In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

The installation creates the default parmlib member IEAAPF00.

Note: The PROGxx parmlib member is an alternative to IEAAPFxx. If your installation decides to use PROGxx, IBM suggests that you add the PROG=xx system parameter (see PROG), remove the APF=xx system parameter from IEASYSxx, and remove APF=xx from IEASYS00. If you specify both the PROG=xx and the APF=xx parameters, the system places into the APF list the libraries listed in IEAAPFxx, followed by the libraries listed in the PROGxx members.

For information about the PROGxx parmlib member, see Chapter 76, “PROGxx (authorized program list, exits, LNKLST sets and LPA),” on page 697.

Value Range: Any two characters (A-Z, 0-9, @, #, and $).

Default Value: The system always places SYS1.LINKLIB and SYS1.SVCLIB in the APF list. If the default for the LNKAUTH system parameter is taken (LNKAUTH=LNKLST), or specified in IEASYSxx or by the operator, libraries in the LNKLST concatenation are also authorized when accessed as part of the LNKLST concatenation. If a library is in the LNKLST concatenation, but is not APF authorized, referencing this library through a JOBLIB or STEPLIB DD statement will cause the library to be considered unauthorized for the duration of the job or step, respectively.

Associated Parmlib Member: IEAAPFxx

AUTOR

```
AUTOR={xx        }
      {(xx[,xx]...)}
      {OFF       }
      {(OFF)     }
```

This parameter enables an installation to specify its own auto-reply policy during IPL, or to request that auto-reply processing is not to be activated. If auto processing is not to be activated, specify AUTOR=OFF or remove AUTOR00 from the member.

If AUTOR= is specified in response to IEA101A SPECIFY SYSTEM PARAMETERS, the AUTOR= value overrides any AUTOR= specification in the parmlib member IEASYSxx.

Value Range: Any two characters (A-Z, 0-9, @, #, and $).

Default Value: AUTOR=00

Associated Parmlib Member: AUTORxx

AXR

```
AXR={aa         }
    {(aa,bb,...)}
```

This parameter identifies the AXRxx parmlib members that specify System REXX options. The two alphanumeric characters are appended to AXR to form the name of the AXRxx member of SYS1.PARMLIB.

Syntax AXR=aa specifies a single member and syntax AXR=(aa,bb,...) for groups System REXX initialization statements across several AXRxx members.

If the same AXRxx parameter is specified in multiple members, the first occurrence takes precedence; however, the exception to this is the REXXLIB parameter that can be included in multiple members.

Value Range: Any two alphanumeric characters.

Default Value: AXR=00

Associated Parmlib Member: AXRxx

CATALOG

```
CATALOG={aa         }
       {(aa,bb,...)}
```

This parameter identifies the IGGCATxx members to use during the current IPL. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to IGGCAT to form the names of the IGGCATxx members.

The IGGCATxx parmlib members specify catalog parameters for the initializing system.

Value Range: Any two alphanumeric characters.

Default Value: CATALOG=00

Associated Parmlib Member: IGGCATxx

CEA

```
CEA={xx         }
    {(xx,yy...)}
```

This parameter identifies the CEAPRMxx parmlib member that describes the common event adapter (CEA) parameters for the initializing system. The two alphanumeric characters, xx, are appended to CEAPARM to form the name of the parmlib member, CEAPRMxx.

Value Range: Any two alphanumeric characters except "NO".

Default Value: CEA=00, causing selection of CEAPRM00.

Associated Parmlib Member: CEAPRMxx

CEE

```
CEE={xx         }
    {(xx,yy...L)}
```

The two character identifier (xx) is appended to CEEPRMxx member of SYS1.PARMLIB. If you specify the L option in the syntax of the CEEPRMxx member, or in reply to the 'SPECIFY SYSTEM PARAMETERS' message, the system writes all statements read from the CEEPRMxx member to the operator's console.

The default IEASYS00 member does not specify a CEEPRM member. If you want to use CEEPRM, you must add the CEEPRM member to IEASYS00. IBM supplies a CEEPRM00 member in SCEESAMP.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: CEEPRMxx

CLOCK

```
CLOCK={aa         }
      {(aa,bb...L)}
```

The two alphanumeric characters are appended to CLOCK to form the name of the CLOCKxx member of SYS1.PARMLIB. If you specify the L option in the syntax of the CLOCKxx member, the system writes all statements read from the CLOCKxx member to the operator's console. The member specifies whether to prompt the operator to set the TOD clock during system initialization or not, provides the difference between the local time and UTC, and controls ETR usage.

Value Range: Any two alphanumeric characters.

Default Value: CLOCK=00

Associated Parmlib Member: CLOCKxx

CLPA

CLPA

(See also the MLPA parameter for temporary additions to the LPA, and the CVIO parameter for the deletion of VIO data sets.) This parameter causes NIP to load the LPA with all modules contained in the LPALST concatenation. Modules listed in the specified LPA pack list member (IEAPAKxx) are packed together, preferably in one-page groups. (See description of IEAPAKxx.) Modules not in the pack list are loaded in size order, large modules first, then smaller modules to fill unused space.

PLPA pages are written to auxiliary storage. Only one set of PLPA pages can exist in paging space. Modules in the LPALST concatenation must be reenterable and refreshable because the system uses the processor's page protection facility, which enforces read-only access to each PLPA page.

CLPA should be specified after the installation has modified a data set in the LPALST concatenation and wants to reload the PLPA with new or changed modules.

Note: CLPA also implies CVIO, so that VIO data set pages on local page data sets are automatically purged. (See the description of CVIO for further information.)

The CLPA parameter is not needed at the *first* IPL. NIP detects the cold start condition internally, noting that the PLPA has not been loaded.

If CLPA is not specified, NIP tries to find a usable PLPA in the existing page data sets. If NIP is successful, a quick start or a warm start occurs, and the auxiliary storage manager (ASM) obtains the records that specify where the PLPA pages reside on auxiliary storage. It then reestablishes the previous set of PLPA pages. The old PLPA may be reused for any number of system initializations, if CLPA is not specified. However, page data sets that contain

the last used set of PLPA pages must be mounted. If they are not, the operator is asked to mount them. If the operator bypasses mounting, ASM initialization requests a different page data set and forces a "cold" start. NIP then reestablishes the PLPA as it does when CLPA is specified. In this cold start, both the previously established PLPA *and* existing VIO data set pages are logically deleted from paging space.

The fixed LPA and the modified LPA, however, are not automatically reused in a quick start or a warm start. They must be respecified. Existing VIO data set pages on local page data sets are retained in a warm start, unless the CVIO or CLPA parameter is forced. Such pages are not retained in a quick start or a cold start. (See the description of the CVIO parameter.)

If CLPA is specified and a set of PLPA pages already exists on a paging data set, NIP frees the existing PLPA and updates the appropriate records to reflect the new PLPA pages on auxiliary storage. NIP loads the LPA from the LPALST concatenation, as previously described.

IBM suggests that you have an IEASYSxx member that does not specify CLPA. This permits you to load the initial program without rebuilding the PLPA if it is not possible to access your LPA data sets.

Value Range: Not applicable

Default Value: Not applicable

Associated Parmlib Member: None

CMB

CMB={option}{(option,option[,option] . . .)}

This parameter allows the installation to specify I/O device classes for which measurement data is to be collected, in addition to the DASD and tape device classes. (Measurement data is always collected for DASD and tape devices.) For each I/O device to be monitored, the system needs 32 bytes of central storage.

If your system is running on a z990 processor, or higher, the system ignores anything you have specified on this parameter. Storage is allocated for ECMBs as needed. Each ECMB requires 64 bytes of central storage. CMBs will only be used when running on a processor lower than z990.

The channel stores measurement data in the CMB on a device basis. SRM uses the measurement data to perform its device selection and I/O load balancing functions.

The operator is notified if the CMB parameter contains a syntax error (message IEA926I) or if there is insufficient storage to accommodate measurement of the specified I/O device classes (message IEA340I). In both cases, SRM will prompt for respecification of the CMB parameter. If, after respecification, storage is still unavailable for measurement of the optional device classes, measurement will be done only for DASD and tape devices. Should storage also be unavailable for measurement of DASD and tape devices, SRM marks the measurement facilities as inoperative and informs the operator that device selection and I/O load balancing will be performed without I/O measurement data (messages IEA966I and IEA340I).

Operand Descriptions: One or more of the following options can be specified:

UNITR
Specifies unit record devices

COMM
Specifies communications equipment, including the channel-to-channel adapter (CTC)

GRAPH
Specifies graphics devices

CHRDR
Specifies character reader devices

n Specifies the sum of the number of devices that you need to measure that are not DASD or tape or other types of devices specified using the options **plus** the number of devices that you plan to dynamically add and measure. See *z/OS HCD Planning* for more information about specifying the CMB parameter.

Examples of Valid CMB Statements:

```
CMB=UNITR
CMB=(COMM,100)
CMB=(UNITR,CHRDR,COMM,150)
CMB=50
```

Value Range: *n* is a decimal integer from 0 through 65535.

Default Value: None

Associated Parmlib Member: None

CMD

CMD={aa}{(aa,bb...)}
The two alphanumeric characters, represented by aa (or bb, and so on), specify one or more COMMNDxx members of parmlib. The installation can specify multiple members. Each member can contain automatic operator commands that the installation wants processed during master scheduler initialization. Examples of such commands are those that start GTF. Job entry subsystem commands are not accepted because automatic commands are processed before the job entry subsystem (JES2 or JES3) is started.

If the CMD parameter is not specified, the COMMND00 member is used if it exists. If COMMND00 does not exist or cannot be read, initialization continues without any internally issued commands.

Value Range: Any two alphanumeric characters.

Default Value: CMD=00

Associated Parmlib Member: COMMNDxx

CON

```
CON={aa}
    {(aa[,L][,NOJES3][,modespec])}
    {modespec}
    {NONE}
    {(NONE[,L][,NOJES3][,modespec])}
    {NOJES3}
    {(NOJES3[,modespec])}
```

The two alphanumeric characters (aa) are appended to CONSOL to form the name of the installation-created CONSOLxx member of SYS1.PARMLIB.

If you specify the L option, the system lists all of the statements read from the CONSOLxx member on the operator's console.

If you specify NONE, the system will use all the IBM defaults for the CONSOLxx parmlib member to bring up the system. For information about when you would want to specify CON=NONE, see *z/OS MVS Planning: Operations*.

When operating JES2 on a system with JES3 installed, IBM suggests the use of the NOJES3 option to allow the comma to be omitted between the REPLY id and the command text when using short form reply.

The NOJES3 option has no effect when specified on a system that does not have JES3 installed.

The aa or NONE specification, if used, must precede the NOJES3 option. The L option, if used, can precede or follow the NOJES3 option. If you specify only NOJES3 on CON=, the system assumes you wanted (NONE,NOJES3).

The NOJES3 option, if specified, remains in effect for the duration of the IPL. If you specify NOJES3 with an error, (such as CON=(aa,NOJES3) for a CONSOLxx member that does not exist, or CON=(NONE,NOJES3) when a CONSOLxx member is required), the NOJES3 value remains in effect, regardless of how you respond to the system's prompt for CON=. To remove the NOJES3 specification, you must reload the initial program and omit the NOJES3 option when you respecify the CON= parameter.

The console support mode option *modespec* is either DISTRIBUTED or SHARED. The default is DISTRIBUTED. See *z/OS MVS Planning: Operations* for more information.

Value Range: NONE or any two characters (A-Z, 0-9, @, #, or $) and, optionally, NOJES3, DISTRIBUTED, or SHARED.

Default Value: CON=(NONE,DISTRIBUTED)

Associated Parmlib Member: CONSOLxx

COUPLE

COUPLE=xx

This parameter identifies the COUPLExx parmlib member that describes the sysplex environment for the initializing system. The two alphanumeric characters, xx, are appended to COUPLE to form the name of the parmlib member, COUPLExx. Only one suffix can be supplied.

COUPLE=** may be specified to cause XCF to initialize the system in XCF-local mode. COUPLE=** does not refer to any actual parmlib member. Instead, it is an internal function within XCF that allows the system initial program to be loaded in XCF-local mode. Specifying COUPLE=** also eliminates the need for the sysplex name specified in LOADxx to match the sysplex name specified in COUPLExx because COUPLE=** uses the sysplex name in LOADxx. If a sysplex name is not specified in LOADxx, COUPLE=** substitutes a sysplex name of LOCAL.

Value Range: Any two alphanumeric characters, or two asterisks.

Default Value: COUPLE=00, causing selection of COUPLE00.

Associated Parmlib Member: COUPLExx

CSA

CSA=(a,b)

This parameter specifies the sizes of the virtual common service area (CSA)

and extended CSA. The subparameter "a" specifies the size of the CSA, located below 16MB. The subparameter "b" specifies the size of the extended CSA, located above 16MB.

The specified size of the CSA is subtracted from the bottom of PLPA, after the bottom PLPA address is rounded down to the next 4KB (page) boundary. If the resulting virtual address for the bottom of the CSA is not on a megabyte boundary, further rounding down occurs so that the bottom CSA address is on the next megabyte boundary.

Similarly, the specified size of the extended CSA is added to the top of the extended PLPA, after the top extended PLPA address is rounded up to the next 4-KB boundary. If the resulting virtual address for the top of the extended CSA is not on a megabyte boundary, further rounding up occurs so that the top extended CSA address is on the next megabyte boundary.

The CSA (including the extended CSA) is an address range in each address space that is used for common system functions (functions not related to a particular address space). For example, the system allocates buffers for LOG and SMF from the CSA.

In selecting values for the CSA parameter, understand that the system's process of rounding to a 1MB boundary can cause up to 1MB of storage from the private area to be allocated to the CSA. However, consider the following:

- If the virtual storage manager runs out of SQA, it will try to obtain space from the CSA.
- A large CSA size will reserve space for future LPA growth. Such growth would be hampered if users were allowed to obtain very large private areas. A large CSA specification effectively limits the maximum private area that a user job can acquire.

If a shortage occurs in CSA, ECSA, SQA, or ESQA, you can use the storage tracking function to collect information about jobs or address spaces that own storage in those areas. With that information, you can identify jobs or address spaces that obtain an excessive amount of storage. If those jobs or address spaces have code to free the storage when they are canceled, you might relieve the shortage and avoid an IPL if you use an operator command to cancel those jobs or address spaces.

When you turn the storage tracking function on, you might experience a small performance degradation and an increase in ESQA usage. For more information about common storage tracking, including how to turn the function on or off, see Chapter 30, “DIAGxx (control common storage tracking and GFS trace),” on page 323.

Note: If you allocate excessive amounts of CSA or SQA, the system generates a warning message, and you must respecify the CSA parameter. The system also generates a warning message when the size of the entire common area below 16MB exceeds 8MB.

Value Range: Each **a** value can be expressed as:

- A decimal number, n, indicating n 1KB (1024-byte) blocks. The number is 0 through 9999.
- A decimal number followed by K, nK, indicating n 1KB blocks. The number is 0 through 9999.
- A decimal number followed by M, nM, indicating n 1MB (1024*1024-byte) blocks. The number is 0 through 9.

Each **b** value can be expressed follows. Note that the maximum values are accepted, but not recommended because they would result in a private region that is too small. Do not specify more than you think you might ever need.

- A decimal number, n, indicating n 1KB blocks. The number is 0 through 2080767.
- A decimal number followed by K, nK, indicating n 1KB blocks. The number is 0 through 2080767.
- A decimal number followed by M, nM, indicating n 1MB blocks. The number is 0 through 2031.

Default Range: For each subparameter, 100 through 1023KB (depending on the amount of storage added because of rounding to a segment boundary).

Associated Parmlib Member: Not applicable.

CSCBLOC

CSCBLOC={ABOVE| BELOW}

This parameter determines whether the CSCB control block chain resides above or below 16 megabytes.

The CSCB is comprised of a 36-byte block which is always obtained below 16 megabytes and a 256-byte block whose residence is governed by this parameter. Be careful when specifying CSCBLOC=BELOW to not over-specify the MAXUSER or RSVSTRT values by too great an amount as they are used to determine the number of CSCBs allocated. At a minimum, over 9000 bytes are allocated below the line and additionally over 64000 bytes are allocated based on the CSCBLOC setting. Using large values for MAXUSER or RSVSTRT reduces the amount of CSA storage below 16MB available for other applications. This might result in an ABEND878 when applications are started after IPL or cause a wait state 040 to occur at IPL time. For example, if you specify a combined MAXUSER and RSVSTRT value of 5000, over 22500 bytes are initially allocated below 16 megabytes and additionally over 160000 bytes are initially allocated based on the CSCBLOC setting.

Value Range: Not applicable

Default Value: CSCBLOC=ABOVE

Associated Parmlib Member: None

CVIO

CVIO (Clear VIO)

This parameter specifies that all VIO data set pages on auxiliary storage are to be deleted from page space. A typical application would be the purging of VIO data set pages when the system initial program is reloaded after a previous end-of-day (EOD).

Note: If you want the auxiliary storage manager (ASM) to purge and reinitialize *both* the VIO data set pages *and* the PLPA pages, specify CLPA. CLPA always implies CVIO.

If you do not specify CVIO, a warm start IPL occurs and VIO data set pages are retained for restart processing. (Such restart would be possible for some data sets after a temporary system failure.) ASM reestablishes the VIO data set pages that had checkpoints set before the system failure. This action, of course, does not ensure that the job entry subsystem will reuse these data sets.

If one or more volumes that contain VIO pages are not mounted on a warm start IPL, ASM requests the operator to mount the missing volumes. If the operator does not mount all the requested volumes that contain VIO pages, ASM deletes all VIO data set pages, just as if CVIO or CLPA had been specified. The operator receives a message that indicates that CVIO has been forced.

Value Range: Not applicable

Default Value: None

Associated Parmlib Member: None

DEVSUP

```
DEVSUP={aa        }
       {(aa,bb...)}
```

This parameter specifies one or more DEVSUPxx parmlib members. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to DEVSUP to form the name of the DEVSUPxx members.

The DEVSUPxx member specifies the installation default for whether data will be stored in a compacted format on a 3480 or 3490 tape subsystem with the Improved Data Recording Capability feature.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: DEVSUPxx

DIAG

```
DIAG={aa         }
     {(aa,bb,...)}
```

This parameter specifies the DIAGxx members of parmlib. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to DIAG to form the names of the DIAGxx members. The DIAGxx parmlib member:

- Turns the common storage tracking function on or off
- Controls GFS tracing.

Value Range: Any two alphanumeric characters.

Default Value: DIAG00

Associated Parmlib Member: DIAGxx

DRMODE

```
DRMODE={NO}|{YES}
```

This parameter indicates whether a recovery system is to be loaded during IPL as part of a disaster recovery scenario and special handling of certain resources is required.

When DRMODE=NO is specified, or defaulted, the z/OS system initial program is being loaded without requesting special resource handling for disaster recovery purposes. The **NO** specification or default can normally be used for most system IPLs.

The **NO** option causes z/OS MVS System Logger to **NOT include** DRXRC-type staging data sets in its log data recovery for coupling facility structure-based log streams that had been connected prior to this IPL.

When DRMODE=YES is specified, it indicates that the initial program a recovery system is being loaded as part of a disaster recovery scenario and special handling of certain resources are required. The **YES** specification is intended to be used when the installation had previously configured their sysplex for specific disaster recovery capabilities.

The **YES** option causes z/OS MVS System Logger to **include** DRXRC-type staging data sets in its log data recovery for coupling facility structure-based log streams that had been connected prior to this IPL. That is, Logger will attempt to recover log data for log streams with the STG_DUPLEX(YES),DUPLEXMODE(DRXRC) specification that had been connected prior to the IPL with the DRMODE=YES option. Also refer to Logger confirmation message IXG068D which is issued when the DRMODE=YES option is specified. When this option is specified, it is assumed that the necessary actions had been taken to establish the DASD consistency groups related to the Logger data sets. For more details refer to DRXRC Considerations and DRMODE=YES IPL Option in *z/OS MVS Setting Up a Sysplex*.

Value Range: Not applicable.

Default Value: NO

Associated Parmlib Member: None

DUMP

```
DUMP={NO          }
     {DASD        }
     {(DASD,xx-yy)}
```

This parameter specifies whether SYS1.DUMP data sets on direct access devices are to be made available at IPL time. SVC dump options are not included in IEASYSxx. The installation can specify the options, if it so desires, through the CHNGDUMP operator command, either in the COMMNDxx parmlib member or from the console.

When planning for dump data sets the installation should be aware the dump data sets can sometimes contain privileged data. By using protected data sets (through passwords or other security methods), the installation can limit access.

Note: You can also allow the system to create dump data sets dynamically. For details, see *z/OS MVS Diagnosis: Tools and Service Aids*.

Operand Descriptions:

NO specifies that no dump data sets will be made available for SVC dump at IPL time.

Note: Dump data sets can be specified after IPL by using the DUMPDS command or by adding the DUMPDS command to COMMNDxx.

DASD
specifies that the all currently cataloged SYS1.DUMPnn data sets (if any), on permanently resident direct access volumes, are to be used. The catalog will be scanned for SYS1.DUMP00 through SYS1.DUMP99. DASD is the default if the DUMP parameter is omitted.

Note: Specifying DASD is equivalent to specifying `DUMPDS ADD,DSN=ALL` in the COMMNDxx parmlib member.

(DASD,xx-yy)
specifies that the currently cataloged SYS1.DUMPnn data sets (if any), on permanently resident direct access volumes, are to be used. The catalog will be scanned for SYS1.DUMPxx through SYS1.DUMPyy, where xx and yy are decimal digits in the range 00 through 99.

Note: Specifying DASD is equivalent to specifying `DUMPDS ADD,DSN=(xx-yy)` in the COMMNDxx parmlib member.

Indicating which dump data sets are to be used by a particular system avoids unnecessary scanning of the possible 100 cataloged dump data sets and the possibility of more than one system using the same data sets.

Examples of Valid DUMP Statements:

```
DUMP=NO
DUMP=DASD
DUMP=(DASD,00-05)
```

How Dump Data Sets Are Used: Dump data sets can only reside on direct access devices. Space for direct access data sets must be pre-allocated, and the data sets must be cataloged. Eligible device types consist of any direct access device supported by the system that has a track size of at least 4160 bytes (4160 bytes equals 1 SVC dump output record).

As many as 100 dump data sets may be allocated. They must be in the form SYS1.DUMPnn, in which nn may be digits 00 to 99.

Note: Specify both primary and secondary allocations for SYS1.DUMPnn data sets. IBM suggests using the DUMPDS ADD command in COMMNDxx and DUMP=NO in IEASYSxx to make the allocated dump data sets available to SVC dump. If you do this, MVS provides better diagnostic messages, which indicate which dump data sets were added, which were not added, and why.

For more information about allocating SYS1.DUMPxx data sets, see *z/OS MVS Diagnosis: Tools and Service Aids* .

Processing of Dump Data Sets: The status and type of each dump data set is maintained by the system. The system records the data set status as empty or full. An empty data set is available for use by the system. A full data set can be printed (and emptied), or made empty by the DUMPDS CLEAR command.

A DASD data set is empty only if the first record is an end of data record. Otherwise, the data set is considered full.

When an SVC dump is requested, an empty data set is selected and that data set is then marked as being in use. When the DUMPDS CLEAR command is issued for a dump data set, that data set is marked as being available.

When an SVC dump is requested and there are no data sets marked as being available, the system reads the first record of each data set to see if it has been emptied (printed). If so, the first record is an end of data record. When the system finds such a data set, it uses it for the requested SVC dump.

Note: Tape data sets are not supported.

Value Range: Not applicable

Default Value: DASD

Associated Parmlib Member: None

EXIT

```
EXIT=aa
```

This parameter specifies the EXITxx members of parmlib that contains the entry points and names of allocation installation exits. The two alphanumeric characters, represented by aa, are appended to EXIT to form the name of the EXITxx members.

Note: IBM provides the PROGxx member as an alternative to EXITxx. IBM suggests that you convert the format of EXITxx to PROGxx and provides the IEFEXPR REXX exec to do this. You should also add the PROG=xx system parameter to IEASYSxx (see Chapter 76, "PROGxx (authorized program list, exits, LNKLST sets and LPA)," on page 697), remove the EXIT=xx system parameter from IEASYSxx, and remove EXIT=xx from IEASYS00. For information about how to begin using PROGxx, see Chapter 76, "PROGxx (authorized program list, exits, LNKLST sets and LPA)," on page 697.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: EXITxx

FIX

```
FIX={aa                    }
    {(aa[,L][,NOPROT])     }
    {(aa,bb...[,L][,NOPROT])}
```

This parameter specifies one or more IEAFIXxx members of parmlib. The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so on), are appended to IEAFIX to name the members. If the L option is specified, the system displays the contents of the IEAFIXxx parmlib members at the operator's console as the system processes the members.

The members contain names of modules that are to be fixed in central storage as a fixed LPA. The fixed LPA modules are active only for the duration of an IPL, and will not be automatically reinstated by a quick start or a warm start IPL. You must respecify the FIX parameter in later IPLs if you want to reinstate the fixed LPA.

The LPA modules that are fixed in storage are also page protected, by default. If an attempt is made to store into a page-protected module, a protection exception occurs. However, an installation can use the NOPROT option to override the page protection default. When NOPROT is specified, the LPA modules in the IEAFIXxx parmlib members are not page protected in storage.

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: None

Associated Parmlib Member: IEAFIXxx

GRS

```
GRS={JOIN   }
    {TRYJOIN}
    {START  }
    {NONE   }
    {STAR   }
```

This parameter specifies whether the system being initialized is to participate in a global resource serialization complex. In a multisystem sysplex, every

system in the sysplex must be in the same global resource serialization complex. This allows global serialization of resources in the sysplex.

Specifying GRS=STAR indicates that the system being initialized is to participate in a global resource serialization star complex.

GRS=NONE indicates that the system is not to participate in a global resource serialization complex. GRS=START, GRS=JOIN, and GRS=TRYJOIN indicate that the system is to coordinate global resource requests with other systems in a global resource serialization ring complex.

Specifying GRS=START indicates that the system is to start a global resource serialization ring complex and honor the requests of other systems (those systems that specify GRS=JOIN) to join the complex. Specifying GRS=JOIN indicates that the system is to join an existing ring complex of active global resource serialization systems and also honor the requests of other systems to join the complex.

Specifying GRS=TRYJOIN allows the system to act on whether a ring complex already exists. If the complex does exist, the system joins the existing complex. If the system does not find any active global resource serialization systems, the system will start the complex.

Attention: When all the systems in a global resource serialization ring complex are not in a multisystem sysplex, there is a possibility of a split ring and a data integrity exposure when GRS=TRYJOIN is used. When all systems are in a multisystem sysplex, however, this option is recommended. For more information about specifying the GRS system parameters, see *z/OS MVS Planning: Global Resource Serialization*.

Note: If you have specified anything other than GRS=NONE, then you must specify the GRSRNL parameter.

Value Range: Not applicable

Default Value: None

Associated Parmlib Member:

GTZ

```
GTZ={aa   }
    {(aa,bb...)}
```

This parameter specifies one or more suffixes of the optional Generic Tracker for z/OS parmlib member GTZPRMxx to be used by the system. The two characters, represented by *aa* (or *bb*, and so on) are appended to GTZPRM to form the name of the GTZPRMxx members when the Generic Tracker address space, GTZ, starts.

The GTZPRMxx parmlib member contains parameters for managing IBM Generic Tracker for z/OS, including EXCLUDE statements for known track events.

In order to let the system successfully process the identified GTZPRMxx parmlib members at GTZ startup the following security setup is required:

- Have a user ID with permission to access the PARMLIB concatenation, for example using:
  - ADDUSER GTZ NOPASSWORD
  - ADDSD 'SYS1.PARMLIB' UACC(NONE)

- PERMIT 'SYS1.PARMLIB' CLASS(DATASET) ID(GTZ) ACCESS(READ)
- Associate this user ID with the GTZ address space, for example using:
  - SETROPTS GENERIC(STARTED)
  - RDEFINE STARTED GTZ.* STDATA(USER(GTZ))
  - SETROPTS RACLIST(STARTED) REFRESH

Value Range: Any two alphanumeric or national ($,#,@) characters.

Default Value: None

Associated Parmlib Member: GTZPRMxx

GRSCNF

```
GRSCNF=xx.
```

This parameter identifies the GRSCNFxx parmlib member to be used to initialize a system in the global resource serialization complex. The two alphanumeric characters, xx, are appended to GRSCNF to form the name of the parmlib member, GRSCNFxx. Only one suffix can be supplied. The installation creates the member and places it in parmlib.

The majority of GRSCNFxx parameters are specific to defining a global resource serialization ring complex. However, GRSQ is specific to star mode, and the following keywords are relevant for all modes, including GRS=NONE:

- ENQMAXA
- ENQMAXU
- SYNCHRES

Value range: Any two alphanumeric characters.

Default: GRSCNF00 is the parmlib member selected if you do not specify GRSCNF=xx.

Associated parmlib member: GRSCNFxx

GRSRNL

```
GRSRNL={aa        }
       {(aa,bb...)}
       {EXCLUDE   }
```

This parameter can specify one or more GRSRNLxx parmlib members. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to GRSRNL to form the name of the GRSRNLxx members.

The GRSRNLxx members contain resource name lists (RNLs). The system (specifically, global resource serialization) uses the RNLs to determine how to treat a resource that the installation defined in an RNL.

GRSRNL=EXCLUDE specifies that no resource name lists (RNLs) are to be used in the complex. All ENQ, RESERVE, and DEQ macro requests with a scope of SYSTEMS are treated as though they had been found in the SYSTEMS exclusion RNL. Their scope is changed to SYSTEM and they are processed locally.

The RNL=NO parameter, if specified on the request, prevents this exclusion. It allows the request to retain its scope of SYSTEMS and be serialized globally on all systems in the complex. See *z/OS MVS Planning: Global Resource Serialization* when to use GRSRNL=EXCLUDE.

Note: If you specify anything other than GRS=NONE during system initialization, the GRSRNL parameter is required.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: GRSRNLxx

HVCOMMON

```
HVCOMMON ={xxxxxxxxxxG}
                      {xxxxxxxxxxT}
```

This parameter specifies the size of the 64-bit common area. The IARV64 macro allows the allocation of common virtual storage above 2 gigabytes. The HVCOMMON parameter allows you to define the size of the 64-bit common area.

The 64-bit common area will be placed below the 4T line.

The value you specify will be rounded up to a 2 gigabyte boundary.

You can also specify the HVCOMMON value by responding to message IEA101A.

Value Range: 2 gigabytes (2G) to 1 terabyte (which is 1024 gigabytes)

Default Value: 64 Gigabytes (64G)

HVSHARE

```
HVSHARE={xxxxxxxxxxG}
        {xxxxxxxxxxT}
        {xxxxxxxxxxP}
```

This parameter specifies the size of the high virtual shared area. The IARV64 macro allows multiple address spaces to share virtual storage above 2 gigabytes (see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*for details). You can define how much virtual storage can be shared.

The size of the HVSHARE value determines where the system places the virtual shared area. If you specify a value less than 2 terabytes (2 T), the system obtains storage that straddles the 4-terabyte line. Half of the storage comes from above the 4-terabyte line, and half of the storage comes from below the 4-terabyte line. If you specify a value larger than 2 T, the system obtains storage starting at the 2-terabyte line.

The value that you specify is rounded up to a 64-gigabyte boundary.

You can also specify the HVSHARE value by responding to message IEA101A.

Value range: 0 gigabytes (0 G) to 1 exabyte (which is 1048576 terabytes or 1024 petabytes)

Default: 510 terabytes (510 T); placement will start at the 2 T line.

Associated parmlib member: None.

HZS

```
HZS={aa       }
    {(aa,bb...)}
```

This parameter specifies one or more suffixes of the optional IBM Health

Checker for z/OS parmlib member, HZSPRMxx to be used by the system. The two characters, represented by *aa* (or *bb*, and so on), are appended to HZSPRM to form the name of the HZSPRMxx members when HZSPRM=SYSPARM (or HZSPRM=PREV for IPL-time) is specified in the HZSPROC startup procedure for IBM Health Checker for z/OS.

The HZSPRMxx parmlib member contains parameters for managing IBM Health Checker for z/OS and checks, including overriding check defaults and defining policies for permanent updates to check defaults.

Value Range: Any two alphanumeric or national ($,#,@) characters.

Default Value: None

Associated Parmlib Member: HZSPRMxx

HZSPROC

HZSPROC=*hzsprocname*

This parameter specifies the name of the HZSPROC procedure you want the system to use to automatically start IBM Health Checker for z/OS at IPL-time. Specify this parameter if you want to use a name other than the default, HZSPROC. The procedure specified must reside in a SYS1.PROCLIB data set.

You can customize your HZSPROC procedure - see Optimizing IBM Health Checker for z/OS in *IBM Health Checker for z/OS User's Guide*.

Value Range: A 1-8 character string, where the first character is alphabetic or national ($,#,@) and the other characters, if any, are alphanumeric or national.

Default Value: HZSPROC

Associated Parmlib Member: None

IKJTSO

IKJTSO=xx

This parameter specifies the parmlib member from which TSO/E settings are obtained. The two alphanumeric characters (xx), specified in the parameter, are appended to IKJTSO to form the parmlib member IKJTSOxx.

Value Range: Any two alphanumeric characters

Default Value: 00 (Specifies IKJTSO00)

Associated Parmlib Member: IKJTSOxx

IOS

IOS=xx

This parameter specifies the parmlib member that contains (1) intervals to be used by the missing interrupt handler (MIH) when scanning for missing interrupt conditions and (2) device threshold values and recovery actions to be used in the detection of, and recovery from, hot I/O conditions. The two alphanumeric characters (xx), specified in the parameter, are appended to IECIOS to form the parmlib member IECIOSxx.

Value Range: Any two alphanumeric characters

Default Value: None

Note: The I/O supervisor assumes default options. For a list of these defaults and their explanations see the description of the IECIOSxx parmlib member.

Associated Parmlib Member: IECIOSxx

IQP

```
IQP={aa       }
    {(aa,bb...)}
```

This parameter specifies the IQPPRMxx members of parmlib. The two alphanumeric characters, represented by aa (or bb, and so forth) are appended to IQPPRM to form the names of the IQPPRMxx members. The IQPPRMxx parmlib members specify parameters that are used for managing PCIE related devices.

Value Range: Any two alphanumeric characters

Default Value: None

Associated Parmlib Member: IQPPRMxx

IXGCNF

```
IXGCNF={aa       }
      {(aa,bb...)}
```

This parameter identifies the IXGCNFxx parmlib members to be used when system logger starts or is restarted on the initializing system within the sysplex. Syntax IXGCNF=aa specifies a single member and syntax IXGCNF=(aa,bb,....) identifies groups of system logger initialization statements across several IXGCNFxx members.

The installation can optionally create one or more members and place them in SYS1.PARMLIB.

When multiple IXGCNFxx members are concatenated, the individual system logger options are merged with the last parmlib member option taking precedence. The Display LOGGER,IXGCNF[,*options*] command displays the merged information

Value Range: Any two alphanumeric characters

Default Value: None

Note: The L (list) option is syntactically allowed but ignored.

Associated Parmlib Member: IXGCNFxx

LFAREA

```
LFAREA = {xM | xG | xT | x%}
         {(xM | xG | xT | x% [,INCLUDE1MAFC])}
         {( [1M=(a [,b]) | 1M=(a% [,b%])]
            [,2G=(a [,b]) | ,2G=(a% [,b%])]
            [,prompt | ,noprompt]
            [,INCLUDE1MAFC]
          )}
```

The LFAREA parameter specifies the amount of online real storage available at IPL to reserve for backing 1 MB pages and 2 GB pages. The *x*M, *x*G, *x*T, and *x*% syntax form reserves 1 MB pages, and the 1M= and 2G= syntax form reserves 1 MB and 2 GB pages. The two syntax forms cannot be combined. Each syntax specification uses a different formula for calculating percentage requests and the system limit, as

described in "Request handling for the large frame area system limit" on page 459 and "Request handling for insufficiently contiguous online real storage" on page 459.

In the following descriptions, 2 GB is equal to a decimal value of 2^{31}, or 2147483648, and 4 GB is equal to a decimal value of 2^{32}, or 4294967296.

Value range: The following value ranges apply to the value specified on the LFAREA parameter. The maximum values that can be specified for the LFAREA are described in "Request handling for the large frame area system limit" on page 459.

xM or xG or xT
Specifies the amount of online real storage to reserve for 1 MB pages. Up to six decimal digits can be specified for *x*: *x*M specifies the amount is in megabytes; *x*G specifies the amount is in gigabytes; *x*T specifies the amount in terabytes. A specification of 0M, 0G, or 0T results in zero 1 MB pages being reserved.

x% Specifies the amount of online real storage to reserve for 1 MB pages, as a percentage of the total online real storage at IPL. Up to two digits can be specified. The percentage is calculated as (*x*% * online real storage at IPL) – 2G. The maximum percentage that can be specified is 80%. A specification of 0% results in zero 1 MB pages being reserved.

1M=(*a*)
Specifies the number of 1 MB pages of online real storage to reserve in the large frame area. Up to eight decimal digits can be specified for *a*. A specification of 1M=(0) results in zero 1 MB pages being reserved.

1M=(*a*,*b*)
Specifies the number of 1 MB pages of online real storage to reserve in the large frame area. Up to eight decimal digits each can be specified for *a* and *b*. The value that is specified for *a* is the target number of pages, and the value that is specified for *b* is the minimum number. The system attempts to meet the request at or as near as possible up to the target number, but at no less than the minimum number. The value that is specified for *b* must be less than or equal to the value specified for *a*, and can be zero. A specification of 1M=(0,0) results in zero 1 MB pages being reserved. Once the LFAREA parameter has been processed, no additional amounts of storage are reserved later in an attempt to reach the target. Both a value and a percentage, such as 1M=(*a*,*b*%) or 1M=(*a*%,*b*), cannot be specified.

1M=(*a*%)
Specifies the amount of online real storage to reserve in the large frame area for 1 MB pages, as a percentage. Up to two digits can be specified. The storage amount is calculated as a% * (online real storage at IPL – 4G), rounded down to the nearest 1 MB boundary. The maximum percentage that can be specified is 80%. A specification of 1M=(0%) results in zero 1 MB pages being reserved.

1M=(*a*%,*b*%)
Specifies the amount of online real storage to reserve in the large frame area for 1 MB pages, as a percentage. Up to two digits each can be specified for *a*% and *b*%. The value specified for *a*% is the target amount, and the value specified for *b*% is the minimum amount. The system attempts to satisfy the request at or as near as possible up to the target amount, but at no less than the minimum amount. The percentages are calculated as percent * (online real storage at IPL – 4G), rounded down to the nearest 1 MB boundary. The maximum percentage that can be specified for *a*% is 80%. The value specified for *b*% must be less than or equal to the value for *a*%, and can be zero. A

specification of 1M=(0%,0%) results in zero 1 MB pages being reserved. Once the LFAREA parameter has been processed, no additional amounts of storage are reserved later in an attempt to reach the target. Both a value and a percentage, such as 1M=(*a*,*b*%) or 1M=(*a*%,*b*), cannot be specified.

2G=(*a*)
Specifies the number of 2 GB pages of online real storage to reserve in the large frame area. Up to eight decimal digits can be specified for *a*. A specification of 2G=(0) results in zero 2 GB pages being reserved. Each 2 GB page is placed only in a 2 GB area of contiguous real storage on a 2 GB boundary, which means that enough 2 GB areas of contiguous online storage on 2 GB boundaries must be available at IPL for the number of 2 GB pages requested.

2G=(*a*,*b*)
Specifies the number of 2 GB pages of online real storage to reserve in the large frame area. Up to eight decimal digits each can be specified for *a* and *b*. The value specified for *a* is the target number of pages, and the value specified for *b* is the minimum number. The system attempts to satisfy the request at or as near as possible up to the target number, but at no less than the minimum number. The value specified for *b* must be less than or equal to the value for *a*, and can be zero. A specification of 2G=(0,0) results in zero 2 GB pages being reserved. Once the LFAREA parameter has been processed, no additional amounts of storage are reserved later in an attempt to reach the target. Both a value and a percentage, such as 2G=(*a*,*a*%) or 2G=(*a*%,*a*) cannot be specified. Each 2 GB page is placed only in a 2 GB area of contiguous real storage on a 2 GB boundary, which means that enough 2 GB areas of contiguous online storage on 2 GB boundaries must be available at IPL for at least the minimum number of 2 GB pages requested.

2G=(*a*%)
Specifies the amount of online real storage to reserve in the large frame area for 2 GB pages, as a percentage. Up to two digits can be specified for *a*. The storage amount is calculated as *a*% * (online real storage at IPL – 4G), rounded down to the nearest 2 GB boundary. The maximum percentage that can be specified is 80%. A specification of 2G=(0%) results in zero 2 GB pages being reserved. Each 2 GB page is placed only in a 2 GB area of contiguous real storage on a 2 GB boundary, which means that enough 2 GB areas of contiguous online storage on 2 GB boundaries must be available at IPL for the amount of 2 GB pages requested.

2G=(*a*%,*b*%)
Specifies the amount of online real storage to reserve in the large frame area for 2 GB pages as a percentage. Up to two digits each can be specified for *a*% and *b*%. The value specified for *a*% is the target amount, and the value specified for *b*% is the minimum. The system attempts to satisfy the request at or as near as possible up to the target amount, but at no less than the minimum amount. The percentages are calculated as percent * (online real storage at IPL – 4G), rounded down to the nearest 2 GB boundary. The maximum percentage that can be specified for *a*% is 80%. The value specified for *b*% must be less than or equal to the value for *a*%, and can be zero. A specification of 2G=(0%,0%) results in zero 2 GB pages being reserved. Once the LFAREA parameter has been processed, no additional amounts of storage is reserved later in an attempt to reach the target. Both a value and a percentage, such as 2G=(a,b%) or 2G=(a%,b), cannot be specified. Each 2 GB page is placed only in a 2 GB area of contiguous real storage on a 2 GB

boundary, which means that enough 2 GB areas of contiguous online storage on 2 GB boundaries must be available at IPL for at least the minimum amount of 2 GB pages requested.

prompt | noprompt
Specifies whether the system prompts for re-specification of the LFAREA parameter when the request cannot be satisfied. The default is **prompt**, which specifies for the system to prompt when either the 1M= or 2G= request cannot be satisfied. The value **noprompt** specifies for the system to ignore the LFAREA request when either or both of the 1M= and 2G= requests cannot be satisfied, and to instead reserve zero 1 MB pages and zero 2 GB pages in the large frame area.

INCLUDE1MAFC
Specifies that 1 MB pages are to be included in the available frame count (RCEAFC).

With APAR OA41968 installed, specifying INCLUDE1MAFC results in the following system behavior:

- RSM performs less paging when there is an abundance of available fixed 1 M pages in the system.
- RSM is more likely to break up fixed 1 M pages to satisfy 4 K page demand. Although RSM attempts to coalesce broken up fixed 1 M pages when there is fixed 1 M page demand, there is no guarantee that coalescing will be successful, especially if one of the 4 K frames making up the fixed 1 M page is fixed long term.

Application programs can check the RCEINCLUDE1MAFC bit to determine if the installation specified INCLUDE1MAFC in their LFAREA specification.

For RMF users, the PTF for RMF APAR OA42510 must be applied prior to specifying INCLUDE1MAFC. RMF uses the RCEAFC to generate some of their reports. Not applying OA42510 might lead to incorrect RMF reports.

In addition, application programs that use the STGTEST SYSEVENT need to be assessed to determine if any changes need to be made. STGTEST returns information about the amount of storage available in the system. If INCLUDE1MAFC is specified, available fixed 1 M pages are included in this amount.

Examples:

1. `LFAREA=(64M,INCLUDE1MAFC)`

   Using the xM|xG|xT|x% syntax, INCLUDE1MAFC is a positional parameter and must be coded after the xM|xG|xT|x% specification.
2. `LFAREA=(20%,INCLUDE1MAFC)`

   Using the xM|xG|xT|x% syntax, INCLUDE1MAFC is a positional parameter and must be coded after the xM|xG|xT|x% specification.
3. `LFAREA=(1M=64,INCLUDE1MAFC)`

   Using the 1M= syntax, INCLUDE1MAFC can be specified anywhere within the parentheses.
4. `LFAREA=(INCLUDE1MAFC,1M=20%,NOPROMPT)`

   Using the 1M= syntax, INCLUDE1MAFC can be specified anywhere within the parentheses.

Request handling for the large frame area system limit

The system limits the amount of online real storage that can be reserved in the large frame area for 1 MB and 2 GB pages. The system applies this limit differently depending on which of the two syntax forms is used for the LFAREA request. The two syntax forms cannot be combined.

If the LFAREA request is specified using *x*M, *x*G, *x*T, or *x*% syntax, to reserve 1 MB pages:

- The system limit is calculated as: (80% * online real storage available at IPL) – 2G
- The value specified by *x*M, *x*G, *x*T, or *x*% is applied toward the limit, with the percentage calculated as (*x*% * online storage available at IPL) – 2G.
- If the limit is exceeded, the system rejects the request and prompts for re-specification of the LFAREA request.

If the LFAREA request is specified using the 1M= or 2G= syntax form, to reserve 1 MB or 2 GB pages (or both):

- The system limit is calculated as: 80% * (online real storage available at IPL – 4G)
- The system limit is imposed on the sum of the 1 MB and 2 GB pages
- For the form 1M=(*a*) and 1M=(*a*%) and 2G=(*a*) and 2G=(*a*%), the value specified for *a* or *a*% is applied toward the limit, with the percentage value calculated as *a*% * (online storage available at IPL – 4G)
- For the form 1M=(*a*,*b*) and 1M=(*a*%,*b*%) and 2G=(*a*,*b*) and 2G=(*a*%,*b*%), the value specified for *b* or *b*% is applied toward the limit, with the percentage value calculated as *b*% * (online storage available at IPL – 4G)
- For any percentage request that is by definitively over 80%, the system will reject the request as a specification error and prompt for re-specification of LFAREA, regardless of the **prompt** or **noprompt** specification. The following examples are specification error cases:
  - LFAREA=(1M=81%)
  - LFAREA=(1M=(40%,20%),2G=(41%,20%))
  - LFAREA=(1M=(1,0),2G=80%))
- If the 1M= or 2G= request is specified (or both), and the applicable value or sum of values exceeds the system limit, and prompt is also specified (either explicitly or by default), the system will issue a prompt to re-specify LFAREA. Otherwise, when noprompt is specified, the LFAREA parameter is ignored and the system will reserve zero 1 MB pages and zero 2 GB pages in the large frame area, and then continue with the IPL.
- For any request containing a target and minimum specification where the target or sum of targets is above the system limit, the system will attempt to satisfy the request at or less than the system limit, provided that the request can be satisfied at or above the minimum values specified.

Request handling for insufficiently contiguous online real storage

In some cases, the LFAREA request cannot be satisfied even though it does not exceed the system limit. For example, real storage could be configured in such a way that some offline areas are interspersed among online areas, preventing the system from locating enough contiguous storage areas that start on a 2 GB boundary to use for the set of 2 GB pages.

When the LFAREA request is specified using the *x*M, *x*G, *x*T, or *x*% syntax form to reserve 1 MB pages:

- For the syntax form *x*M, *x*G, and *x*T, the value specified for *x* is the target amount of real storage to reserve for 1 MB pages in the large frame area.
- For a percentage request, the amount is calculated as: (percent * online real storage at IPL) – 2 GB
- Once calculated, the percentage request is rounded down to the nearest 1 MB boundary.
- If rounding down the percentage request results in zero, the system will reserve zero pages in the large frame area and continue with the IPL.
- If the amount of storage required to satisfy the request cannot be located, the request will be satisfied at less than the requested amount and the system will then continue with the IPL.

When the LFAREA request is specified using the 1M= or 2 GB= syntax form to reserve 1 MB or 2 GB pages (or both):

- For the form 1M=(a) and 1M=(*a*%) and 2G=(*a*) and 2G=(*a*%), the value specified for *a* or *a*% is both the target and the minimum amount of real storage that the system is to reserve for 1 MB or 2 GB pages in the large frame area.
- For the form 1M=(*a*,*b*) and 1M=(*a*%,*b*%) and 2G=(*a*,*b*) and 2G=(*a*%,*b*%), the value specified for *a* or *a*% is the target amount, and *b* or *b*% is the minimum amount of real storage that the system is to reserve in the large frame area.
- For a percentage request, the amount is calculated as: percent * (online real storage at IPL – 4G)
- Once calculated, the 1M= or 2G= percentage request is rounded down to the nearest respective 1 MB or 2 GB boundary.
- If rounding down the percentage request results in zero, and prompt is specified (either explicitly or by default), the system will issue a prompt to re-specify LFAREA. Otherwise, when noprompt is specified, the LFAREA parameter is ignored and the system will reserve zero 1 MB pages and zero 2 GB pages in the large frame area, and then continue with the IPL.
- If the 1M= or 2G= request cannot be satisfied with at least the minimum amount requested, and prompt is specified (either explicitly or by default), the system will issue a prompt to re-specify LFAREA. Otherwise, when noprompt is specified, the LFAREA parameter is ignored and the system will reserve zero 1 MB pages and zero 2 GB pages in the large frame area, and then continue with the IPL.

Default Value: None (No LFAREA is defined)

Associated Parmlib Member: None

LICENSE

LICENSE={Z/OS}|{ZNALC}

This parameter specifies whether the system is running with a z/OS or ZNALC license.

Value Range: Not applicable

Default Value: LICENSE=Z/OS

Associated Parmlib Member: None

LNK

```
LNK={aa            }
    {(aa,bb,...[,L])}
    {(,L)          }
```

This parameter specifies one or more LNKLSTxx parmlib members that list program libraries to be concatenated to SYS1.LINKLIB, thus forming the LNKLST concatenation.

The two characters (A-Z, 0-9, @, #, or $) represented by aa (or bb, and so on), are appended to LNKLST to identify one or more LNKLSTxx members of parmlib.

If the L option is specified, the names of the data sets that are concatenated to SYS1.LINKLIB are displayed at the operator's console as the data sets are opened.

For information about the LNKLSTxx member, see Chapter 67, "LNKLSTxx (LNKLST concatenation)," on page 615. You can also use PROGxx to specify the concatenation.

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: LNK=00, causing selection of LNKLST00.

Associated Parmlib Member: LNKLSTxx

LNKAUTH

```
LNKAUTH={LNKLST}|{APFTAB}
```

This parameter specifies whether all libraries in the LNKLST concatenation are to be treated as APF-authorized when accessed as part of the concatenation, or whether only those libraries that are named in the APF table are to be treated as APF-authorized.

Value Range: Not applicable

Default Value: LNKLST, meaning that all libraries in the LNKLST concatenation are to be treated as APF-authorized when accessed as part of the concatenation. If the default for the LNKAUTH system parameter is taken, or is specified in IEASYSxx or by the operator, libraries in the LNKLST concatenation are APF-authorized when accessed as part of the LNKLST concatenation.

If a LNKLST library is not listed in the APF table, referencing the library through a JOBLIB or STEPLIB DD statement causes the library to be considered unauthorized for the duration of the job or step.

Associated Parmlib Member: None

LOGCLS

```
LOGCLS=x
```

This parameter specifies the JES output class for the log data sets. A log data set is queued to this class when its WTL limit has been reached. (The limit is specified by the LOGLMT initialization parameter.)

Example: LOGCLS=L

In this example, the current log data set is queued to output class L when the limit on the number of WTLs has been reached.

If the specified LOGCLS value is not valid, or an I/O error occurs while the IEASYSxx member is being read, master scheduler initialization prompts the operator for a replacement LOGCLS value. If prompting is forbidden (the OPI operand was specified), the default value A is assigned.

For the other log parameter, see LOGLMT.

Value Range: A single alphabetic or numeric character: A-Z or 0-9.

Default Value: A, which represents output class A.

Associated Parmlib Member: None

LOGLMT

LOGLMT=nnnnnn

This parameter specifies the maximum number of WTLs (messages) allowed for each log data set. The value is used by log processing to determine when a log data set should be scheduled for sysout processing by JES. When the value is reached, log processing issues a simulated WRITELOG command to close and free the current log data set, and to allocate and open a new log data set.

Example: LOGLMT=004852

In this example, when 4,852 WTLs have been issued to a log data set, the data set is scheduled for sysout processing on the output class specified by the LOGCLS parameter. Log processing then allocates and opens a new log data set.

If the specified value is not valid or an I/O error occurs while the IEASYSxx member is being read, master scheduler initialization prompts the operator for a replacement LOGLMT value. If prompting is forbidden (the OPI operand was specified), the default value of 500 is assigned.

For the other log parameter, see LOGCLS.

Value Range: 000000-999999

Default Value: 500

Associated Parmlib Member: None

LOGREC

LOGREC={dsname | LOGSTREAM | IGNORE}

LOGREC specifies the logrec recording medium to be used for error and environmental recording. If this parameter is omitted, then SYS1.LOGREC is the default data set name specification.

Before specifying LOGSTREAM to define a log stream as the logrec recording medium, **IBM suggests** that you load the initial program with a logrec data set initialized by IFCDIP00. If you do not load the initial program with a logrec data set, you cannot change the logrec recording medium from LOGSTREAM to DATASET using the SETLOGRC command.

dsname
Specifies the name of the logrec data set to be used for error recording. In a multisystem environment, if you specify a unique name for each of your logrec data sets, IBM suggests that you not place these names in the SYSTEMS exclusion resource name list in parmlib member GRSRNLxx.

Before an IPL, the logrec data set must have been allocated, cataloged (unless on the SYSRES) in the system master catalog, and initialized using

IFCDIP00. In a multisystem environment, take care in running IFCDIP00 to insure, if using the SYS1.LOGREC data set name default, that the correct logrec data set is initialized.

LOGSTREAM
Specifies that the logrec log stream (SYSPLEX.LOGREC.ALLRECS) is to be used for the error and environmental recording. The log stream provides a single repository for all of the MVS images in a sysplex. The log stream eliminates the need to allocate, catalog, and initialize a logrec data set on each system. When reporting programs, such as EREP, are run, the single log stream can be used as input to the program.

IGNORE
Specifies that error and environmental recording are to be ignored. Logrec records will not be recorded to the output medium; no records are written to a logrec data set or to the logrec log stream. Also, the system does not issue the ENF event code 36 signal for records when IGNORE is specified.

Attention: This specification is intended to be used only on test systems when a logrec data set is not established and the logrec log stream is not defined.

Syntax Examples for the LOGREC Parameter: Table 23 shows syntax examples of the LOGREC parameter and the results they produce.

Table 23. Syntax Examples for the LOGREC Parameter

Example	Result
`LOGREC=SYSA.LOGREC`	The data set 'SYSA.LOGREC' will be used.
`LOGREC=SYSTEMA.LOGREC`	The data set 'SYSTEMA.LOGREC' will be used.
`LOGREC=&SYSNAME;.LOGREC`	Assuming the value specified on the SYSNAME= parameter is SYSTEMA, the data set 'SYSTEMA.LOGREC' will be used.
`LOGREC=&SYSNAME; &SYSNAME;DATA.FILE`	Assuming the value specified on the SYSNAME= parameter is S1, the data set 'S1S1DATA.FILE' will be used.
`LOGREC=LOGSTREAM`	The logrec log stream, SYSPLEX.LOGREC.ALLRECS, will be used by SVC 76 (LOGREC) to record error and environmental records.
`LOGREC=IGNORE`	No error or environmental recording by SVC 76 will occur.

Value Range: One of the allowable specifications. The value for *dsname* can contain system symbols. When the dsname option is used, only one data set can be specified.

Default Value: SYS1.LOGREC

Associated Parmlib Member: When LOGSTREAM is specified, see Chapter 24, "COUPLExx (cross-system coupling facility (XCF) parameters)," on page 263.

LPA

```
LPA={aa            }
    {(aa,bb,...[,L])}
```

This parameter specifies one or more LPALSTxx parmlib members. The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so on), are

appended to LPALST to form the name of the LPALSTxx members. If the L option is specified, the system displays (at the operator's console) the names of the data sets successfully concatenated to SYS1.LPALIB.

The LPALSTxx members list data sets that are to be concatenated to SYS1.LPALIB. (For information about the use, contents, and syntax of LPALSTxx, see Chapter 69, "LPALSTxx (LPA library list)," on page 641.)

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: None

Associated Parmlib Member: LPALSTxx

MAXCAD

MAXCAD=nnn

Reserves the number of entries available for SCOPE=COMMON data spaces on all primary address space access lists (PASN-ALs) in the system. For a description of data spaces, see *z/OS MVS Programming: Extended Addressability Guide*.

A SCOPE=COMMON data space can be used by all programs in the system. It provides a commonly addressable area similar to the common storage area (CSA). SCOPE=COMMON data spaces are used by MVS and can also be used by subsystems and applications that need common storage.

Each SCOPE=COMMON data space uses one entry on all PASN-ALs in the system. Because the maximum number of entries in a PASN-AL is 510, each SCOPE=COMMON data space your program creates and adds to the PASN-AL, reduces the number of SCOPE=SINGLE and SCOPE=ALL data spaces that a program can address through its PASN-AL.

Note that when you select a value for MAXCAD, you must take into account the number of SCOPE=COMMON data spaces that subsystems, applications and MVS use.

If you code an incorrect number (less than 10 or greater than 250) the system uses the default number of 50 and issues an informational message.

Value Range: 10-250

Default Value: 50

Associated Parmlib Member: None

MAXUSER

MAXUSER=nnnnnn

This parameter 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 system tasks, the master scheduler, 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 is extended by the value specified for RSVNONR when non-reusable ASIDs exist.

The MAXUSER value must be large enough to include all the active address spaces running concurrently during a given IPL. Therefore, the value you specify for MAXUSER must take into account the number of initiators, TSO/E USERMAX, available VTAM APPLs and any other factors that contribute to the number of active address spaces.

However, too large a buffer can cause the system to allocate more system resources than are needed during a given IPL. The excessive allocation of system resources because of incorporating too large a buffer can lead to potential problems such as the following examples:

- The specification of MAXUSER in conjunction with CSCBLOC=BELOW and RSVSTRT can cause the system to allocate excessive storage below 16 M. See CSCBLOC for additional details.
- An aggressively large MAXUSER, RSVSTRT, and RSVNONR can cause the system to allocate an excessive amount of storage, which can potentially lead to a wait state 204 reason 008. (Wait state 204 with reason 008: An error occurred during allocation initialization; the system could not obtain storage for device information.) The system enters a non-restartable wait state 204 at IPL time.

When there are no available MAXUSER ASIDs, the system can use the value specified on the RSVSTRT system parameter to allow creation of additional started tasks.

When the system is heavily used, it can use the value specified on the RSVSTRT system parameter to allow more concurrent jobs and started tasks than the number specified by MAXUSER.

When there are non-reusable ASIDs, it can use the value specified on the RSVNONR systems parameter to allow the MAXUSER value to be honored until the number of non-reusable ASIDs exceeds the value of RSVNONR. When this happens, the number of jobs and started tasks that can run concurrently will be reduced by the difference between the number of non-reusable ASIDs and the value of RSVNONR.

Assume, for example, that MAXUSER specifies 500 and RSVSTRT specifies 5. If there is an attempt to start an address space (using the START command), and none of the 500 address space entries defined by the MAXUSER parameter is available (meaning heavy system use), but an entry defined by the RSVSTRT parameter is available, the system uses that entry. Thus, when the system is heavily used, there can be more concurrent jobs and started tasks in the system than the number defined by MAXUSER. The absolute limit to the number of concurrent jobs and started tasks is the sum of the values specified for the MAXUSER and RSVSTRT system parameters. The maximum ASID value is the sum of the values specified for the MAXUSER, RSVSTRT, and RSVNONR system parameters.

If started tasks or batch jobs that create non-reusable ASIDs end enough times, they will exhaust all available ASIDs and the system will require an IPL. When IPL is not an acceptable option, determine which programs caused the problems and fix them. For methods that prevent running out of ASIDs, see *z/OS MVS Programming: Extended Addressability Guide*.

Value Range: 0-32767. Note that the sum of the values specified for the MAXUSER, RSVSTRT, and RSVNONR system parameters cannot exceed 32767.

Default Value: 255

Associated Parmlib Member: None

MLPA

```
MLPA={aa                    }
     {(aa[,L][,NOPROT])     }
     {(aa,bb...[,L][,NOPROT])}
```

This parameter specifies one or more IEALPAxx parmlib members, which list modules to be added to the pageable LPA, as a temporary LPA extension.

The two characters (A-Z, 0-9, @, #, or $) represented by aa (or bb, and so on), are appended to IEALPA to form the name of the IEALPAxx members. If the L option is specified, the system displays the contents of the IEALPAxx parmlib members at the operator's console as the system processes the members.

The LPA modules in the IEALPAxx parmlib members are page protected in storage, by default. If an attempt is made to store into a page-protected module, a protection exception occurs. However, the NOPROT option allows an installation to override the page protection default. When NOPROT is specified, the LPA modules in the IEALPAxx parmlib members are not page protected.

The installation can use the MLPA parameter to temporarily modify an existing LPA at a quick start or a warm start IPL (without creating a new LPA through the CLPA parameter). The added modules are temporary in that they remain as an LPA extension only for the duration of the current IPL. The temporary modules will not be *automatically* reinstated by a quick start or a warm start IPL. That is, the MLPA parameter must be specified again in the next IPL to reinstate the modified LPA.

If the installation wants to retain the temporary modules as a permanent part of the LPA, it should use the IEBCOPY utility or the linkage editor to place the modules in a data set that is part of the LPALST concatenation, and specify the CLPA and LPA parameters at a future IPL to load the specified LPALST concatenation into the LPA.

For additional information about the MLPA option, see IEALPAxx. For information about the fixed LPA option, see the FIX parameter and Chapter 47, “IEAFIXxx (fixed LPA list),” on page 389.

Value Range: Any two characters (A-Z, 0-9, @, #, or $), repeated if desired.

Default Value: None If MLPA is not specified, no modified LPA is created.

Associated Parmlib Member: IEALPAxx

MSTRJCL

```
MSTRJCL=(xx[,L])
```

This parameter specifies the name of the data set that contains the JCL used to start the master scheduler address space and route JCL messages that are issued for master scheduler processing to consoles that receive operator information or system/error maintenance messages. Two alphanumeric characters, represented by xx, are appended to MSTJCL to form the name MSTJCLxx. The system first looks for a MSTJCLxx parmlib member that contains the master JCL. If one exists, the system uses that member. If the MSTJCLxx parmlib member does not exist, the system looks for the MSTJCLxx module in SYS1.LINKLIB.

When MSTRJCL=(xx[,L]) is specified, JCL messages issued for master scheduler processing are routed to consoles that receive operator information (routing code 2) or system/error maintenance (routing code 10) messages. Use MSTRJCL=(xx[,L]) only for debugging purposes.

Using the MSTRJCL system parameter allows an installation to test new master scheduler JCL. For example, you can store test JCL in MSTJCL01 and select it by using the MSTRJCL parameter (MSTRJCL=01). For more information, see "Understanding the master scheduler job control language" on page 10.

MSTRJCL can be specified in the default system parameter list (IEASYS00), IEASYSxx members, or entered by the operator in response to the SPECIFY SYSTEM PARAMETERS message. The default value of 00 selects MSTJCL00 when no value is specified for MSTRJCL or if the operator presses the ENTER key in response to a prompt. If MSTJCL00 is not found, the operator is prompted until a MSTJCLxx member can be found. The IPL cannot continue without the JCL needed to start the master scheduler.

Value Range: Any two alphanumeric characters.

Default Value: MSTRJCL=00, causing selection of MSTJCL00.

Associated Parmlib Member: MSTJCLxx

NONVIO

```
NONVIO={dsname                        }
       {(dsname1,dsname2,...,dsnameN)}
```

This parameter allows an installation to direct VIO paging away from the specified local page data sets. Specify one or more local page data sets that are not to be used for VIO paging when space is available on other local page data sets. The page data sets that are designated as non-VIO will contain only address space pages or free slots. However, if space is depleted on the page data sets that allow VIO paging, the non-VIO page data sets will be used for VIO paging. (Using non-VIO page data sets for VIO pages is called an *overflow condition*. The operator is notified if an overflow condition occurs.)

Each dsname specified must be a valid name consisting of a maximum of 44 characters whose format is the same as that required for the PAGE system parameter. Each dsname must specify a data set that was specified as a local page data set on the PAGE parameter, either in the IEASYSxx parmlib member or by the operator in response to the "SPECIFY SYSTEM PARAMETERS" prompt.

If you specify a name that the system cannot identify as the name of a local page data set, then the system ignores that name and issues a message to inform the operator that the data set cannot be recognized as a non-VIO page data set. If you omit the NONVIO parameter, then VIO pages are allowed on all local page data sets.

The name of the NONVIO data sets you specify can include the &SYSNAME system symbol. Using the same system symbol syntax with the NONVIO parameter that was used with the PAGE parameter can simplify your system maintenance.

For cold or quick starts a data set's NONVIO designation is not preserved; you must respecify the NONVIO system parameter. If you do not designate a data set as non-VIO on a cold or quick start, you can designate it in the NONVIO system parameter when you do a subsequent warm start; that data set is marked as non-VIO then, and no more VIO pages will be sent to it (unless an overflow condition occurs). If the non-VIO paging data set remains on the system long enough, all VIO pages on it will eventually migrate, during the normal course of system operation, to other page data sets used for VIO.

For warm starts, a data set's NONVIO designation is preserved. A warm start preserves journaled VIO data set pages. Therefore, all local page data sets that

contain VIO pages are required for a warm start. These required data sets would include non-VIO page data sets to which VIO paging was done because of an overflow condition.

Note: During a warm start, a quick start will be forced if (1) a local page data set, not specified as NONVIO, is unavailable or unusable, or (2) a non-VIO local page data set that contains VIO pages was removed before all of its VIO pages had migrated to other page data sets used for VIO.

If you specify all local page data sets on the NONVIO data set name list, a message is issued to inform the operator of this condition. VIO pages can be written to all local page data sets. Similarly, if the directed VIO function is turned off via the DVIO parameter in the IEAOPTxx parmlib member, all local page data sets can receive VIO pages. If the directed VIO function is turned on again, then auxiliary storage management directs VIO away from any local page data sets designated as NONVIO. The VIO pages on these data sets will eventually migrate to VIO page data sets.

Value Range: Any number of data set names may be specified up to the same limit as exists for the PAGE system parameter. The data set names can contain system symbols.

Default Value: None

Associated Parmlib Member: While there is no directly-associated parmlib member you must specify DVIO=YES in the IEAOPTxx parmlib member to activate the directed VIO function; DVIO=NO allows VIO pages to go to any local page data set.

Note: DVIO=YES is the default in the IEAOPTxx parmlib member allowing the use of the directed VIO.

NSYSLX

NSYSLX={nn | (nn,mm)}

This parameter allows you to specify the number of linkage indexes (LXs) (in addition to those in the system function table) to be reserved for system LXs. See *z/OS MVS Programming: Extended Addressability Guide* for information about system LXs.

If you omit the NSYSLX parameter, the system reserves 165 system LXs. You might need to specify NSYSLX if either of the following conditions is true:

1. Your installation runs applications that request (through the LXRES macro) more than 165 system LXs.
2. 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 reloading the initial program to reclaim the system LXs.

The total number of LXs consists of the sum of the non-system LXs, plus the number of LXs in the system function table, plus the value specified in the NSYSLX parameter. The total number of LXs you can have depends on your z/OS release level and hardware level:

- The total number of LXs can be up to 2048 on a system that is running on a processor prior to either a z990 or z890 processor at driver level 55.
- The total number of LXs can exceed 2048, up to a value of 32768 (extended LX support) if the system is running on either a z890 or z990 processor at driver level 55 or above.

nn Specifies the number of LXs up to the value of 2048 to be reserved for system LXs. The value range for *nn* is between 10-512

mm The second value, *mm*, specifies the number of LXs that exceed 2048, up to the value of 32768 (extended LX support) to be reserved for system LXs. To get more than 2048 LXs, your system must support extended LX function. If you specify *mm* on a system that does not support extended LXs, the system will not provide additional LXs above the 2048 limit. The value range for *mm* is between 40-8192.

Note: If the value of *nn* is specified but the value of *mm* is not, *mm* will be of the same value as *nn*.

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 means you can put off reloading the initial program to reclaim the system LXs, until when it is less disruptive.

Value Range: 10-512 for *nn*, 40-8192 for *mm*

Default Value: 165 for both *nn* and *mm*

Associated Parmlib Member: None.

OMVS

```
OMVS={nn     }
     {(nn)   }
     {(nn,mm)}
     {DEFAULT}
```

This parameter specifies the parmlib member or members to use to locate the parmlib statements to configure the kernel.

nn specifies the BPXPRMnn parmlib member and **(nn,mm)** specifies the set of parmlib members BPXPRMnn and BPXPRMmm. If you specify more than one parmlib member, any redundant parmlib statement data found in a later member in the list is ignored.

Specifying OMVS=DEFAULT indicates that the kernel is to be started in its minimum configuration mode with all parmlib statements taking their default values.

Example:

```
OMVS=(XX,YY,ZZ)
```

This parameter provides the ability to easily reconfigure a large set of z/OS UNIX system characteristics and to provide the ability to keep the reconfiguration settings in a permanent location for subsequent reuse or reference. For more information about setting your parmlib values, see *z/OS UNIX System Services Planning*.

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: DEFAULT

Associated Parmlib Member: BPXPRMxx

OPI

```
OPI={YES}|{NO }
```

This parameter specifies whether the operator is to be allowed to override system parameters contained in IEASYSxx members of parmlib. The YES operand allows operator overrides. The NO operand causes overrides to be ignored. If, however, NIP detects an incorrect parameter in an IEASYSxx member in which OPI=NO applies, NIP ignores the OPI specification and prompts the operator.

OPI may be specified only in an IEASYSxx member; it may not be specified by the operator.

OPI may be specified either for individual system parameters or for the entire set of parameters.

The following examples show how this can be specified:

```
IEASYSAA:  MLPA=(00,01),SQA=(10,OPI=NO)
IEASYSBB:  MLPA=(00,01),SQA=10,OPI=NO
```

For IEASYSAA, the operator can override MLPA values but not the SQA value. For IEASYSBB, however, the operator can override neither MLPA nor SQA values.

When you specify OPI for individual system parameters, you can use system symbols in the specified value. When you specify OPI for the entire set of parameters, do not use system symbols in the specified value.

Note: During system initialization, the system first uses the IEASYS00 parmlib member to establish parameters, then uses parameters from the operator or any other IEASYSxx parmlib member (identified by the SYSPARM parameter in the LOADxx parmlib member, the SYSPARM parameter in the IEASYMxx parmlib member, or the SYSP=xx parameter) to replace established parameters or add new ones. The OPI parameter in IEASYS00 carries over to any other IEASYSxx parmlib member identified during the IPL. If you specify OPI=NO in IEASYS00 for an IPL, the parameters affected by the OPI=NO cannot be changed by operator command during the IPL. Even if you use another IEASYSxx parmlib member and specify OPI=YES for that SYSP=xx, the operator cannot change any parameters affected by OPI=NO in IEASYS00.

: In the following IEASYSxx parmlib members, the specification of CSA and SQA in IEASYS00 will prevail for the life of the IPL even if the operator uses SYSP=01 to select IEASYS01 for the initialization process:

```
IEASYS00:  CSA=(100,OPI=NO),MLPA=(00,01),SQA=(10,OPI=NO)
IEASYS01:  CSA=(100,OPI=YES),MLPA=(00,01),SQA=10
```

Value Range: Not applicable

Default Value: YES

Associated Parmlib Member: Not applicable

OPT

```
OPT={xx      }
    {(xx[,L])}
    {(,L)    }
```

This parameter specifies a parmlib member that contains the parameters that affect some decisions made by the system resources manager (SRM). The two alphanumeric characters, represented by xx, are appended to IEAOPT to form the name of the IEAOPTxx member. If the L option is specified, the system displays the contents of the IEAOPTxx parmlib member at the operator's console as the system processes the member.

If the member cannot be found or contains incorrect specifications, SRM prompts the operator to specify an alternate member by respecifying OPT=xx. If the operator cancels the parameter by replying with the ENTER key, SRM uses default values.

The operator can select a new OPT (that is, indicate that the system is to run under the control of an alternate IEAOPTxx member) between IPLs by issuing the SET OPT command.

Value Range: Any two alphanumeric characters

Default Value: None If OPT is not specified, SRM uses default values. (See *z/OS MVS Initialization and Tuning Guide* for the default values.)

Associated Parmlib Member: IEAOPTxx

PAGE

```
PAGE={dsname|*NONE*                    }
     {(dsname1|*NONE*)[,dsname2|*NONE*,[dsname3,...[,L])}
     {(,L)                        }
```

This parameter allows the installation to name page data sets as additions to existing page data sets. The maximum number of page data sets is 256. The system determines which page data sets to use by merging information from three sources: IEASYS00, IEASYSxx, and the PAGE parameter.

During system initialization, the system first uses the list of page data sets specified on the PAGE parameter of the IEASYS00 parmlib member. It then uses any other IEASYSxx parmlib member (identified via the SYSP=xx parameter). The IEASYSxx PAGE data set name list overrides the one in IEASYS00.

PAGE=dsname and PAGE=(dsname1,dsname2,...[,L]) allow the operator to add page data sets to the list of data sets already specified in IEASYSxx. If the PAGE data set name list in IEASYSxx is null, the operator specification is used.

The system generates a list of all the page data sets that the initialization routines have opened. If the “L” keyword is specified (either in parmlib or from the operator's console) this list is then written to the operator's console and to syslog. If the “L” keyword is not specified, the list is written only to the syslog.

The system interprets the final merged sequence of page data set names specified as follows:

- The first named data set on the list is used as the PLPA page data set. This data set contains pageable link pack area (PLPA) pages. If ***NONE*** is specified, no PLPA page data set is used and no quick or warm starts can be performed. ***NONE*** can only be used when Storage Class Memory (SCM) is online and used for paging (refer to “PAGESCM” on page 474).
- The second named data set in the list is used as the common page data set. This data set contains all of the common area pages that are not PLPA pages. If ***NONE*** is specified, no common area page data set is used. ***NONE*** can only be used when Storage Class Memory (SCM) is online and used for paging (refer to “PAGESCM” on page 474).
- The third and all subsequently named data sets are used as local page data sets. These data sets contain all the system pages (including VIO pages) that are considered neither PLPA nor common data set pages.

Note: To replace local page data sets during an IPL, you must specify the CVIO parameter. (Note that CLPA implies CVIO.)

If Storage Class Memory (SCM) is online and used for paging (refer to "PAGESCM" on page 474), SCM will be used for all page types, in addition to the page data sets specified by the **PAGE=** parameter.

When defining page data sets, you must ensure that the desired PLPA page data set is the first entry in the data set list, in both IEASYS00 and IEASYSxx.

During initialization, there are no checks on the sizes of user-supplied data sets. When initialization completes, the PAGEADD command can be used to add more local page data sets to the system. The most current page data set information is preserved so that it can be used for subsequent quick start and warm start IPLs.

The data set intended for PLPA should contain enough space for the entire PLPA, including the extended PLPA. If the entire PLPA cannot fit on this data set, the system puts the excess on the common page data set. And, if the common page data set gets full, its excess goes to the PLPA page data set. In the interest of good performance, however, you should make the common page data set big enough to prevent its "spilling over" to the PLPA page data set (except in cases forced by error situations). Finally, if both SCM and the PLPA and common data sets are full (or have ***NONE*** specified) then the system puts the excess on the local page data sets. Spilling PLPA into local page data sets results in the failure of subsequent warm and quick starts. For specific data set size and placement recommendations, see *z/OS MVS Initialization and Tuning Guide*.

How Page Data Sets Are Specified: Page data sets are specified by a merging of information from three sources: 1) IEASYSxx; 2) operator-issued PAGE parameter; and 3) the page data sets from a previous cold start. The system merges this information as follows:

- *From the PAGE parameter in IEASYSxx* (an alternate system list):
- *From the PAGE parameter specified by the operator in the current IPL*: The system merges this page specification with that in either IEASYS00 or IEASYSxx, but not both. The operator specification of page data sets lasts until the next cold or quick start.
- *The page data sets previously in use*: The system uses information about quick starts and warm starts (IPLs that do not specify the CLPA parameter). If a non-demountable device (such as a 3350) is used for the IPL, the device must be online before the IPL.

Note: Two other conditions are prerequisite for certain warm start or quick start situations:

1. The local page data sets that contain VIO pages from the previous IPL must be mounted for *all* warm starts, to make VIO slots available. Otherwise, ASM forces a quick start instead.
2. The common page data set from the previous IPL must be mounted for *both* quick starts and warm starts if the system's writing of the PLPA (and extended PLPA) to the PLPA page data set during the previous cold start resulted in spilling some of the PLPA pages into the common page data set.

Usually, page data sets specified by any means must have been allocated, cataloged in the system's master catalog, and preformatted in VSAM format before an IPL can start. You can format the data sets by using the DEFINE PAGESPACE

command of access method services (for information about the formatting process, see *z/OS DFSMS Access Method Services Commands*).

Attention: Do not change or swap the names of systems in the sysplex when using page data set names containing system symbols. If name changes are required, each system in the sysplex must have a cold start IPL.

Syntax Examples for the PAGE Parameter: Table 24 shows syntax examples of the PAGE parameter and the results they produce.

Table 24. Syntax examples for the PAGE parameter

Syntax	Result
`PAGE=dsname` `PAGE=(dsname)`	Statements each specify one page data set.
`PAGE=(dsname1,dsname2,dsname3)`	Specifies three page data sets; dsname1 holds the PLPA pages, dsname2 holds the common pages, and dsname3 holds the private area pages.
`PAGE=(dsname1,dsname2,...,` `dsname`*n*`)`	Specifies *n* page data sets; dsname1 holds the PLPA pages, dsname2 holds the common pages, and dsname3 through dsname*n* all hold private area pages.

Example 4: The following statement specifies only one page data set:
PAGE=(*NONE*,*NONE*,dsname1)

Storage-class memory (SCM) holds all page types.

Note:

1. If the operator specifies the PAGE parameter, ASM initialization adds (but does not replace) the data sets as specified. The PAGE data set name list in IEASYS00 or IEASYSxx contains the first named data sets. To ensure that the operator-specified data sets are used for the PLPA and common page data sets, it is necessary to use an IEASYSxx member that contains a null PAGE parameter; one that does not specify page data sets.
2. It is unnecessary to specify either UNIT or VOLSER because all page data sets must be cataloged in the system's master catalog. ASM initialization therefore does not need externally specified volume serial numbers. The operator may either pre-mount volumes or await a mount message.

Minimum Paging Space: ASM enforces minimum requirements for paging space. If the requirements are not satisfied, ASM is forced to end the IPL. Additionally, the use of minimum paging space is inadvisable because it can result in poor performance.

Minimum requirements are as follows:

- There must be at least a PLPA, a common, and a local page data set to IPL the system. However, ***NONE*** can be used in place of PLPA and common area page data sets when SCM is used for paging (refer to "PAGESCM" on page 474 for details).
- The PLPA and common page data sets must be able to hold the total combination of PLPA and common pages (excluding the SQA). If no errors

occur, the auxiliary storage space for the PLPA and common page data sets is sufficient if the space equals the size of the PLPA and CSA divided between the two data sets.

If the PLPA and common page data sets spill back and forth because the space is not properly divided between the data sets, performance degradation can result. Severe performance degradation can result if the common page data set is not large enough and, therefore, spills to the PLPA page data set, which is normally read-only after IPL.

- Local page data sets are used to hold all private area and VIO pages. The amount of storage necessary varies with with each system and can be calculated using the guidelines in *z/OS MVS Initialization and Tuning Guide*.

Page Space Shortage: Two warning messages appear when the system resources manager (SRM) detects a shortage of page space, the first when 70% of the available local paging space including SCM, if any has been allocated, and the second when 85% has been allocated. SRM reacts to the situation by preventing the creation of new address spaces. That is, new "start initiator" commands ($SInn), LOGONs, MOUNT commands, and START commands for system tasks that run in their own address spaces do not work. Upon receipt of these messages, you can add paging space to the system dynamically by using the CONFIG SCM,ONLINE or the PAGEADD operator command. For information about using PAGEADD or CONFIG, and for related information about using the PAGTOTL parameter, refer to *z/OS MVS System Commands*. For these situations, it is valuable to have preformatted, cataloged VSAM paging data sets or offline SCM available. The data sets can be formatted by using the DEFINE PAGESPACE processor of access method services; for more information, refer to *z/OS DFSMS Access Method Services Commands*.

When the page space usage has been decreased below 70% utilization, SRM informs the operator that there is no longer a shortage.

Value Range: The total number of data sets specified must not exceed the combined limit for page data sets on the PAGTOTL= parameter. The data sets can contain system symbols.

Default Value: None

Associated Parmlib Member: None

Note: During NIP processing, the system might exhaust SQA and extended SQA if many local paging data sets were specified on the PAGE parameter. If this condition occurs, the value specified for the SQA parameter might be set too low. You can increase the SQA value in IEASYSxx .

PAGESCM

```
PAGESCM={xxxxxxM                }
        {xxxxxxG                }
        {xxT                    }
        {ALL                    }
        {NONE                   }
        {0                      }
```

This parameter specifies the minimum amount of Storage Class Memory (SCM) that should be made available for use as auxiliary storage. The system reserves this

amount of SCM during IPL for subsequent use as auxiliary storage. Additional SCM will be allocated on an as-needed basis if use of this initial amount of SCM is exceeded.

Value Range: The following value ranges can be specified for the PAGESCM parameter to reserve SCM for paging at IPL:

0 | 0M | 0G | 0T
Indicates that no SCM will be reserved for paging at IPL. Instead, SCM will be allocated as needed, based on paging demand.

*xxxxxx***M**
Specifies the amount of SCM to reserve for paging at IPL, in megabytes. This value can be from 1–6 decimal digits.

*xxxxxx***G**
Specifies the amount of SCM to reserve for paging at IPL, in gigabytes. This value can be from 1–6 decimal digits.

xxT
Specifies the amount of SCM to reserve for paging at IPL, in terabytes. This value can be from 1–2 decimal digits. The maximum amount of SCM supported for paging is 16 terabytes (16T).

NONE
SCM is not used for paging. This parameter remains in effect until the next IPL.

ALL
Reserves all SCM for paging at IPL.

Default Value: **ALL**

Associated Parmlib Member: None

PAGTOTL

PAGTOTL

This parameter allows you to specify the total number of page data sets available to the system. The value you specify for ppp is valid for the life of the IPL. You must include the following in the ppp value:

- a PLPA page data set
- a common page data set
- a permanently reserved slot
- a local page data set. You must have at least 1 local page data set to IPL or a WAIT03C will occur.

Therefore, the minimum ppp value required to IPL is 4. In addition, ppp should include any page data sets that can be dynamically added after IPL.

You can add page data sets by using the PAGEADD operator command until the total number reaches the value specified on the PAGTOTL parameter. However, if you try to exceed the limits set on the PAGTOTL parameter, the system will truncate the excess specification.

The system supports a maximum of 256 page data sets (including 1 required page data set, a PLPA page data set, a common page data set, a reserved slot, and up to 252 additional local page data sets).

The system reserves ESQA space for each page data set that can be defined. Additional ESQA is required for each page data set that is actually in use by the system. The amount of additional storage will depend on the size of every in-use page data set.

How Page Number Values Are Obtained: The PAGTOTL parameter is specified in one of two sources: 1) IEASYS00 or IEASYSxx, or 2) by the operator-issued PAGTOTL parameter. An outline of these specifications is as follows:

- *The PAGTOTL parameter in IEASYS00.*
- *The PAGTOTL parameter in IEASYSxx,* an alternate system parameter list: If the operator selects this list (by using the SYSP parameter), the PAGTOTL parameter in IEASYSxx overrides the PAGTOTL parameter in IEASYS00.
- *The PAGTOTL parameter specified by the operator* in the current IPL: This PAGTOTL specification overrides the specification in IEASYS00 or IEASYSxx. The operator specification lasts only for the life of the IPL.
- If the number of page data sets specified on the PAGE parameter exceeds the PAGTOTL value, the PAGTOTL value will be dynamically increased at IPL. A message is issued if this occurs.

Syntax Example for the PAGTOTL parameter: This specification causes the system to allow for the total of page data sets (*ppp*).

```
PAGTOTL=(ppp)
```

Value Range: Valid *ppp* values are 0-256. This value is the maximum allowable number of page data sets that may be in use in the paging configuration at a time. Three spaces are reserved for IBM use (for the PLPA and common data sets, and one additional space). Therefore, the maximum number of local page data sets is 253.

Default Value: 40

Associated Parmlib Member: None

Note: Swap data sets are no longer supported. If your PAGTOTL parameter is coded to include the number of swap data sets (in other words, if you are using the syntax PAGTOTL=(ppp,sss)), the system will generate informational message IEA051I during IPL to indicate that the swap parameter will be ignored.

PAK

```
PAK={aa            }
    {(aa,bb,...[,L])}
    {(,L)          }
```

This parameter specifies one or more IEAPAKxx parmlib members that contain groups of names of modules in the LPALST concatenation that are processed together or in sequence. The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so on), are appended to IEAPAK to form the name of the IEAPAKxx members. If the L option is specified, the system displays the contents of the parmlib members at the operator's console when the system processes the members.

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: PAK=00, causing selection of IEAPAK00, if it exists.

Associated Parmlib Member: IEAPAKxx.

PLEXCFG

```
PLEXCFG={XCFLOCAL   }
        {MONOPLEX   }
        {MULTISYSTEM}
        {ANY        }
```

Specifies the type of configuration into which the system is allowed to IPL. You can specify one or more of the following system configurations:

PLEXCFG=MULTISYSTEM
Specifies that the system is to be part of a sysplex consisting of one or more MVS systems that reside on one or more processors. The same sysplex couple data sets must be used by all systems.

You must specify a COUPLExx parmlib member that identifies the same sysplex couple data sets for all systems in the sysplex (on the COUPLE statement) and signalling paths between systems (on the PATHIN and PATHOUT statements). You must also specify in the CLOCKxx parmlib member whether you are using a Sysplex Timer that is real (ETRMODE=YES) or simulated (SIMETRID=YES).

Use MULTISYSTEM when you plan to IPL two or more MVS systems into a multi-system sysplex and exploit full XCF coupling services. GRS=NONE is not valid with PLEXCFG=MULTISYSTEM.

PLEXCFG=XCFLOCAL
Specifies that the system is to be a single, standalone MVS system that is not a member of a sysplex and cannot use couple data sets. The COUPLExx parmlib member cannot specify a sysplex couple data set, and, therefore, other couple data sets cannot be used. Thus, functions such as WLM, that require a couple data set, are not available.

In XCF-local mode, XCF does not provide signalling services between MVS systems. However, multi-system applications can create groups and members, and messages can flow between group members on this system. If signalling paths are specified, they are tested for their operational ability, but they are not used.

Use XCF-local mode for a system that is independent of other systems. In XCF-local mode, XCF services (except permanent status recording) are available on the system and you can do maintenance, such as formatting a couple data set or changing the COUPLExx parmlib member. The IBM-supplied default parmlib member COUPLE00 and the COUPLE=** system parameter are intended to be used to bring up the system in XCF-local mode.

PLEXCFG=MONOPLEX
Specifies that the system is to be a single-system sysplex that must use a sysplex couple data set. Additional couple data sets, such as those that contain policy information, can also be used. XCF coupling services are available on the system, and multi-system applications can create groups and members. Messages can flow between members on this system (but not between this system and other MVS systems) through XCF signalling services. If signalling paths are specified, they are not used.

You must specify a COUPLExx parmlib member that gives the system access to a sysplex couple data set to be used only by this system. When a system IPLs into a single-system sysplex, no other system is allowed to join the sysplex.

Use MONOPLEX when you want only one system in the sysplex (for example, to test multi-system applications on one system) or when you want to use a function, such as WLM, that requires a couple data set.

PLEXCFG=ANY
Specifies that the system can be part of any valid system configuration. Specifying ANY is logically equivalent to specifying XCFLOCAL, MONOPLEX, or MULTISYSTEM. ANY is the default.

Generally avoid specifying PLEXCFG=ANY and explicitly specify the sysplex environment that you intend for the system to join. See *z/OS MVS Setting Up a Sysplex* for additional information about the PLEXCFG parameter and planning for XCF-local mode.

To prevent the operator from overriding the PLEXCFG parameter, specify OPI=NO on the PLEXCFG keyword in the IEASYSxx parmlib member. (For example, PLEXCFG=MULTISYSTEM,OPI=NO).

Default Value: PLEXCFG=ANY

Associated Parmlib Member: None

PRESCPU

PRESCPU

This parameter causes system initialization CPU processing to bring logically online those CPUs (and only those CPUs) that are physically online when the IPL is initiated, without regard to the number of CPUs defined to be initially online in the logical partition profile.

The purpose is to ensure that all CPUs that are online when an IPL is initiated will be online when the IPL has completed.

Note that CPUs that are offline when an IPL is initiated will remain offline. The installation might wish to not employ PRESCPU when WLM CPU management is active, because that processing might have configured CPUs offline.

If PRESCPU is not coded, CPU initialization will take CPUs offline or bring CPUs online as needed to make the number of online CPUs equal to the number of initial CPUs defined for the partition, when that information is provided by the machine (as it should be on model 2064 (GA-3), model 2066, and later machines) and is not 0.

If the number of initial CPUs is not provided or is 0, CPU initialization will attempt to bring online all CPUs that are physically online. On most models, those will be the CPUs that were online just before the IPL was initiated, but models 2064 and 2066 may configure additional CPUs online when an IPL is initiated, and CPU initialization will bring those CPUs online as well.

Value range: Not applicable

Default: Not applicable

Associated parmlib member: None

PROD

```
PROD={aa        }
     {(aa,bb...)}
```

The two characters (A-Z, 0-9, @, #, or $), represented by xx, are appended to IFAPRD to specify the name of one or more IFAPRDxx members of parmlib. You can specify one or more members, which define the product enablement policy for the system.

The specified IFAPRDxx parmlib members must exist. The system processes the members in the order in which you specify them, stopping if it encounters any parmlib member that does not exist.

Value Range: Any two alphanumeric characters

Default Value: None

Associated Parmlib Member: IFAPRDxx

PROG

```
PROG={aa         }
     {(aa,bb,...)}
```

This parameter specifies the PROGxx members of parmlib. The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so on), are appended to PROG to form the names of the PROGxx members. The PROGxx parmlib member contains four statement types: APF, EXIT, SYSLIB, and LNKLST.

- APF statements list the names and volume serial identifiers for APF-authorized libraries, and specify the format of the APF list, which is dynamic or static.
- EXIT statements control the use of exits and exit routines.
- SYSLIB statements define alternate data sets for SYS1.LINKLIB, SYS1.MIGLIB, and SYS1.CSSLIB at the beginning of the LNKLST concatenation and an alternate data set for SYS1.LPALIB at the beginning of the LPALST concatenation.
- LNKLST statements control the definition and activation of the LNKLST set that forms the LNKLST concatenation.

Use the PROG=xx rather than the APF=xx system parameter, which indicates the current IEAAPFxx parmlib member. IEAAPFxx can also be used to define the APF list, but only in a static format. If you specify both the PROG=xx and the APF=xx parameters, then the system places into the APF list the libraries listed in IEAAPFxx, followed by the libraries listed in the PROGxx member or members. IBM suggests that you convert IEAAPFxx to PROGxx (using a procedure described in Chapter 41, "IEAAPFxx (authorized program facility list)," on page 371), remove the APF=xx system parameter from IEASYSxx, and remove APF=xx from IEASYS00.

Use PROG=xx rather than the EXIT=xx system parameter, which indicates the current EXITxx parmlib member. EXITxx can also be used to specify exits, but you can specify only one exit routine at a time for each exit. IBM suggests that you convert EXITxx to PROGxx (using the IEFEXPR REXX exec provided by IBM), remove the EXIT=xx system parameter from IEASYSxx, and remove EXIT= from IEASYS00.

Use PROG=xx rather than LNK=xx to activate the LNKLST concatenation at IPL. In the PROGxx member, you can also define alternate data sets for the system defaults to appear at the beginning of the LNKLST and LPALST concatenations. If you specify both PROG=xx for a member with a LNKLST ACTIVATE statement and LNK=xx the system uses the definitions in PROGxx and issues message CSV478I:

```
LNK IPL PARAMETER HAS BEEN IGNORED.
LNKLST SET lnklstname IS BEING USED
```

Value Range: Any two characters (A-Z, 0-9, @, #, or $).

Default Value: If PROG=xx and APF=xx are not specified, the system automatically places SYS1.LINKLIB and SYS1.SVCLIB in the APF list. If the default for the LNKAUTH system parameter (LNKAUTH=LNKLST) is accepted or specified, libraries in the LNKLST concatenation are also authorized (when they are accessed as part of the LNKLST concatenation). If a library is in the LNKLST concatenation but is not APF-authorized, the system will consider the library to be unauthorized for the duration of the job or step if the library is referred to through a JOBLIB or STEPLIB DD statement.

Note: In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

If PROG=xx is not specified, the system uses LNK=xx for the LNKLST concatenation.

Associated Parmlib Member: PROGxx

RDE

RDE={YES}|{NO }

This parameter specifies whether the reliability data extractor (RDE) feature is to be included (YES) or not (NO). For information about RDE, see *z/OS MVS Diagnosis: Tools and Service Aids*.

Value Range: Not applicable.

Default Value: NO. This default applies if you omit the RDE parameter, or specify an incorrect value for it.

Associated Parmlib Member: None

REAL

REAL=nnnn

This parameter specifies the maximum amount of central storage, in 1 KB blocks, that can be allocated concurrently for ADDRSPC=REAL jobs (that is, V=R jobs). The value is rounded to a multiple of 4 KB.

Syntax Example: REAL=150 150/4=37.5 pages. Rounding to the next page boundary yields 38 pages, or a value of 152K. This statement allows up to 152K (152 x 1024) bytes to be allocated for use by V=R jobs.

Note:

1. If possible, avoid a large value for the REAL parameter because a large value degrades system performance even when no REAL regions are allocated.
2. The system allocates virtual equals real (V=R) regions upon request by those programs that cannot tolerate dynamic relocation. Jobsteps can request to run in a V=R region by using the ADDRSPC=REAL keyword on the EXEC or JOB statement. If you are sure you have no requirement for running V=R jobs, then specifying REAL=0 will disallow the use of ADDRSPC=REAL on any JCL.

3. In order to eliminate the impact of transition swaps, set REAL=0 and RSU=0 if you have no requirement for reconfiguring storage and no need to run a V=R job.

Value Range: 0-9999. The operand can be from one to four digits.

Default Value: 76. (This means a default value of 76K.)

Associated Parmlib Member: None

RER

```
RER={YES}|{NO}
```

This parameter specifies whether the reduced error recovery (RER) procedures for magnetic tapes are to be used (YES) or not (NO). This parameter has meaning only when the procedures are stated on the OPTCD parameter on a data definition (DD) statement or on the DCB macro.

Value Range: Not applicable

Default Value: NO. This default applies if you omit the RER parameter, or specify an incorrect value for it.

Associated Parmlib Member: None

RSU

```
RSU=xxxx

RSU={xxxx      }
    {OFFLINE   }
    {xx%       }
    {xxxxxxM   }
    {xxxxxxG   }
    {xxxxxxT   }
```

This parameter specifies the amount of central storage to be made available for storage reconfiguration. The frames in these storage increments are not to be used for long-term pages and will be designated the non-preferred area. (Long-term pages include SQA pages, common area fixed pages and LSQA or private area fixed pages associated with non-swappable address spaces.)

An address space is non-swappable:

- If the program name is in the program properties table (PPT) and the appropriate flags are set. (For more information about the PPT, see Chapter 77, “SCHEDxx (PPT, master trace table, and abend codes for automatic restart),” on page 729.)
- If ADDRSPC=REAL is specified on the JOB or EXEC statement.
- If the address space issues the TRANSWAP sysevent. The DONTSWAP and HOLD sysevents also make an address space non-swappable, but these specifications are considered to be of short duration, and associated LSQA and private area pages are not necessarily put into preferred storage.

During IPL, MVS assigns the V=R area to the low end of storage and assigns the current SQA to the high end of storage. Then, to satisfy an RSU specification, the system can define reconfigurable storage increments, starting with the first offline storage increment at the low end of storage and proceeding upward. If the system cannot satisfy the request from offline storage, it proceeds to define reconfigurable storage increments with the online storage increments. The system starts with the first online storage increment at the high end of storage and proceeds downward, defining reconfigurable

storage increments using only those online storage increments that do not contain V=R, SQA, LSQA, or nucleus frames.

After the system has defined the reconfigurable storage increments, it defines the remainder of the processor storage increments as the preferred area for long-term pages.

Value Ranges:

0 If your processor complex is not using PR/SM, or if your processor complex is using PR/SM but you are not using dynamic storage reconfiguration, set the RSU parameter to 0 (the default).

Values from 1–9999 are supported, but it is recommended that you either use the megabyte, gigabyte, or terabyte format described below, or use RSU=OFFLINE.

If you specify a value of 1-9999 without a qualifier (M, G, T, or %), the value is considered to be the number of the units, and the default storage increment size is used. For example, if your machine has a storage increment size of 64 megabytes, specifying 20 causes 20 units of 64M (1.25G in total) to be set aside for storage reconfiguration. Note that the storage increment size is entirely hardware dependent, based not only on the hardware model, but possibly also on the amount of real storage installed on the physical machine (not the LPAR). This means using an unqualified value of 1-9999 can have unexpected results, because its meaning can change dramatically with a simple upgrade to the amount of real storage on the system.

xxxxxxM
Specifies the RSU value in megabytes of storage. You can specify a six-digit value in z/Architecture mode. The value you specify might be rounded up. See *PR/SM Planning Guide* for specific information.

xxxxxxG
Specifies the RSU value in gigabytes of storage. You can specify a six-digit value in z/Architecture mode. The value you specify might be rounded up. See *PR/SM Planning Guide* for specific information.

xxxxxxT
Specifies the RSU value in terabytes of storage. You can specify a six-digit value in z/Architecture mode. The value you specify might be rounded up. See *PR/SM Planning Guide* for specific information.

OFFLINE
Indicates that all of the offline storage increments are to be made reconfigurable.

xx%
Indicates that the RSU is specified as a percentage of all storage, both online and offline. The actual number of storage increments that become reconfigurable will be rounded up to a whole number of storage increments.

Note:

1. An uncorrectable storage error might prevent you from reconfiguring the processor. If this happens, specify an additional storage increment (on the RSU parameter), at the next IPL, to increase the chance that reconfiguration might work.

2. In order to eliminate the impact of transition swaps, set REAL=0 and RSU=0 if you have no requirement for reconfiguring storage and no need to run a V=R job.

If you specify an RSU value that the system cannot fully satisfy, the system defines as many reconfigurable storage increments as possible and issues message IAR004I to indicate that the RSU parameter was not completely satisfied. The operator can then issue the display matrix (D M) command to determine how many increments were made available for reconfiguration.

However, if you specify an RSU value that is greater than the total amount of real storage available on the system, message IAR026I is issued by the system during IPL indicating an RSU over-specification condition and showing the amount of real storage available on the system. This message is followed by an IAR006A message prompting for a valid RSU value. You must select the right RSU value for the system. A large RSU value can ultimately cause system performance problems and degradation.

For information about how to choose the correct RSU value, see RSU Parameter Specification section and Specifying the RSU parameter section in *z/OS MVS System Commands*.

Syntax Examples for the RSU Parameter:

Example 1: RSU=0. Indicates that you are not using PR/SM, or that you are not using dynamic storage reconfiguration.

Example 2: RSU=25%. This example requests that 25 percent of all storage, including online and offline, be made available for storage reconfiguration.

Example 3: RSU=300M. This example requests that 300 megabytes of storage be made available for storage reconfiguration.

Default Value: 0. If the RSU parameter is omitted or specified as 0, all processor storage increments are available for preferred storage.

Associated Parmlib Member: None

RSVNONR

RSVNONR=nnnnnn

This parameter specifies the number of entries in the address space vector table that are to be reserved for replacing entries that are marked non-reusable. RSVNONR entries are used to keep the number of entries specified by MAXUSER and RSVSTRT 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.

A non-reusable address space is one where a job that ended had been running in a cross memory environment. When such a job ends, the system ends the address space and marks its associated ASVT entry non-reusable (unavailable) until all of the address spaces the job had cross memory binds with have ended.

The value you choose for RSVNONR depends on what subsystems you are running, how often you restart the subsystems, and how often you IPL. The more subsystems you have, the higher the RSVNONR value must be. Frequent restarts of subsystems also require a higher RSVNONR value. Conversely, the more often you IPL, the lower the RSVNONR value needs to be. The ASIDs are reset each time you IPL. If you IPL frequently you are less likely to exhaust the available ASIDs, and you need not reserve as many entries in the address

space vector table. For example, if you IPL once every three months, you need a RSVNONR value that is three times higher than if you IPL every month.

However, too large a buffer can cause the system to allocate more system resources than are needed during a given IPL. The excessive allocation of system resources because of incorporating too large a buffer can lead to potential problems such as the following examples:

- An aggressively large RSVNONR value could cause a wait state 040 to occur at IPL time. Wait state 040 means the system abended a task during nucleus initialization program (NIP) processing.
- An aggressively large MAXUSER, RSVSTRT, and RSVNONR can cause the system to allocate an excessive amount of storage, which can potentially lead to a wait state 204 reason 008 at IPL time. (Wait state 204 with reason 008: An error occurred during allocation initialization; the system could not obtain storage for device information.)

If started tasks or batch jobs that create non-reusable ASIDs end enough times, they will exhaust all available ASIDs and an IPL will be required. When loading the initial program is not an acceptable option, determine which programs caused the problems and fix them. For methods that prevent running out of ASIDs, see *z/OS MVS Programming: Extended Addressability Guide*.

Value Range: 0-32767. Note that the sum of the values specified for the RSVNONR, RSVSTRT, and MAXUSER system parameters cannot exceed 32767.

Default Value: 100

Associated Parmlib Member: None

RSVSTRT

RSVSTRT=nnnnnn

This parameter specifies the number of entries in the address space vector table (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 lookaside address space). The RSVSTRT entries are only used after the number of entries specified by MAXUSER is obtained. It is not recommended to depend on using the RSVSTRT entries under normal operating conditions. By reserving RSVSTRT entries in the ASVT for such address spaces, you can often avoid having to reinitialize the system because there is no available entry in the ASVT for a critical address space. For example, if library lookaside (LLA) ends and you issue a START LLA command to restart LLA, but no ASVT entry is available, LLA does not be restarted. However, if there are reserved entries for critical address spaces (such as LLA), it is more probable that an entry will be available and the LLA address space can be created without having to reinitialize the system.

However, too large a buffer can cause the system to allocate more system resources than are needed during a given IPL. The excessive allocation of system resources because of incorporating too large a buffer can lead to potential problems such as the following examples:

- The specification of RSVSTRT in conjunction with CSCBLOC=BELOW and MAXUSER can cause the system to allocate excessive storage below 16M. See CSCBLOC for details.
- An aggressively large MAXUSER, RSVSTRT, and RSVNONR can cause the system to allocate an excessive amount of storage, which can potentially lead

to a wait state 204 reason 008 at IPL time. (Wait state 204 with reason 008: An error occurred during allocation initialization; the system could not obtain storage for device information.)

If started tasks or batch jobs that create non-reusable ASIDs end enough times, they will exhaust all available ASIDs and an IPL will be required. When loading the initial program is not an acceptable option, determine which programs caused the problems and fix them. For methods that prevent running out of ASIDs, see *z/OS MVS Programming: Extended Addressability Guide*.

Value Range: 0-32767. Note that the sum of the values specified for the RSVSTRT, RSVNONR, and MAXUSER system parameters cannot exceed 32767.

Default Value: 5

Associated Parmlib Member: None

SCH

```
SCH={(aa)       }
    {(aa,L)     }
    {(aa,bb..)  }
    {(aa,bb..,L)}
```

This parameter specifies certain system characteristics. The two alphanumeric characters, represented by aa,bb.., specify one or more SCHEDxx members of SYS1.PARMLIB. SCHEDxx can include the following statement types:

MT Specifies the size of the master trace table, if one exists.

PPT Allows an installation to add entries to the program properties table.

RESTART
Allows an installation to add eligible restart codes to the restart codes table.

NORESTART
Allows an installation to delete eligible restart codes from the restart codes table.

If you specify the L option, the system displays the contents of the SCHEDxx SYS1.PARMLIB member at the operator's console as the system processes the member.

If a SCHEDxx member contains incorrect specifications, the system prompts the operator to specify another member. The response is in the form SCH=aa. To cancel the parameter, the operator replies by hitting the ENTER key. For more information about the SCHEDxx member, see Chapter 77, “SCHEDxx (PPT, master trace table, and abend codes for automatic restart),” on page 729.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: SCHEDxx

SMF

```
SMF=xx
```

This parameter specifies the parmlib member, SMFPRMxx, from which SMF obtains its options. The two alphanumeric characters, represented by xx, are appended to "SMFPRM" to name the member. For detailed information about SMF parameters, see Chapter 78, “SMFPRMxx (system management facilities (SMF) parameters),” on page 741.

Value Range: Any two alphanumeric characters.

Default Value: 00 (Specifies SMFPRM00, the IBM-supplied default parmlib member.

Associated Parmlib Member: SMFPRMxx

SMS

SMS=xx

This parameter specifies the parmlib member, IGDSMSxx, from which the storage management subsystem (SMS) will obtain its options when the system is initialized for partitioned data set extended (PDSE) support. The two alphanumeric characters, represented by xx, are appended to IGDSMS to name the member. NIP saves the name until SMS is initialized. If initialization of PDSE support fails, the IGDSMSxx parmlib member is selected from the IEFSSNxx parmlib member by using the ID=xx keyword (for detailed information, see Chapter 59, "IGDSMSxx (storage management subsystem definition)," on page 537).

Value Range: Any two alphanumeric characters.

Default Value: 00 (Specifies IGDSMS00, the IBM-supplied default parmlib member)

Associated Parmlib Member: IGDSMSxx

SQA

SQA=(a,b)

This parameter specifies the sizes of the virtual system queue area (SQA) and extended SQA. The subparameter "a" specifies the size of the SQA, located below 16MB. The subparameter "b" specifies the size of the extended SQA, located above 16MB. These values are added to the system's minimum SQA of eight 64KB blocks (or 512KB) and minimum extended SQA of approximately 8MB. Both the SQA and extended SQA are fixed in central storage as they are used.

The system also reserves additional SQA and ESQA storage for the I/O configuration. The amount of SQA and ESQA depends on the number of devices and control units installed. Because the system adds these amounts to the SQA and ESQA blocks specified on the SQA parameter in IEASYSxx, the actual amounts of SQA and ESQA allocated might be more than you specified. Also, MVS will not round the lowest address of SQA down, or the uppermost address of ESQA up, to cause these areas to start or end on a 64K, page, or segment boundary.

Note:

1. During IPL processing, the system reserves eight 64K blocks of virtual storage for SQA and thirty-five 64K blocks of virtual storage for extended SQA. During the IPL, both these storage areas are available to fulfill requests for SQA virtual storage. This SQA virtual storage is the minimum allocation for SQA and extended SQA. It is also called the 'initial allocation of SQA'.

   The system also reserves 8MB of extended SQA to be used for the common area page tables. This storage area is not available to fulfill requests for SQA virtual storage until NIP completes processing the CSA parameter. After the CSA parameter is processed when the size of the common area

and the amount of storage required for the common area page tables are known, the reserved storage not needed for the page tables becomes available to fulfill requests for extended SQA virtual storage.

2. During a quick start (that is, when the CLPA parameter is not specified), NIP determines if the currently specified SQA values are equal to the previously specified (cold start) values. If not, NIP issues an informational message and uses the cold-start SQA values. (See CLPA for a discussion of *cold start* and *quick start*.)
3. Large amounts of messages at IPL time (whether suppressed or displayed) increase ESQA storage needs. For example, the installation of unlabeled DASD without OFFLINE=YES specified used to result in many issuances of the following message:

   ```
   IEA311I UNLABELED DASD ON device
   ```

   To avoid common storage exhaustion, message IEA656I is issued if it finds at least 32 unlabeled DASD devices. If this happens, message IEA311I will not be issued for any additional unlabeled DASD, and subsequent unlabeled DASD devices will continue to be marked offline. See the description of message IEA656I in *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for details.
4. Additional device-related ESQA is required by self-describing devices (self-description is the ability of components of a system to provide unique identification information about themselves dynamically on request). It may be necessary to increase the ESQA specification when new devices are added to the system.

MVS determines the amount of expanded storage installed on the processor and automatically calculates the blocks of extended SQA necessary to initialize the expanded storage. This amount is added to the minimum system allocation and any increase you specify with the SQA parameter.

SQA Space Shortage: For a GETMAIN storage request for SQA storage where the request can be satisfied with storage either below or above 16MB, the virtual storage manager (VSM) will try to satisfy the request with space above 16MB. That is, VSM attempts to satisfy the request with space in the extended SQA, and, if such space is not available, VSM then attempts to satisfy the request with space in the extended common service area (extended CSA). If there is no space available above 16MB, VSM will then try to satisfy the request with space below 16MB, first from the SQA, and finally from the CSA.

For a GETMAIN request for SQA storage where the request must be satisfied with storage below 16MB, VSM tries to satisfy the request with space in the SQA, and, if none is available, VSM then tries to satisfy the request with space in the CSA.

Note: If excessive amounts of CSA or SQA are allocated, a warning message is generated and the parameter must be respecified. A warning message is also generated when the size of the entire common area below 16MB exceeds 8MB.

VSM keeps track of the remaining virtual SQA below 16MB, and informs the system resources manager, via a SYSEVENT macro, when the total of available virtual SQA plus available virtual CSA has reached two threshold values. The system resources manager reacts to the situation by issuing an “SQA shortage” message at each of the two thresholds. During an SQA shortage, the system inhibits the creation of new address spaces by disallowing start initiator commands, LOGON commands, MOUNT commands, and START commands for system tasks that require their own address spaces, such as START VTAM.

For additional information regarding SQA shortages and the two thresholds, see the "Virtual Storage Overview" topic in *z/OS MVS Initialization and Tuning Guide*.

SQA Space Shortage During NIP: During NIP processing, it is possible that the system's minimum allocation for SQA and extended SQA might be depleted before NIP processes the SQA parameter. If this situation occurs, you can increase the minimum SQA and/or extended SQA allocations. See the INITSQA parameter in the LOADxx parmlib member (Chapter 68, "LOADxx (system configuration data sets)," on page 619). However, if SQA is exhausted after processing local page data sets has begun, the value specified for the SQA parameter may be too low. Here, do not increase the initial SQA allocation. Instead, increase the SQA value in IEASYSxx.

Value Range: Each **a** value can be expressed as:

- A decimal number, n, indicating n 64KB (65536-byte) blocks. The number is 0 through 256.
- A decimal number followed by K, nK, indicating n 1KB blocks. The number is 0 through 16384.
- A decimal number followed by M, nM, indicating n 1MB (1024*1024-byte) blocks. The number is 0 through 16.

Each **b** value can be expressed follows. Note that the maximum values are accepted, but not recommended because they would result in a private region that is too small. Do not specify more than you think you might ever need.

- A decimal number, n, indicating n 64KB blocks. The number is 0 through 32511.
- A decimal number followed by K, nK, indicating n 1KB blocks. The number is 0 through 2080767.
- A decimal number followed by M, nM, indicating n 1MB blocks. The number is 0 through 2031.

When a and b are specified in kilobytes and megabytes, the system rounds the space up to a whole number of 64KB blocks.

Syntax Example: SQA=(4,5). The first value requests that, for the SQA, (4 x 64K) be added to the system's minimum virtual allocation. The second value requests that, for the extended SQA, (5 x 64K) be added to the system's minimum allocation.

This example can also be specified as SQA=(256K,320K).

Default Value: For the "a" subparameter, the default is 1. This means one 64K block will be added to the initial SQA allocation of 512K for a default size of 576K for the SQA. For the "b" subparameter, the default is 0. This means no 64K blocks will be added to the minimum extended SQA allocation of approximately 8MB.

Associated Parmlib Member: None

SSN

```
SSN={aa         }
    {(aa,bb,...)}
```

You specify this parameter to identify one or more IEFSSNxx parmlib members that contain the information needed to define and initialize selected subsystems. The two alphanumeric characters are appended to the characters, IEFSSN, to form the name of the parmlib member. NIP saves the parmlib name or names until the subsystems are to be initialized. For information about the

IEFSSNxx parmlib member, see Appendix A, "IEFSSNxx (subsystem definitions) - positional parameter form," on page 791.

Value Range: Any two alphanumeric characters

Default Value: SSN=00

Associated Parmlib Member: IEFSSNxx

SVC

```
SVC={aa         }
    {(aa,bb,...)}
```

The two alphanumeric characters, (aa, bb, and so on) are appended to IEASVC to form the name of the parmlib members. This member contains the installation-defined SVCs. NIP processing uses the members to place the specified SVCs in the SVC table.

Value Range: Two alphanumeric characters

Default Value: None

Associated Parmlib Member: IEASVCxx

SYSNAME

```
SYSNAME=name
```

This parameter optionally specifies the name that identifies this system. The name must be 1-8 characters long; the valid characters are A-Z, 0-9, $, @, and #. The name specified remains in effect for the duration of the IPL. The following values can override the value specified on the SYSNAME parameter:

- The value specified on the SYSNAME parameter in the IEASYMxx parmlib member.
- The value specified on the SYSNAME parameter in one or more alternate sets of parameters in IEASYSxx. The operator specifies the alternate parameters on the SYSP parameter on the REPLY command.

The value of SYSNAME is also the substitution text for the &SYSNAME system symbol. &SYSNAME can be specified in parmlib members to help support multisystem environments. For details about how to use &SYSNAME and other system symbols, see Chapter 2, "Sharing parmlib definitions," on page 33.

The system name defined by the SYSNAME parameter is used during the initialization of global resource serialization to identify the GRSDEF statement in the GRSCNFxx parmlib member that is to apply to this system. The system prompts the operator for a valid system name if the system name specified does not have the proper syntax. If the system name specified does not match any of the MATCHSYS keyword values in the GRSCNFxx parmlib member, the operator is informed, via an error message, to either reload the initial program or specify GRS=NONE. The operator also uses the system name defined by SYSNAME parameter to refer to the system in the system commands that control global resource serialization.

The system name defined by the SYSNAME parameter is also used to identify the originating system for messages in the multiple console support (MCS) hard-copy log and in the display created by the DISPLAY R command. Without specific SYSNAME= values in multisystem complexes, it is impossible

to determine which system actually issued the message in the hard-copy log. SYSNAME will be used in the SYSLOG heading and in all message formats where the system name is requested.

System logger may use the system name as part of the staging data set name. The system name specified here allows you to use a digit as the first character, but this is not valid in a staging data set name. If you use a digit as the first character in the system name, system logger substitutes 'STG' for the digit and uses part of the system name as the low level qualifier for the staging data set name. For complete information, see *z/OS MVS Setting Up a Sysplex* and look for the topic on *Naming Conventions for the Log Stream and DASD Log data sets.*

Value Range: 1-8 of these characters: A-Z, 0-9, $, @, #.

Default Value: If the system name is not specified in IEASYMxx or IEASYSxx, the default is one of the following:

- The processor name that is defined to HCD (if not in LPAR mode).
- The LPAR name for the processor.
- The VM userid, if the MVS image is running as a guest under VM/ESA.

Associated Parmlib Member: None

SYSP

```
SYSP={aa        }
     {(aa,bb...)}
```

Note: This parameter may be specified only by the *operator*. It cannot validly be specified in a system parameter list. It is included here for the sake of completeness.

The SYSP parameter specifies that one or more alternate system parameter lists (for example, IEASYS01, IEASYS02, and so on) that the system is to read after the default list IEASYS00. The two alphanumeric characters, represented by aa, bb, and so on, are appended to IEASYS to name the alternate lists. You can specify any number of alternate parameter lists. The specification cannot be prohibited by the OPI parameter.

When specifying alternate parameter lists on SYSP, consider the following points about the way the system processes parameter lists:

- If a system parameter is specified in both IEASYS00 and an alternate parameter list selected by the operator, the value in the alternate parameter list overrides the value in IEASYS00.
- You can also specify alternate parameter lists on the SYSPARM parameter in the IEASYMxx and LOADxx parmlib members. See “Step 2. Determine where to specify system parameters” on page 42 for information about how the system processes the alternate parameter lists that are specified in IEASYMxx, LOADxx, and on SYSP.
- The system processes the suffixes on SYSP from left to right. If the suffixes indicate two or more parameter lists that contain the same parameter, the value on the parameter that is processed *last* overrides the values on any parameters that were processed previously.

Do not specify system symbols on the SYSP parameter.

Example: The operator responds to SPECIFY SYSTEM PARAMETERS by entering:

```
R 00,SYSP=(01,02)
```

Assume that the two specified members contain the following parameters:

- IEASYS01: MLPA=(00,01),SQA=8
- IEASYS02: MLPA=02,SQA=10

NIP would accept MLPA=02 and SQA=10, in addition to other parameters contained in IEASYS00.

Value Range: Any two alphanumeric characters

Default Value: 00 (This specifies IEASYS00.)

Associated Parmlib Member: IEASYSxx

UNI

```
UNI=xx
```

This parameter specifies the CUNUNIxx member of parmlib that contains the image name and real storage information required to enable the Unicode Conversion Services using a customized image, rather than performing IPL with no UNI= parameter specified and running with the Unicode default image. See *z/OS Unicode Services User's Guide and Reference* for more information. The two alphanumeric characters (*xx*) are appended to CUNUNI to form the name of the parmlib member CUNUNIxx.

Value Range: Any two alphanumeric characters.

Default Value: None

Associated Parmlib Member: CUNUNIxx

VAL

```
VAL={aa         }
    {(aa,bb,...)}
```

This parameter specifies the VATLSTxx member or members of parmlib. Volume attribute processing reads this member or members during initialization to obtain *mount* and *use* attributes for direct access volumes. The *mount* and *use* attributes are set in the UCBs whose volume serial numbers are listed in the VATLSTxx members. If multiple members are specified, the lists are read in the order in which they appear in the VAL parameter. If a particular volume serial number appears on more than one entry, the volume attributes specified in the last entry for that volume serial will be accepted. (For additional information, refer Chapter 80, "VATLSTxx (volume attribute list)," on page 775.)

Value Range: Any two alphanumeric characters

Default Value: 00 (This means that VATLST00, if it exists, will be read.)

Associated Parmlib Member: VATLSTxx

VIODSN

```
VIODSN={dsname | IGNORE}
```

VIODSN specifies the name of the VSAM data set for holding information about journaled VIO data sets. For warm start processing, the system assumes that the name of the VSAM data set you specify is the same one used for the last IPL. If you do not specify a value for VIODSN or if the system cannot allocate or open the data set you specify, VIO journaling will not take place, and journaled VIO jobs from the previous IPL will have to be restarted. If you specify a new name, and the data set is allocated and opened, this new data set will be used, and will require the restart of previous jobs.

If you specify IGNORE, no VIO journaling is activated. No VIO journaling can take place until the next IPL.

In a multisystem environment, if you specify a unique name for each of your VIODSN data sets, IBM suggests that you not place these names in the SYSTEMS exclusion resource name list in parmlib member GRSRNLxx.

Before an IPL, the VIO journaling data set must have been allocated, and cataloged in the system master catalog.

Syntax Examples for the VIODSN Parameter: Table 25 shows syntax examples of the VIODSN parameter and the results they produce.

Table 25. Syntax examples for the VIODSN parameter

Example	Result
`VIODSN=SYS1.VIODSN`	The data set 'SYS1.VIODSN' will be used.
`VIODSN=IGNORE`	There will be no VIO journaling during this IPL.
`VIODSN=&SYSNAME;.STGINDEX`	Assuming the value specified on the SYSNAME= parameter is SYSTEMA, the data set 'SYSTEMA.STGINDEX' will be used.
`VIODSN=&SYSNAME; &SYSNAME;DATA.FILE`	Assuming the value specified on the SYSNAME= parameter is S1, the data set 'S1S1DATA.FILE' will be used.

Value Range: Only one data set can be specified. The data set can contain system symbols.

Default Value: SYS1.STGINDEX

Associated Parmlib Member: None

VRREGN

VRREGN=nnnn

This parameter specifies the default real-storage region size for an ADDRSPC=REAL job step that does not have a REGION parameter in its JCL. The numerical value of the operand (nnnn) indicates the real-storage region size in 1KB blocks.

Note: The following VRREGN values will prevent an ADDRSPC=REAL job step from running if it omits a REGION parameter:

- VRREGN value that is greater than the value of the REAL parameter or
- VRREGN value of zero.

Value Range: 0-9999

Default Value: 64 (This means a default REGION size of 64K.)

Associated Parmlib Member: None

WARNUND

WARNUND

This parameter asks that the system warn when finding an undefined system parameter rather than prompt for new system parameters. When the WARNUND parameter is found, it applies to all subsequent system parameter processing. If you want WARNUND processing for all IEASYSxx parmlib members, IBM suggests that you put WARNUND at the beginning of

IEASYS00 so that it covers all subsequent system parameters within IEASYS00 and any other IEASYSxx members processed. If you only need WARNUND processing before a specific parameter in an IEASYSxx member, you can place WARNUND just before that parameter. If you want WARNUND processing for your reply to IEA101A SPECIFY SYSTEM PARAMETERS, you must specify it within the reply text (earlier within that reply than any unknown parameter). For example, you might reply:

```
R 0,WARNUND,SYSP=01,NEWPARM=VALUE
```

Note: WARNUND only applies to primary system parameters that are listed in IEASYSxx and not to their associated sub-parms.

IBM suggests that you avoid using unknown parameters within this reply text that imply that you do not need WARNUND within the reply text.

Default Value: None

Associated Parmlib Member: None

ZAAPZIIP

ZAAPZIIP=NO|YES

You can use the ZAAPZIIP (or ZZ as its alias) parameter to request the system to run zAAP processor eligible work on zIIP processors when certain conditions are met.

The zAAP on zIIP function is active only when all of the following conditions are true:

- ZAAPZIIP=YES is specified, defaulted, or enforced.
- There are (or could be, through dynamic CPU addition) zIIPs defined to this LPAR.
- No zAAPs are defined to this LPAR, whether online, offline or in the reserved state.
- There are no zAAPs installed on the machine or, when there is at least one zAAP installed, the number of installed ZAAPs plus the number of installed zIIPs does not exceed the number of installed standard CPs. The installed processors mentioned in the preceding sentence are those in the configured state only; processors in the standby or reserved state do not apply.
- This LPAR is allowed to find out about the presence of zAAPs on the entire machine. This security protocol is controlled by the Global Performance Data Control setting, described in *Processor Resource/Systems Manager™ Planning Guide,IBM System z10*, SB10-7153.

If you specify ZAAPZIIP=NO on a machine that does not support zAAPs, the system issues message IEA998I and continues as if you had specified ZAAPZIIP=YES.

To correct an error you make in specifying ZZ, you must specify the value on ZAAPZIIP instead of ZZ. The system issues message IEA341A to indicate this situation.

Default Value: YES

Note: The default of ZAAPZIIP is NO in z/OS V1R9 and z/OS V1R10; and the default is YES as of V1R11. Only the alias ZZ is valid in z/OS V1R9 and z/OS V1R10.

Associated Parmlib Member: None

Chapter 55. IECIOSxx (I/O related parameters)

IECIOSxx contains records that the installation creates to accomplish the following tasks:

- Modify the time intervals which, when exceeded, result in the missing interrupt handler (MIH) detecting a missing interrupt. (MIH parameter)
- Modify the I/O timing limits. The system uses the I/O timing limits to monitor queued and active I/O requests. If an I/O request exceeds the time limit that you specify, the system abnormally ends the request. (MIH IOTIMING parameter)
- Specify whether an I/O Timing condition is allowed to trigger a HyperSwap, and if so, whether the timed-out I/O operation is posted back with permanent error. (MIH IOTHSWAP and IOTTERM parameters)
- Specify I/O timing HyperSwap options.
- Modify the device threshold values and recovery actions in the hot I/O detection table (HIDT). The system uses the HIDT information to detect, and recover from, hot I/O conditions. (HOTIO parameter)
- Specify recovery actions to recover from a hung interface (that is, a terminal condition). (TERMINAL parameter)
- Specify trace options other than the default for IOS component trace. (CTRACE parameter)
- Specify whether FICON® switch statistics gathering is to be enabled or disabled on a system. (FICON parameter)
- Specify whether IOS storage blocks are to be obtained in 24 or 31-bit storage. (STORAGE parameter)
- Specify whether to enable write protection on captured UCBs or as a result of a SETIOS=xx command. (CAPTUCB parameter)
- Specify whether the MIDAW facility is to be enabled or disabled on a system. (MIDAW parameter)
- Specify whether the HyperPAV facility is disabled, enabled or in base only mode. (HYPERPAV parameter)
- Specify whether IBM zHyperWrite data replication is enabled or disabled. (HYPERWRITE parameter)
- Specify the Encryption Key Management primary and secondary host names and the maximum number of concurrent socket connections that are allowed for the communication with the encryption key manager. (EKM parameter)
- Specify IOS recovery options. (RECOVERY parameter)
- Specify whether the High Performance FICON for System z (zHPF) facility is to be enabled or disabled on a system. (ZHPF parameter)

Parameter in IEASYSxx (or specified by the operator)

IOS=xx

The two alphanumeric characters (xx) are appended to IECIOS to identify the IECIOSxx parmlib member. If the IOS parameter is not specified, the system uses defaults for MIH processing, defaults for hot I/O, and defaults for IOS CTRACE processing.

To change the IECIOSxx member after initialization, you can specify the SET IOS=xx command. The SET IOS=xx command changes the I/O timing limits, MIH intervals and FICON statistics option. It does not, however, change the values initialized for hot I/O detection and it does not change the current IOS CTRACE options. Changing the hot I/O detection or recovery values requires a reIPL. Changing the CTRACE parmlib member or options requires the use of the TRACE operator command. For a description of the TRACE operator command, see *z/OS MVS System Commands*.

You can change the FICON, STORAGE, CAPTUCB, MIDAW, HYPERPAV, HYPERWRITE, EKM, RECOVERY, and ZHPF parameters dynamically using the SETIOS command. See *z/OS MVS System Commands*.

IBM-supplied default for IECIOSxx

There is no default IECIOSxx parmlib member. However, the system establishes default IOS component tracing options at IPL time.

Syntax rules for IECIOSxx

The following rules apply to the creation of IECIOSxx:

- Use columns 1 through 71 for data; columns 72 through 80 are ignored.
- Continuation is indicated by a comma followed by one or more blanks. The next record can start in any column.
- If a verb can have multiple keywords, they can be expanded to multiple records, however, a keyword and its value must be specified on the same record.
- An asterisk (*) in column 1 indicates a comment record. Comment records cannot be continued and that comments cannot appear in a parameter record.
- Each record must start with MIH, HOTIO, TERMINAL, FICON, STORAGE, CAPTUCB, EKM, or RECOVERY followed by one or more blanks, or must be a valid CTRACE, MIDAW, HYPERPAV, HYPERWRITE, or ZHPF specification in the format given as follows.
- If duplicate keywords are found, the last occurrence of the keyword is used.
- Two different verbs cannot be specified in the same record.
- The format for each record is as follows. The following sections describe the IECIOSxx I/O-related parameters and list-specific syntax requirements.

```
MIH parameter[,parameter]. . .
MIH IOTIMING mm:ss,parameter[,parameter]. . .
HOTIO parameter[,parameter]. . .
TERMINAL parameter[,parameter]. . .
CTRACE(parmlib_member_name)
FICON parameter
STORAGE parameter
CAPTUCB parameter
MIDAW=parameter
HYPERPAV=parameter
HYPERWRITE=parameter
EKM parameter
RECOVERY parameter
ZHPF=parameter
```

Missing interrupt handler (MIH)

The MIH detects missing interrupt conditions. The following device conditions qualify as missing interrupts if the specified time interval has elapsed:

- Primary status interrupt pending
- Secondary status interrupt pending
- Start pending condition
- Idle with work queued
- Mount pending

If the MIH detects a missing interrupt, processing will be done that is dependent on the detected condition. For any detected missing interrupt, the MIH performs one or more of the following actions:

- Invokes the device dependent MIH exit (if it exists)
- Records the condition in the SYS1.LOGREC data set
- Notifies the system operator
- Clears the condition
- Simulates an interrupt
- Redrives the device
- Requeues the I/O request

When specifying time intervals, consider the following:

- The MIH detects a missing interrupt condition within 1 second of the time interval that you specify.
- If the time interval is too short, a false missing interrupt can occur and cause early termination of the channel program. For example, if a 30-second interval is specified for a tape drive, a rewind might not complete before the MIH detects a missing interrupt.
- If the time interval is too long, a job or system could hang because the MIH has not yet detected a missing interrupt. For example, if a 15-minute time interval is specified for a tape drive used as an IMS™ log tape, the MIH could delay IMS for 15 minutes because of MIH detection.
- During IPL (if the device is defined to be ONLINE) or during the VARY ONLINE process, some devices may present their own MIH timeout values, via the primary or secondary MIH timing enhancement contained in the self-describing data for the device. The primary MIH timeout value is used for most I/O commands, but the secondary MIH timeout value may be used for special operations such as long-busy conditions or long running I/O operations. Any time a user specifically sets a device or device class to have an MIH timeout value that is different from the IBM-supplied default for the device class, that value will override the device-established primary MIH time value. This implies that if an MIH time value that is equal to the MIH default for the device class is explicitly requested, IOS will **not** override the device-established primary MIH time value. To override the device-established primary MIH time value, you must explicitly set a time value that is not equal to the MIH default for the device class.

Note:

1. To cancel MIH processing for specific devices, specify 00:00 for the time interval value (mm:ss) defined for the associated devices.
2. During IOS recovery processing, the system will override your time interval specification and may issue MIH messages and MIH error records in SYS1.LOGREC at this IOS determined interval.

3. Any dynamic change that causes a device's UCB to be deleted and then re-added will cause the device's MIH time interval to be reset to the default MIH setting for its device class. Once the dynamic change is completed, users should reissue the SETIOS MIH MVS command to reestablish any specific MIH setting for the device. For more information about the SETIOS MIH command, see *z/OS MVS System Commands*.

I/O timing

The I/O timing facility abnormally ends the following I/O requests that exceeded the I/O timing limits that are specified for a device:

- Queued requests that are waiting for execution.
- Start pending requests.
- Active requests.

The I/O timing facility can be enabled to trigger a HyperSwap when an I/O timeout occurs for a device that is monitored for HyperSwap. Optionally, the user can specify whether a timed-out I/O operation that initiates a HyperSwap is to be terminated or allowed to be started on the swap 'TO' device.

For any I/O requests that exceeds the I/O timing limit, the system performs the following actions:

- When the I/O timing trigger is not enabled for HyperSwap, or is enabled and the IOTTERM option is also enabled:
  - Abnormally ends the I/O request that exceeded the time limit, and does not requeue the request for execution.
  - Issues a message.
  - Writes an entry in the SYS1.LOGREC data set for the abnormally ended I/O request.
- When the I/O timing trigger is enabled for HyperSwap and the IOTTERM option is disabled:
  - Abnormally ends the I/O request that exceeded the time limit, and requeues the request for later execution on the swap 'TO' device at the completion of the HyperSwap.
  - Issues a message for the first timeout condition that triggers a HyperSwap on the associated DASD subsystem.
  - Writes an entry in the SYS1.LOGREC data set for the abnormally ended I/O request.

Installations that want the system to issue an I/O timing message and record in SYS1.LOGREC, but do not want to abnormally end the I/O request can do so with I/O timing message-only processing. Message-only processing allows the system to detect I/O timeout conditions, but, provides the ability to allow the user to decide which I/O requests should be terminated.

The system records data associated with the original I/O error, if any exists.

IBM recommendations for I/O timing limits: Within 1 second, the system abnormally ends an I/O request that exceeded the I/O timing limit. IBM suggests setting your I/O timing limits to 6 seconds or longer. An I/O timing limit of less than 6 seconds can result in a loss of service information that is required to correct a DASD subsystem hardware failure that triggered the long I/O event. In addition, a limit of less than 6 seconds can result in a loss of SYS1.LOGREC records.

Notes:

1. To cancel the I/O timing limit, (that is, to have no time limit on I/O requests) specify IOTIMING=00:00.
2. I/O timing is only supported for non-paging DASD devices.
3. An I/O timeout will not trigger a HyperSwap if message-only recovery is specified for the device or as the result of a timeout condition that is specified by an I/O driver program.

Interaction of MIH and I/O timing processing

You may specify that both MIH processing and I/O timing processing monitor your I/O requests. When the value you specify for the I/O limit is less than or equal to the value you specify for MIH, I/O timing processing takes precedence over MIH processing. The system will abnormally end the I/O request when it exceeds the I/O timing limit that you have specified. MIH processing will not get control.

When the value that you specify for the I/O timing limit is greater than the value that you specify for MIH, normal MIH recovery will be in effect until the I/O timing limit is reached. Once the I/O request exceeds the I/O timing limit that you have specified, the system will abnormally end the I/O request.

To get the full benefit of MIH recovery, your MIH time intervals plus your MIH time intervals for the HSCH and CSCH instructions should be at least 1 second less than your I/O timing limit.

I/O timing trigger for HyperSwap

The I/O timing trigger for HyperSwap defines an I/O timing timeout event to initiate a HyperSwap operation. This allows an application to switch over from failing devices to their mirror images without failing transactions for the end user. When the I/O timing timeout condition is detected for DASD devices monitored for HyperSwap, a HyperSwap event is triggered. If the HyperSwap operation is successful, the I/O request remains queued to the device so that it can be re-driven when the swap completes. If the HyperSwap operation is not successful, the I/O request is posted back with a permanent error.

Statements/parameters for MIH

Note: IBM suggests you not set a MIH time value for any device that establishes its own primary or secondary time interval. Overriding the device-supplied primary MIH timeout value might adversely affect MIH recovery processing for the device or device class. Refer to the specific device's reference manuals to determine if the device supports self-describing MIH time values. See the section "Missing interrupt handler (MIH)" on page 497 for more information.

CHAR=*mm*:*ss*

Specifies the MIH time interval to be used for the character reader device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss* = 00-59

Default: 03:00

COMM=*mm*:*ss*
Specifies the MIH time interval to be used for the communications device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm*= 00-99
- *ss* = 00-59

Default: 03:00

CTC=*mm*:*ss*
Specifies the MIH time interval to be used for the channel-to-channel device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss* = 00-59

Default: 03:00

DASD=*mm*:*ss*
Specifies the MIH time interval to be used for all direct access devices (DASD), where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss* = 00-59

In general, the default time interval (15 seconds) is sufficient for most DASD configurations. However, IBM suggests you not specify any MIH for DASD if you have any DASD hardware in your configuration that uses the primary or secondary MIH timing enhancement. See the section "Missing interrupt handler (MIH)" on page 497 for more information.

Default: 00:15

IOTDASD=*mm*:*ss*
Specifies the I/O timing limit to be used for all non-paging direct access storage devices (DASD), where *mm* is minutes and *ss* is seconds. The maximum I/O timing limit is 5,999 seconds. When IOTDASD is set to 00:00, I/O timing is not in effect for the DASD device class.

Value Range:

- *mm* = 00-99
- *ss* = 00-59

Note that initially, the time interval is set to 00:00, which indicated that the I/O timing facility is not active for any DASD.

Note also that the MSGONLY keyword can also be used to set I/O timing message-only processing on or off for all DASD devices.

Default: 00:00

DEV=devnum
Specifies the device number(s) for the device(s) for which specific time intervals are to be used. To specify more than one device number, use a comma to separate the device numbers. To specify a range of device numbers, use a hyphen to separate the beginning and ending device numbers. If more than one device number is specified, enclose the device numbers in parentheses.

Note: The DEV keyword must precede or follow the TIME keyword on the same record.

Value Range: 1 to 4 hexadecimal digits, optionally preceded by a slash (/).

Default: None

GRAF=*mm*:*ss*
Specifies the MIH time interval to be used for the graphics device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: 03:00

HALT=*mm*:*ss*
Specifies the MIH time interval to be used for monitoring Halt (HSCH) and clear (CSCH) subchannel operations. This keyword is device independent; setting it affects all devices on the system.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: 00:05

IOTHSWAP=YES|NO
Specifies whether the I/O timing facility is enabled for HyperSwap.

Default: NO. The I/O timing facility is not enabled to trigger a HyperSwap.

IOTTERM=YES|NO
Valid only when IOTHSWAP=YES. Specifies whether a timed out I/O operation that triggers a HyperSwap is terminated with a permanent error or allowed to be started on the swap 'TO' device.

Default: NO. The timed out I/O operation is allowed to be started on the swap 'TO' device.

MSGONLY=YES|NO
Specifies whether an I/O timeout condition is processed using full I/O timing recovery (MSGONLY=NO) or message only recovery (MSGONLY=YES). When MSGONLY=NO is specified and an I/O request exceeds the I/O timing interval, a message is issued to the operator, a record is written to SYS1.LOGREC, and the I/O request is abnormally terminated. When MSGONLY=YES is specified and an I/O request exceeds the I/O timing interval, a message is issued to the operator and a record is written to SYS1.LOGREC, however, the I/O request is NOT abnormally terminated. Instead, the request is left in the system.

Message-only processing allows the system to detect I/O timeout conditions, but, provides the ability to allow the user to decide which I/O requests should be terminated.

Note:

1. If more than one MSGONLY keyword appears on a record, the last valid MSGONLY keyword is used.
2. The MSGONLY keyword is only valid when the IOTDASD keyword is specified or when the DEV and IOTIMING keywords are specified.

Otherwise, the MSGONLY keyword is ignored. This implies that the MSGONLY keyword value only relates to devices that are affected by the IOTDASD or DEV/IOTIMING keywords.

Default: NO

MNTS=*mm*:*ss*
Specifies the MIH time interval to be used for monitoring 'mount pending' conditions for DASD and TAPE devices.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: 03:00

MOUNTMSG=YES|NO
Specifies that MIH is to issue all MIH mount pending messages.

Default: NO

STND=*mm*:*ss*
Specifies the MIH time interval to be used for all of the following device classes:

- Character readers (CHAR)
- Communications (COMM)
- CTCs (CTC)
- Graphics (GRAF)
- Tapes (TAPE)
- Unit records (UREC).

Specify the time interval as mm:ss, where mm is minutes and ss is seconds.

If you code STND following CHAR, COMM, CTC, GRAF, TAPE, or UREC, the value for STND overrides the values for those device classes. Similarly, if you code CHAR, COMM, CTC, GRAF, TAPE, or UREC following STND, the values for those device classes override the value for STND.

In this example, the value for the CTC device class is 4:00, because the value specified for STND overrides the value specified for CTC. However, the value for the tape device class is 5:00, because the value specified for TAPE overrides the value specified for STND. Thus, this MIH record sets the values for CHAR, COMM, CTC, GRAF, and UREC to 4:00, and sets the value for TAPE to 5:00.

```
SETIOS MIH,CTC=01:00,STND=04:00,DASD=00:10,HALT=00:08,TAPE=05:00
```

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: 03:00

TAPE=*mm*:*ss*
Specifies the MIH time interval to be used for the tape device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

IBM suggests the following time intervals, based on tape device class:

Tape Device Class	Time Interval
3480	5 minutes
3490	5 minutes
3490E	10 minutes
3490E (When loaded with IBM Enhanced Capacity Cartridge System Tape)	20 minutes

Note: You may specify either TAPE (such as 'TAPE=10:00'), or a combination of DEV and TIME, for example:

```
DEV=(4A1-4E7),TIME=10:00
```

The time interval you set with the TAPE parameter will apply to all tape device types. You cannot set time intervals for specific esoteric or generic groups.

IBM suggests you not specify any MIH for the tape device class if you have any tape device in your configuration that uses the primary or secondary MIH timing enhancement. See the section "Missing interrupt handler (MIH)" on page 497 for more information.

Default: 03:00

TEST

Specifies the system is to dynamically test MIH parameters and values. The TEST parameter is ignored during initialization. The system uses this parameter when you issue a SET IOS=xx command to syntax check the MIH updating.

TIME=*mm*:*ss*

Specifies the MIH time interval to be used for the devices identified on the DEV keyword, where *mm* is minutes and *ss* is seconds.

Note: The TIME keyword must precede or follow the DEV keyword on the same record.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: None

UREC=*mm*:*ss*

Specifies the MIH time interval to be used for the unit record device class, where *mm* is minutes and *ss* is seconds.

Value Range:

- *mm* = 00-99
- *ss*= 00-59

Default: 03:00

Syntax considerations for MIH:

- On MIH records, specify the DEV and TIME keywords as a pair. That is, use them together on one record to define time intervals for specific device numbers.

- On MIH records, only one pair of DEV and TIME keywords can appear on any one record. They can appear in any order, and can be separated by other keywords.
- On MIH records, if the same device number exists in more than one DEV keyword, the time interval specified for the last occurrence of the DEV keyword (containing the device number) will be used.
- You can code an MIH time interval and I/O timing limit on the same record when either of the following is true:
  - All the devices listed in the record are to be monitored by both the MIH and the I/O timing facility. In the following example, device A20 and devices A11, A12, and A14 will have both an MIH time interval of 5 seconds and an I/O timing limit of 11 seconds:

    ```
    MIH DEV=(A20,A11-A14),TIME=00:05,IOTIMING=00:11
    ```

  - A device class is to be monitored by the MIH, and an individual device (or range of devices) is to be monitored by the I/O timing facility.

    When the device class is DASD, the MIH will monitor all DASD devices, including those specified on the DEV parameter. The I/O timing facility will monitor only those devices specified on the DEV parameter. In the following example, the DASD device class will have an MIH time interval of 5 seconds. Device A20 and devices A11, A12, A13, and A14 will have an MIH time interval of 5 seconds, and will also have an I/O timing limit of 11 seconds.

    ```
    MIH DASD=00:05,DEV=(A20,A11-A14),IOTIMING=00:11
    ```

    When the device class is not DASD, the MIH will monitor the device class, and the I/O timing facility will monitor only those devices specified on the DEV parameter. In the following example, the tape device class will have an MIH time interval of 5 minutes. Device B12 is a DASD device, because only DASD devices can be monitored by the I/O timing facility. Device B12 will have an I/O timing limit of 10 seconds.

    ```
    MIH TAPE=05:00,DEV=B12,IOTIMING=00:10
    ```

- To request that the MIH monitor one device (or range of devices) and the I/O timing facility monitor device (or range of devices), code two separate records.
- You can establish I/O timing for the entire DASD device class by using a device class name of IOTDASD. In the example below, an I/O timing limit of 1 minute is established for all DASD devices.

  ```
  MIH IOTDASD=01:00
  ```

  - When the IOTDASD device class name is specified with the DEV and IOTIMING keywords, all devices specified on the DEV keyword will have an I/O timing limit equal to the limit specified on the IOTIMING keyword. In the example below, an I/O timing limit of 5 seconds is established for DASD devices 180 through 18F and an I/O timing limit of 30 seconds is established for all other DASD devices.

    ```
    MIH IOTDASD=00:30,DEV=(180-18F),IOTIMING=00:05
    ```

  - To request that the I/O timing facility monitor a device (or devices) with message-only processing active, use the MSGONLY keyword. In the following example, an I/O timing limit of 1 minute is established for all DASD devices. Furthermore, message-only processing is established for all DASD devices. This implies that if any I/O request to any DASD device exceeds the I/O timing limit, then the system will issue a message, record the condition in SYS1.LOGREC, and will NOT abnormally terminate the I/O request. Instead, the request is left in the system. If another I/O timing interval expires, the system will reissue a message, and rerecord the condition in SYS1.LOGREC.

    ```
    MIH IOTDASD=01:00,MSGONLY=YES
    ```

In the following example, an I/O timing limit of 10 minutes and message-only processing is established for device 180 through 18F.

```
MIH DEV=(180-18F),IOTIMING=10:00,MSGONLY=YES
```

To turn off message-only processing, either use the MSGONLY=NO keyword, or set the I/O timing limit for the device (or devices) to zero (00:00).

Note: Indicating IOTDASD=00:00 will not turn off I/O timing nor reset message-only processing for any devices that were explicitly set up for I/O timing using the DEV and IOTIMING keywords.

- You can code I/O Timing HyperSwap options to specify if the I/O timing facility is allowed to trigger a HyperSwap and if so, whether to terminate the I/O operation that is timed out or allow it to be started on the swap 'TO' device.

  The following conditions must be true before an I/O timeout will trigger a HyperSwap:

  – The I/O timing facility must be enabled for HyperSwap.
  – The product must be installed at a level that supports the I/O timing trigger.
  – Message-only recovery is not in effect for the device on which the timeout occurred and the timeout is not the result of timing specified by the I/O driver program.

  The following example specifies that the I/O timing facility is enabled for HyperSwap and that timed-out I/O operations are allowed to be started on the swap 'TO' device:

  ```
  MIH,IOTHSWAP=YES,IOTTERM=NO
  ```

- On MIH records, specify the IOTHSWAP and IOTTERM keywords as a pair. That is, use them together on one record to define the actions to be taken for an I/O Timing HyperSwap trigger.

Hot I/O (HOTIO)

"Hot I/O" refers to a hardware malfunction that causes repeated, unsolicited I/O interrupts. If such I/O interrupts are not detected, the system can loop or the system queue area (SQA) can be depleted. IOS tries to detect the hot I/O condition and perform recovery before the system requires a reIPL.

When the number of repeated, unsolicited I/O interrupts reaches the threshold value defined in the HIDT, IOS assumes that there is a hot I/O condition and proceeds to perform recovery actions. Recovery actions taken for a "hot" device depend on the type of device and its reserve status.

Table 26. HOTIO Parameters

Device	Non-ESCON		ESCON	
	Reserved	Not Reserved	Reserved	Not Reserved
DASD or dynamic pathing device	DFLT112	DFLT111	SDFT112	SDFT111
non-DASD and non-dynamic pathing device	--	DFLT110	--	SDFT110

You can use HOTIO statements in IECIOSxx to modify the HIDT, and to eliminate operator intervention where recovery actions defined in the HIDT (by default) require the operator to respond to a message.

Note: To cancel hot I/O processing (that is, to have no detection and no recovery from hot I/O conditions) specify 0 for the device threshold value — DVTHRSH=0.

HOTIO DVTHRSH=ddddd
Specifies the device threshold for unsolicited I/O interrupts, which must be reached for hot I/O recovery to begin.

Value Range: 0-32767

Default: 100

DFLT110=(options)
Specifies the recovery action to be taken for a non-DASD, non-dynamic pathing device. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (BOX,BOX)

DFLT111=(options)
Specifies the recovery action to be taken for a DASD or a dynamic pathing device that is not reserved or assigned. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (CHPK,BOX)

DFLT112=(options)
Specifies the recovery action to be taken for a DASD or a dynamic pathing device that is reserved or assigned. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (CHPK,BOX)

SDFT110=(options)
Specifies the recovery action to be taken for a non-DASD, non-dynamic pathing device that is attached to an IBM ESCON (or future) channel path. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (BOX,BOX)

SDFT111=(options)
Specifies the recovery action to be taken for a DASD or a dynamic pathing device that is not reserved or assigned and is attached to an IBM ESCON (or future) channel path. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (CUK,BOX)

SDFT112=(options)
Specifies the recovery action to be taken for a DASD or a dynamic pathing device that is reserved or assigned and is attached to an IBM ESCON (or future) channel path. (For the recovery actions that can be specified as options, see "Options for HOTIO recovery" on page 507.)

Default: (CUK,BOX)

BOX_LP=(device_classes)
Specifies which device classes should be boxed instead of performing channel path recovery when all of the following conditions are true:

- the device is defined on the CHPID being recovered.
- either all of the device's paths are undergoing recovery or this is the only path to the device.
- the device is currently reserved or assigned.

The device classes you can specify for BOX_LP are DASD, TAPE, UREC, COMM, CTC, and GRAF. You can also specify:

- NONE, to turn off BOX_LP processing for all device classes.
- ALL, to turn on BOX_LP processing for all device classes. ALL is the recommended setting.

Default: (ALL)

Options for HOTIO recovery

The parameters (DFLT110, DFLT111, DFLT112, SDFT110, SDFT111, and SDFT112) that are used to specify recovery actions have default options defined in the HIDT. Each of these parameters has a default option for non-recursive hot I/O conditions and a default option for recursive hot I/O conditions. You can override one or both options.

For example, if you want to override the DFLT111 non-recursive default option (CHPK) so that the recovery action would have to be obtained from the operator, code: DFLT111=(OPER,). Note that the options are enclosed in parentheses and separated by a comma.

The following options can be specified for both non-recursive and recursive hot I/O conditions. Code the options with the DFLT110, DFLT111, DFLT112, SDFT110, SDFT111, and SDFT112 parameters, and use the following format:

```
Parameter=(a,b)
```

a the action to be taken for a non-recursive hot I/O condition

b the action to be taken for a recursive hot I/O condition.

Option	Meaning and Use
BOX	Specifies that the device is to be forced offline.
CHPF	Specifies that the channel path over which the last interrupt for the hot device was received is to be forced offline.
CHPK	Specifies that channel path recovery is to be initiated for the channel path over which the last interrupt for the hot device was received. If successful, the channel path is to remain online. **Note:** If a device on the channel path undergoing channel path recovery belongs to a device class specified in the BOX_LP=(...) parameter, then that device may become boxed instead of undergoing channel path recovery.
OPER	Specifies that the recovery action is to be obtained from the operator. **Note:** There are conditions when the message issued to prompt the operator will be issued using the Disabled Console Communication Facility (DCCF). IBM suggests that you do not choose this option unless the installation can properly handle responding to messages that were issued using DCCF.
CUK	Specifies that control unit recovery is to be initiated for the channel path over which the last interrupt for the hot device was received. **Note:** The CUK option is valid only for IBM ESCON channel paths (SDFT110, SDFT111, and SDFT112).

TERMINAL

TERMINAL BOX_LP =(*device_classes***)**

Specifies special recovery actions for recovering from a terminal or hung interface condition. The TERMINAL statement specifies which device classes should be boxed instead of performing channel path recovery to recover from a hung interface (or terminal condition). Box processing will override channel path recovery processing when all of the following are true:

- the device is defined on the CHPID being recovered
- either all of the device's paths are undergoing recovery or this is the only path to the device.
- the device is currently reserved or assigned.

The device classes you can specify for BOX_LP are DASD, TAPE, UREC, COMM, CTC, and GRAF. You can also specify:

- NONE to turn off BOX_LP processing for all device classes.
- ALL to turn on BOX_LP processing for all device classes. ALL is the recommended setting.

Default: (ALL)

Syntax considerations for TERMINAL

- If multiple TERMINAL statements are found, only the last valid one specified applies. All others will be ignored.
- TERMINAL statements that are found in IECIOSxx members that are activated via the SET IOS=xx operator command will be processed so you can dynamically change the setting for this value, or make the initial specification.

CTRACE

IOS traces events during its processing that may aid in the debugging of problems involving the IOS component. You can use the CTRACE statement in IECIOSxx to identify the SYS1.PARMLIB member that contains the IOS component trace options. If no CTRACE statement is specified, IOS component trace is initialized with default minimum trace options. For more information about IOS component tracing, see *z/OS MVS Diagnosis: Tools and Service Aids*.

CTRACE(*parmlib_member_name***)**

where parmlib_member_name is an 8-character SYS1.PARMLIB member name which must be in the format CTnIOSxx, where:

n is an alphanumeric character that specifies the source of the member. IBM-supplied members use “I”.

xx is any two alphanumeric characters.

Default: None

Note that if the CTRACE parameter is not used, then IOS component tracing is activated to trace minimum operations.

For more information about specifying options for the IOS component trace, or for more information IOS minimum operations see *z/OS MVS Diagnosis: Tools and Service Aids*. For information about coding the component trace member of SYS1.PARMLIB, see Chapter 26, “CTncccxx (component trace parameters),” on page 283.

On CTRACE records, you cannot indicate more than one CTRACE specification per record. That is, once "CTRACE(xxxxxxxx)" has been found, the rest of the record is treated as comments.

FICON

Use the FICON statement to enable or disable switch statistics gathering (FICON STATS) on a system.

FICON STATS

If enabled, FICON switch statistics are gathered on each system for all online FICON switches. In a sysplex environment it is not necessary for all systems to gather statistics, and, in fact, doing so might lead to I/O contention issues. If no FICON statement is specified, the system gathers switch statistics by default. To reduce I/O contention at FICON switches, IBM suggests that when in a sysplex environment FICON switch statistics gathering be disabled on all but one system.

FICON STATS=YES|NO
Specifies whether FICON switch statistics gathering is enabled or disabled on a system.

Tip: If you specify STATS=NO to disable FICON statistics gathering, you might also want to turn off FICON Director Activity Reporting in Resource Measurement Facility™ (RMF).

Default: Yes

STORAGE

Use the STORAGE statement to set storage properties for IOS. If no STORAGE statement is specified, IOS blocks will be obtained in 31-bit storage.

STORAGE IOSBLKS=24|31
Specifies if IOS blocks will be obtained in 24 or 31 bit storage.

Default: 31

CAPTUCB

Use the CAPTUCB statement to enable or disable write protection on captured UCBs. If no CAPTUCB statement is specified, write protection will be enabled on the system by default.

CAPTUCB PROTECT=YES|NO
Specifies whether to enable write protection on captured UCBs.

Default: YES

MIDAW

Use the MIDAW statement to enable or disable the MIDAW facility on a system. If no MIDAW statement is specified, the MIDAW facility is enabled on the system by default.

MIDAW=YES|NO
Specifies whether the MIDAW facility is enabled or disabled on a system.

Default: YES

HYPERPAV

Use the HYPERPAV statement to indicate whether HyperPAV mode is to be used and, if so, whether I/O is only to be run on non-PAV-alias devices. The absence of this keyword in the parmlib member will cause no change to the system. For the initial IPL, the absence of the HYPERPAV keyword will cause the system to IPL in HYPERPAV=NO mode.

HYPERPAV=YES|NO|BASEONLY
Specifies whether HyperPAV mode is to be used.

BASEONLY
I/O is to be run only on non-PAV-alias devices in HyperPAV mode.

NO Do not use HyperPAV mode.

YES Use HyperPAV mode.

Default: NO

HYPERWRITE

Use the HYPERWRITE statement to enable or disable IBM zHyperWrite data replication on a system. If no HYPERWRITE statement is specified, IBM zHyperWrite data replication is enabled on the system by default.

HYPERWRITE=YES|NO
Specifies whether IBM zHyperWrite data replication is enabled or disabled on a system-wide basis.

Note: Certain exploiting applications, such as DB2®, can also provide their own controls to enable or disable the use of this function.

IBM zHyperWrite processing can be used by I/O drivers, such as Media Manager, for certain write I/O operations to perform software mirroring to peer-to-peer remote copy (PPRC) devices that are monitored for HyperSwap processing (with GDPS or TPC-R). IBM zHyperWrite data replication can be used to reduce latency in these HyperSwap environments. For maximum benefit, IBM zHyperWrite data replication should only be used when all synchronously mirrored relationships are managed by HyperSwap. Devices support IBM zHyperWrite data replication when the following conditions are true:

- The devices support IBM zHyperWrite data replication. Both the primary and secondary devices in a synchronous PPRC relationship must support this function.
- The devices in the synchronous PPRC relationship are managed by HyperSwap (either GDPS HyperSwap or TPC-R HyperSwap).

NO IBM zHyperWrite data replication is disabled.

YES
IBM zHyperWrite data replication is enabled system-wide.

Default: YES

EKM

Specify the encryption key management primary and secondary host names and the maximum number of concurrent socket connections that are allowed for the communication with the encryption key manager. By default, the primary and the secondary host names for the encryption key manager are set to NONE. Use the MAXCONN keyword to set the maximum number of socket connections that are possible with the encryption key manager. Some of the socket connections can remain open for faster communication. Use the MAXPCONN keyword to set the number of connections that remain open for faster communication. By default, the maximum number of socket connections to the encryption key manager is 255; eight socket connections are permanently connected. However, if the sockets for communication are not needed, the sockets are not opened.

In-band tape encryption requires that the IOS address space has security permission for a z/OS UNIX System Services segment. The z/OS UNIX System Services segment is only for TCP/IP connectivity. UID(0) or super user ability is not required. For example, in RACF, issue the following command:

```
ADDUSER IOSAS OMVS(UID(xxxx) HOME('/'))
where xxxx is a unique user id.
```

PRIMARY=*host_name***[:***port***|,PRIPORT=***port***]|***ipv4_address***[:***port** **|,PRIPORT=***port***]|***ipv6_address***[,PRIPORT=***port***]| NONE**

Specifies the host name, IPv4 address or IPv6 address and optional port number of the primary key manager. The primary host is used exclusively unless a failure occurs and all attempts to retry fail. In subsequent requests after a failure, the primary host is retried before the use of the secondary host. When a connection to the primary host is established again, normal operation continues.

*host_name***[:***port* | **,PRIPORT=***port***]**

The host name of the encryption key manager.

port The port number of the encryption key manager. The specification of port is mutually exclusive with the PRIPORT keyword. If port is not specified and the PRIPORT keyword is not specified, the default port is 3801.

PRIPORT=*port*

Specifies the port number for the host name or IP address for the primary key manager. PRIMARY= must be specified for PRIPORT= to be valid.

Default: 3801 (if not specified on the PRIMARY keyword)

*ipv4_address***[:***port* | **,PRIPORT=***port***]**

The IP address of the encryption key manager. The IP addresses must be specified as a dotted decimal quad: *ddd.ddd.ddd.ddd*.

port The port number of the encryption key manager. The specification of port is mutually exclusive with the PRIPORT keyword. If port is not specified and the PRIPORT keyword is not specified, the default port is 3801.

PRIPORT=*port*

Specifies the port number for the host name or IP address for the primary key manager. PRIMARY= must be specified for PRIPORT= to be valid.

Default: 3801 (if not specified on the PRIMARY keyword)

ipv6_address[**,PRIPORT=***port*]
The IP address of the encryption key manager in IPv6 format (for example, ::FFFF:127.0.0.1 or 2001:0db8:85a3:08d3:1319:8a2e:0370:7344).

The optional port number for an IPv6 address must be specified with the PRIPORT keyword.

PRIPORT=*port*
Specifies the port number for the host name or IP address for the primary key manager. PRIMARY= must be specified for PRIPORT= to be valid.

Default: 3801 (if not specified on the PRIMARY keyword)

NONE
No encryption key manager is specified.

Default: None.

SECONDARY=*host_name***[:***port* **|,SECPORT=***port***]** | *ipv4_address***[:***port* **|,SECPORT=***port***]**| *ipv6_address***[,SECPORT=***port***]**| **NONE**
Specifies the host name, IPv4 address, or IPv6 address and optional port number of the secondary key manager.

*host_nam***e[:***port* | **,SECPORT=***port*]
The host name of the encryption key manager.

port The port number of the encryption key manager. The specification of port is mutually exclusive with the SECPORT keyword. If port is not specified and the SECPORT keyword is not specified, the default port is 3801.

SECPORT=*port*
Specifies the port number for the host name or IP address for the primary key manager. SECONDARY= must be specified for SECPORT= to be valid.

Default: 3801 (if not specified on the SECONDARY keyword)

ipv4_address[**:***port* | **,SECPORT=***port*]
The IP address of the encryption key manager. The IP addresses must be specified as a dotted decimal quad: *ddd.ddd.ddd.ddd*.

port The port number of the encryption key manager. The specification of port is mutually exclusive with the SECPORT keyword. If port is not specified and the SECPORT keyword is not specified, the default port is 3801.

SECPORT=*port*
Specifies the port number for the host name or IP address for the secondary key manager. SECONDARY= must be specified for SECPORT= to be valid.

Default: 3801 (if not specified on the SECONDARY keyword)

ipv6_address[**,SECPORT=***port*]
The IP address of the encryption key manager in IPv6 format (for example, ::FFFF:127.0.0.1 or 2001:0db8:85a3:08d3:1319:8a2e:0370:7344). The optional port number for an IPv6 address must be specified with the SECPORT keyword.

SECPORT=*port*
Specifies the port number for the host name or IP address for the secondary key manager. SECONDARY= must be specified for SECPORT= to be valid.

Default: 3801 (if not specified on the SECONDARY keyword)

NONE
No encryption key manager is specified.

Default: None.

MAXCONN=*ddd*
Specifies the maximum number of concurrent socket connections for Encryption Key Management. If high network stress occurs because of the high socket utilization for Encryption Key Management, this number can be lowered.

Value range: 1 to 255

Default: 255

MAXPCONN=*ddd*
Specifies the maximum number of concurrent socket connections for Encryption Key Management that remain open to prevent the overhead of opening and closing socket communication. This MAXPCONN keyword specifies the number of total connections reserved to be permanent connections.

Value range: 0 to the number specified in MAXCONN

Default: 8

The following figure shows syntax examples. To prevent contention while updating the EKM parameter, it is suggested that all operands are specified on the same EKM statement.

```
EKM PRIMARY=key.manager.com:3801,SECONDARY=key.manager.com
EKM PRIMARY=127.0.0.1:3801,SECONDARY=127.0.0.1
EKM PRIMARY=127.0.0.1,PRIPORT=3801,
    SECONDARY=127.0.0.1,SECPORT=3801
EKM PRIMARY=::1
EKM PRIMARY=::FFFF:127.0.0.1,PRIPORT=3801
EKM PRIMARY=2001:0db8:85a3:08d3:1319:8a2e:0370:7344,
    PRIPORT=3801
EKM PRIMARY=NONE,SECONDARY=NONE
EKM MAXCONN=255,MAXPCONN=8
```

RECOVERY

Use the RECOVERY statement to enable or disable certain IOS recovery options on a system. The following recovery options are available:
- Limited recovery time
- Path recovery
- DCCF usage

DCCF usage

If a no-paths or intervention-required condition is detected, MSGIOS002A or MSGIOS003A may be issued during IOS-enabled interrupt processing. If the error is on a paging device that is not currently enabled for hyperswap, IOS-enabled interrupt processing will not issue the IOS002A or IOS003A message, but will invoke the Disabled Console Communication Facility (DCCF) to issue a IOS115A WTOR. DCCF prevents all other work from running on the system until the operator replies to the WTOR. Loading a non-restartable wait state is preferable to issuing the IOS115A WTOR message.

The DCCF keyword is used to specify whether IOS-enabled interrupt processing will load a non-restartable wait state instead of issuing the IOS115A WTOR if a no-paths or intervention-required condition occurs on a paging device.

DCCF=MESSAGE

If DCCF=MESSAGE is specified, when a no-paths or intervention-required condition occurs on a paging device, IOS-enabled interrupt processing will invoke DCCF to issue the IOS115A WTOR. Refer to *z/OS MVS Planning: Operations* for information about DCCF processing. DCCF=MESSAGE is the default.

DCCF=WAIT_STATE

If DCCF=WAIT_STATE is specified, when a no-paths or intervention-required condition occurs on a paging device, IOS-enabled interrupt processing will load a non-restartable wait state X'140'. The wait state reason code will be 0000 for a no-paths condition and 0001 for an intervention-required condition. Message IOS140W will be issued with an indication of the error condition.

IBM recommends setting RECOVERY DCCF=WAIT_STATE in IECIOSxx to avoid the complexity of DCCF processing for the IOS115A WTOR message.

Limited recovery time

A LIMITED_RECTIME between 2 and 14 seconds can be specified for either Direct Access Storage Devices (DASD) or only devices that have the I/O timing facility enabled. When the limited recovery time function is enabled, IOS will use this value to time an IOS recovery I/O request to the qualified devices. If this I/O request suffers a start pending, a missing channel end or device end, or an interface timeout condition, the IOS recovery I/O request will not be retried on that device-path. To disable the limited recovery time function, specify LIMITED_RECTIME=0.

When specifying limited recovery time interval, consider the following situations.

- If the limited recovery time interval is longer than the I/O timing time interval for the device, an I/O time out condition is detected first and exploiting the limited recovery time function provides no value.
- If the limited recovery time interval is longer than the MIH time interval for the device, the MIH time out condition is detected first and exploiting the limited recovery time function provides no value.

Attention: With the specification of LIMITED_RECTIME, a Sense Path Group ID (SNID) CCW timeout on a device-path will be considered a permanent error, which will result in the path being taken offline. If the SNID to the last available path also times out, the device will be boxed. Therefore, the LIMITED_RECTIME option should be used with caution by installations that want to reduce the allowable time for a recovery I/O request but are also able to handle device availability issues. Customers who consider exploiting the limited recovery time function are making a conscious trade-off between minimizing IOS recovery elapsed time and 100% availability of I/O resources.

Note that, in general, specifying LIMITED_RECTIME will not help in the case of a no paths condition and will not cause a device to be boxed unless a reserve is lost.

Path recovery

When you define your I/O configuration, many devices share common hardware components (such as channels, channel cards, switches, control unit ports, control unit adapters, and fiber-optic links). For example, all devices for a specific control unit definition share hardware components because they share channels and control unit ports. Therefore, when a hardware-related error occurs on a channel path, multiple devices are affected.

When an error occurs on a channel path, the system performs path recovery, which consists of issuing one or more recovery-related I/Os to test the channel path to see if it is still usable. If path recovery determines that the channel path is no longer usable, the path is removed (varied offline) from the affected device. Otherwise, the channel path remains online to the device.

Path recovery is typically performed one device at a time. This means that when an error occurs on one device, only that device is processed. Errors on other devices are processed independently, even if they share common hardware components. This may affect application performance since the application is delayed while the system performs path recovery and then retries the original I/O request. If the application uses multiple devices that share a failing or malfunctioning hardware component, additional errors are encountered and further delays occur.

Additionally, certain types of path errors can be intermittent. That is, an error occurs, but path recovery is successful, so the path is not removed from the device. This also affects performance because applications might encounter errors multiple times. If this occurs, you may need to manually remove the bad path or paths from the affected devices to stop the errors from occurring.

Specify a PATH_SCOPE of either CU or DEVICE to enable path recovery either for all devices that are attached to the control unit (CU) or on a device-by-device basis (DEVICE). The default is PATH_SCOPE=DEVICE.

The PATH_SCOPE option of CU, along with the PATH_THRESHOLD and PATH_INTERVAL options, allows you to reduce the elapsed time that it takes for the system to recover from channel path-related errors and helps prevent system performance problems that can occur when a significant amount of time is spent in repetitive channel path error recovery.

PATH_SCOPE=CU

If PATH_SCOPE=CU is specified and path recovery determines that a channel path needs to be removed from a device, the path is removed from all devices defined to the control unit. Additionally, for intermittent channel path errors, the system

collects error statistics over a period of time, and if the number of errors reaches or exceeds a threshold value, the channel path is removed from all devices defined to the control unit. The time period and threshold are controlled by the PATH_INTERVAL and PATH_THRESHOLD parameters as follows:

- PATH_INTERVAL specifies the length of time to monitor for errors (for example, 10 minutes).
- PATH_THRESHOLD specifies the minimum number of path-related errors that must occur every minute for the specified period of time (PATH_INTERVAL) before the path from the failing hardware component is removed from all the affected devices.

For example, specifying a PATH_INTERVAL of 10 (minutes) and a PATH_THRESHOLD of 20 (errors per minute) means that at least 20 errors must occur every minute for 10 consecutive minutes before the path from the failing hardware component is removed from all the affected devices.

Note:

1. Do not set the interval and threshold values to a very low value (for example: 1) because this may interfere with normal system recovery and cause the system to remove channel paths unnecessarily.
2. Changing the value of the interval and the threshold keywords will not take effect until the next error is encountered and will not affect actions previously taken.

PATH_SCOPE=DEVICE

If PATH_SCOPE=DEVICE is specified, then path recovery is on a device-by-device basis and no monitoring of intermittent errors is performed.

Note: PATH_INTERVAL and PATH_THRESHOLD are not valid keywords with PATH_SCOPE=DEVICE.

Statements/parameters for RECOVERY

LIMITED_RECTIME=*ss*

Specifies the time in seconds to be used for certain IOS recovery functions. Specify 0 seconds to disable the limited recovery time function. If the limited recovery time function is disabled, IOS continues to use it's pre-defined value to time the IOS recovery I/O requests. The limited recovery time function is disabled by default.

Value Range: 2-14 (seconds)

DEV=DASD|IOTIMING

Specifies the devices that exploits the limited recovery time function. Valid options are all DASD devices or only devices that have the IOTIMING facility enabled.

PATH_SCOPE={CU|DEVICE}

Specifies whether the scope of the path recovery is for all devices attached to the control unit (CU) or on a device-by-device basis (DEVICE).

Default: DEVICE

PATH_INTERVAL=*nnn*

Specifies the length of time to monitor for channel path errors in minutes. This keyword can only be used when PATH_SCOPE=CU is specified or when PATH_SCOPE=CU is the current value for the scope of the path recovery.

Value Range: 1-10 (minutes)

Default: 10

PATH_THRESHOLD=*nnn*
Specifies the minimum number of channel path-related errors that must occur every minute for the specified period of time (PATH_INTERVAL) before IOS takes action. This keyword can only be used when PATH_SCOPE=CU is specified or when PATH_SCOPE=CU is the current value for the scope of the path recovery.

Value Range: 1-100 (errors)

Default: 10

DCCF=MESSAGE|WAIT_STATE
Specifies the action that IOS-enabled interrupt processing will take when a no-paths or intervention-required condition occurs on a paging device. With the MESSAGE option, DCCF will be invoked to issue the IOS115A WTOR. With the WAIT_STATE option, a non-restartable wait state X'140' will be loaded.

IBM recommends setting RECOVERY DCCF=WAIT_STATE in IECIOSxx to avoid the complexity of DCCF processing for the IOS115A WTOR message.

Default: MESSAGE

Syntax example

```
RECOVERY PATH_SCOPE=CU,PATH_INTERVAL=10,PATH_THRESHOLD=100
```

ZHPF

Use the ZHPF statement to enable or disable the High Performance FICON for System z (zHPF) facility on a system. If no ZHPF statement is specified, the zHPF facility is disabled on the system by default.

ZHPF=YES|NO
Specifies whether the zHPF facility is enabled or disabled on a system.

Default: NO

Syntax examples for IECIOSxx

In Figure 19 on page 518, the last two MIH records specify IOTIMING=00:00 and TIME=00:00. These specifications cancel MIH or I/O timing processing for the device numbers on the associated DEV keyword. Similarly, you can cancel hot I/O processing by specifying:

```
HOTIO DVTHRSH=0
```

Note: For 3800 devices, the default of 3 minutes for the MIH interval might not be sufficient. The recommended MIH interval for a 3800 is 5 minutes.

```
HOTIO DVTHRSH=200
MIH STND=02:30,DASD=00:10,DEV=03C0,TIME=01:30
MIH DEV=(2E8-2FF,7300-7370),TIME=00:30
MIH 3851=15:00
HOTIO DFLT110=(BOX,BOX)
HOTIO DFLT111=(CHPK,BOX)
HOTIO DFLT112=(CHPK,BOX)
HOTIO SDFT110=(BOX,BOX)
HOTIO SDFT111=(CUK,BOX)
HOTIO SDFT112=(CUK,BOX)
HOTIO BOX_LP=(DASD,TAPE,UREC,COMM,CTC,GRAF,CHAR,ALL)
HOTIO BOX_LP=(DASD,TAPE,UREC)
HOTIO BOX_LP=(ALL)
MIH IOTIMING=00:12,DEV=(2E8-2FF,730-737)
MIH IOTIMING=00:12,DEV=200,MSGONLY=YES
MIH IOTDASD=10:00,MSGONLY=YES
MIH CTC=02:30,IOTIMING=00:00,DEV=(180-187,230,B10-B17)
MIH TIME=00:00,DEV=(180-187,230,B10-B17)
TERMINAL BOX_LP=(DASD)
TERMINAL BOX_LP=(TAPE,UREC)
TERMINAL BOX_LP=(ALL)
CTRACE(CTIIOS00)
CAPTUCB PROTECT=YES
FICON STATS=YES
STORAGE IOSBLKS=31
MIDAW=YES
HYPERPAV=YES
ZHPF=YES
HYPERWRITE=YES
```

Figure 19. Syntax examples for IECIOSxx

Chapter 56. IEFSSNxx (subsystem definitions) - keyword parameter form

Note: IBM suggests that you use the keyword parameter form of the IEFSSNxx parmlib member, which is described here. However, the positional parameter form of the IEFSSNxx parmlib member is still supported. See Appendix A, "IEFSSNxx (subsystem definitions) - positional parameter form," on page 791 for more information. Subsystems defined in the keyword parameter form of the IEFSSNxx parmlib member can use dynamic SSI services, while subsystems defined in the positional form of the IEFSSNxx parmlib member cannot use dynamic SSI services. See *z/OS MVS Using the Subsystem Interface* for more information about dynamic SSI services.

IEFSSNxx contains parameters that define the primary subsystem and the various secondary subsystems that are to be initialized during system initialization. IEFSSNxx allows you to:

- Name the subsystem initialization routine to be given control during master scheduler initialization.
- Specify the input parameter string to be passed to the subsystem initialization routine.
- Specify a primary subsystem name and whether you want it started automatically.
- Specify whether the initialization routines for the subsystem are to run in parallel.

For information about writing subsystems, see *z/OS MVS Using the Subsystem Interface*.

Unless you are starting the Storage Management Subsystem (SMS), specify the primary subsystem (JES) first. Some subsystems require the services of the primary subsystem in their initialization routines. Problems can occur if subsystems that use the subsystem affinity service in their initialization routines are defined before the primary subsystem. If you are starting SMS, specify its record before you specify the primary subsystem record.

Before z/OS V1R12, the order in which the subsystems were defined depended on the order in which they were specified in the IEFSSNxx parmlib member on the SSN parameter. Beginning with z/OS V1R12, you can specify the BEGINPARALLEL statement that allows the initialization routines for any subsystem that supports parallel processing to be invoked in parallel. For the SMS subsystem or any subsystem that does not support parallel processing, be sure to specify the BEGINPARALLEL statement after you specify the subsystem definitions. For the SMS subsystem, if the BEGINPARALLEL statement is encountered before the SUBSYS statement, the system issues message IEF009E about potential errors. See the BEGINPARALLEL statement in Statements/parameters for IEFSSNxx.

The format of the IEFSSNxx record for SMS is described in "Defining SMS through the IEFSSNxx member" on page 537.

Note: In general, it is a good idea to make the subsystem name the same as the name of the member of sys1.proclib used to start the subsystem. If the name does not match, you may receive error messages when you start the subsystem.

Restrictions for IEFSSNxx

The following restrictions apply:

- All subsystem definitions in a single IEFSSNxx member must use the same format. A single member cannot contain both positional and keyword definitions.
  - If a member begins with the positional format and switches to the keyword format, processing of the member stops, and the IEFJ002I message is issued. The last subsystem definition processed for the member is the last positional format definition before the switch. The system does not process another definition of either format from the member, but continues processing with the next member, if any.
  - If a member begins with the keyword format and a positional format definition is found, the system issues a syntax error message, and processing continues with the next definition of the keyword format.
  - Once a subsystem name is defined to the system, any attempt to start that subsystem (or any started task with the same name as that subsystem) via a START command which does not explicitly specify SUB=JES2 (or JES3) will result in that subsystem or started task being started under the Master subsystem rather than under the job entry subsystem. Because the only procedure libraries available to the Master subsystem are those specified in the MSTJCLxx's IEFPDSI data set, any procedures being started that are defined in the job entry subsystem's PROC00 data set, but not in the MSTJCLxx's IEFPDSI data set, will be unavailable. Therefore they will not be found; the system will issue message IEFC612I.

Only subsystems that have been defined using the keyword format IEFSSNxx parmlib member, the IEFSSI REQUEST=ADD macro or the SETSSI ADD system command can use the following dynamic SSI services:

- Macros
  - IEFSSI REQUEST=ACTIVATE
  - IEFSSI REQUEST=DEACTIVATE
  - IEFSSI REQUEST=OPTIONS
  - IEFSSI REQUEST=SWAP
  - IEFSSI REQUEST=GET
  - IEFSSI REQUEST=PUT
  - IEFSSVT
- System commands
  - SETSSI ACTIVATE
  - SETSSI DEACTIVATE

You cannot use dynamic SSI services for subsystems defined with the positional form of this member. See Appendix A, “IEFSSNxx (subsystem definitions) - positional parameter form,” on page 791 for more information about the positional form of this member.

Parameter in IEASYSxx (or supplied by the operator)

The SSN parameter in IEASYSxx identifies the IEFSSNxx member that the system is to use to initialize the subsystems, as follows:

```
SSN={aa         }
    {(aa,bb,...)}
```

The two-character identifier, represented by aa (or bb, and so forth) is appended to IEFSSN to identify IEFSSNxx members of parmlib. If the SSN parameter is not specified, the system uses the IEFSSN00 parmlib member.

The order in which the subsystems are specified on the SSN parameter is the order in which they are defined. For example, a specification of SSN=(13,Z5) would cause those subsystems specified in the IEFSSN13 parmlib member to be defined first, followed by those subsystems specified in the IEFSSNZ5 parmlib member. If you specify duplicate subsystem names in IEFSSNxx parmlib members, the system issues message IEFJ003I to the SYSLOG, the INTIDS console, and consoles that monitor routing code 10 messages.

Some exits that use system services may run before other system address spaces are active. You must ensure that any address spaces required by the system services are available prior to invoking the service.

For more information, see the section on handling errors in defining your subsystem in *z/OS MVS Using the Subsystem Interface*.

Syntax rules for IEFSSNxx

The following rules apply to the creation of IEFSSNxx:

- Each SUBSYS statement in IEFSSNxx defines one subsystem to be initialized.
- Use columns 1 through 71. Do not use columns 72-80, because the system ignores these columns.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- A statement must begin with a valid statement type followed by at least one blank or end-of-line. If there is an error in the statement type, the system will flag the error on the preceding statement.
- A statement ends with the beginning of the next valid statement type or end-of-file. Therefore, if there is an error in a statement type, the system will flag the error on the preceding statement.
- A statement can be continued even though there is no explicit continuation character. Be aware that if the system cannot recognize the start of a statement (for example, if a statement starts with an error in the statement type), the system will automatically assume that it is a continuation of the previous statement, and flag an error on the previous statement.
- Operands must be separated by valid delimiters. Valid delimiters are a blank, or column 71. If the operand contains parentheses, then the right parenthesis is accepted as a valid delimiter.
- Multiple occurrences of a delimiter (except for parentheses) are accepted, but treated as one.

IBM-supplied default for IEFSSNxx

If you do not specify the SSN system parameter, the system uses the IEFSSN00 parmlib member. IEFSSN00 specifies JES2 as the primary subsystem.

If you specify a set of IEFSSNxx members that do not identify a primary subsystem, the system issues a message that prompts the operator to specify the primary subsystem.

Statements/parameters for IEFSSNxx

```
SUBSYS     SUBNAME(subname)
           [CONSNAME(consname)]
           [INITRTN(initrtn)
               [INITPARM(initparm)]]
           [PRIMARY({NO|YES})
               [START({YES|NO})]]
BEGINPARALLEL
```

SUBSYS
The statement that defines a subsystem that is to be added to the system. If more than one SUBSYS statement appears for the same subsystem name, the first statement will be the one used to define the subsystem. The duplicate statements will be rejected with a failure message that is sent to the console.

SUBNAME(subname)
The subsystem name. The name can be up to 4 characters long; it must begin with an alphabetic or national character (#, @, or $), and the remaining characters (if any) can be alphanumeric or national.

CONSNAME(consname)
The name of the console to which any messages that the SSI issues as part of initialization processing are to be routed. This name is optional and can be 2-8 characters long. This console name is also passed to the routine named on the INITRTN keyword if it is specified.

The default is to issue messages to the consoles that are receiving the INTIDS routing attribute.

INITRTN(initrtn)
The name of the subsystem initialization routine. This name is optional and can be 1-8 characters long. The first character can be either alphabetic or national (#, @, or $). The remaining characters can be either alphanumeric or national. The routine receives control in supervisor state key 0. It must be the name of a program accessible through LNKLST.

INITPARM(initparm)
Input parameters to be passed to the subsystem initialization routine. The input parameters are optional and are variable in length for a maximum of 60 characters. If blanks, commas, single quotation marks, or parentheses are included in the input parameters, the entire parm field must be enclosed in single quotation marks. If the parm field is enclosed in single quotation marks, a single quotation mark within the field must be specified as two single quotation marks.

The INITPARM keyword can only be specified if the INITRTN keyword is specified.

PRIMARY({NO|YES})
Parameter indicating whether this is the primary subsystem. The primary subsystem is typically a job entry subsystem (either JES2 or JES3).

This parameter is optional. Initialize the primary subsystem before any secondary subsystem(s) except SMS. If you specify PRIMARY on more than one statement, the system issues message IEFJ008I and defines the second subsystem but ignores the PRIMARY specification.

The IEFSSNxx parmlib member is the only place you can define the primary subsystem. It cannot be defined using the dynamic SSI services IEFSSI REQUEST=ADD macro or the SETSSI ADD command.

The default is NO.

START({YES|NO})
Parameter indicating whether an automatic START command should be issued for the primary subsystem.

If the parmlib entry for the primary subsystem is START(NO), the operator must start it later with a START command. If the parmlib entry for the primary subsystem does not specify the START parameter, it defaults to START(YES).

The START parameter cannot be specified for a secondary subsystem. If you specify the PRIMARY(NO) parameter, there is no default for the START parameter.

BEGINPARALLEL
The statement that indicates that the subsystem initialization routines specified in SUBSYS statements that follow the BEGINPARALLEL statement are invoked in parallel to reduce the amount of time it takes for all subsystems to initialize.

For subsystems that do not support parallel processing, you must ensure that the SUBSYS statement appears before the BEGINPARALLEL statement. For example, for the SMS subsystem definition IGDSSIN, be sure to specify the subsystem definitions before you specify the BEGINPARALLEL statement. For SMS, if you specify BEGINPARALLEL before the subsystem definition, the system issues message IEF009E about potential problems. Note that all initialization routines specified before the BEGINPARALLEL keyword are invoked serially, and all routines specified after the BEGINPARALLEL keyword are invoked in parallel. The BEGINPARALLEL keyword must be specified after the SMS entry.

For the z/OS Communications Server TNF and VMCF subsystems, you must specify the subsystem definitions before you specify the BEGINPARALLEL statement if the INITRTN parameter is included on the subsystem definitions. For information about starting the TNF and VMCF subsystems, see *z/OS V2R1.0 Communications Server: IP Configuration Guide*

If you do not specify BEGINPARALLEL, the subsystem initialization routines run serially, and you do not obtain any performance benefits of parallel processing. See the configuration or installation documentation of the subsystem for information about whether the subsystem initialization routines can support running in parallel.

You only need to specify one BEGINPARALLEL statement. If you specify more than one statement, the system issues a message indicating that it is using the first BEGINPARALLEL statement it finds and ignores any other statements. This is also true if you specify multiple concatenated IEFSSxx members with BEGINPARALLEL statements across a sysplex.

You can include any number of records in the IEFSSNxx parmlib member.

Examples of IEFSSNxx member

This section contains several examples of how to code the IEFSSNxx member.

- This example defines subsystem 'JES2' as a primary subsystem and the START command to be issued by the system. No initialization routine is required because subsystem JES2 builds the SSVT when the START command is issued.

  ```
  SUBSYS SUBNAME(JES2) PRIMARY(YES)
  ```

- This next example defines subsystem 'ABC'. Call its initialization routine, which will build the SSVT.

  ```
  SUBSYS SUBNAME(ABC) INITRTN(INITPGM)
  ```

- This example specifies that the initialization routines for RACF, and IRLM run in parallel:

  ```
  SUBSYS SUBNAME(SMS)
    INITRTN(IGDSSIIN)
    INITPARM(ID=ZX)
  SUBSYS SUBNAME(JES2) /* JES2 AS PRIMARY SUBSYSTEM */
    PRIMARY(YES)  START(YES)
  BEGINPARALLEL
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00) INITPARM('#,M')
  SUBSYS SUBNAME(IRLM)
  ```

Chapter 57. IFAPRDxx (product enablement policy)

Use the IFAPRDxx parmlib member to define the enablement policy for products or product features that support product enablement. The policy lists the products and features, as well as the system environment in which they are enabled to run.

With z/OS, IBM supplies a tailored IFAPRD00 member of SYS1.PARMLIB. This tailored member enables the product and any optional features ordered with the product. Your installation can, however, order additional optional features from IBM at a later time, then enable these features after contacting IBM, subject to the product's license terms and conditions. For information about how to define a specific IBM product, see *z/OS MVS Product Management*.

To enable an optional product or feature, you add it to the policy; that is, you add the product to an IFAPRDxx member, then activate the member. Adding a product to the policy is the most common task related to IFAPRDxx; you can, however, disable a product or remove it from the policy. Note that adding a product or feature might require changes to other SYS1.PARMLIB members and an IPL before the product or feature can run.

The system builds the enablement policy from the PRODUCT statements and WHEN statements in the active IFAPRDxx member(s). Each WHEN statement defines a system environment. PRODUCT statements identify products and product features that are enabled or disabled when running in the system environment defined on the preceding WHEN statement.

The system checks the policy when a product, such as an optional z/OS feature, calls the Register service during its initialization. If a product in the policy is not defined or not found, the type of register request determines whether the product is treated as enabled or disabled. For information about the Register service, see *z/OS MVS Programming: Product Registration*. For information about how to set up your system to report on registered products, including those that support product enablement, see *z/OS MVS Product Management*.

When the system checks the policy for a match with a product that is registering, all comparisons allow wildcard characters (* and ?). You can thus define WHEN statements that match multiple system environments and PRODUCT statements that match multiple products.

To determine enablement, the system matches the product that is registering against the statements in the enablement policy. It is possible, because of wildcard characters (? and *) in the policy statements, that multiple policy statements might match the given input product. In that case, MVS uses the "best" match to determine whether or not the product is enabled, using the following rules:

1. An exact match is better than a wildcard match. There is no differentiation between two wildcard matches.
2. The parameters are processed in the following order: Prodowner, ProdID, Prodname, Featurename, Prodvers, Prodrel, and Prodmod. An exact match on a parameter earlier in the list (such as Prodowner) is better than a match on a parameter later in the list (such as Prodname).
3. If, after applying the first two rules, more than one match remains, MVS uses the first match of those that remain.

Before creating the member

The contents of this member are controlled by the z/OS product terms and conditions for those PRODUCT statements that contain both IBM CORP as product owner and z/OS as product name. Before making changes to the PRODUCT statements supplied by IBM in the tailored IFAPRD00 member, see *z/OS MVS Product Management*.

To create or change an IFAPRDxx member, you need to know how to define a system environment and how to define a product:

- The system environment can include the system name, sysplex name, LPAR name, hardware name, and VM user ID, if the system is running as a VM guest.
- The product definition can include the product owner (such as "IBM CORP"), product name, name of a product feature, product identifier, and version, release, and modification level for the product. The definition must include the state, which indicates whether the product is enabled to run on the system, disabled (not allowed) to run on the system, or not defined.

  If the active member includes more than one definition for a product, the system uses the last definition it encounters.

Usage considerations

Note the following when using the IFAPRDxx member:

- You can use the SET PROD operator command to modify the enablement policy dynamically by specifying which IFAPRDxx member(s) the system is to use. Statements in the member(s) modify, not replace, an existing policy.

  The change to the policy takes place immediately but does not affect any product instances that are already running.
- The system does not automatically list the IFAPRDxx parameters at IPL or when the operator issues SET PROD, but the operator can issue the DISPLAY PROD,STATE command to display the active enablement policy.

For more information, see *z/OS MVS System Commands*.

Parameter in IEASYSxx (or issued by the operator)

```
PROD={aa        }
     {(aa,bb...)}
```

The two alphanumeric characters (aa or bb) are appended to IFAPRD to identify an IFAPRDxx parmlib member. If you do not specify the PROD parameter, there is no active enablement policy; all products that attempt to register are treated as not found.

To change the IFAPRDxx member(s) after IPL, the operator can issue the SET PROD command to specify one or more different active members, Some products check the enablement policy during IPL; a policy change for such a product does not take effect until the next IPL.

Syntax rules for IFAPRDxx

The following syntax rules apply to IFAPRDxx:

- Use columns 1-71. Do not use columns 72-80; the system ignores them.
- Blank lines are allowed anywhere in the member.
- Comments can appear in columns 1-71 and must begin with "/*" and end with "*/". You can continue a comment; it does not need to end on the line on which it begins.
- You can use both uppercase and lowercase letters; the system translates lowercase letters to uppercase letters before processing.
- The system recognizes the end of a statement when it encounters either the beginning of the next valid statement or an end-of-file (EOF) indicator.
- Use valid delimiters to separate keyword parameters. A valid delimiter is a comma, a blank, or column 71. The system treats multiple blanks as one. Column 71, when within a string enclosed in quotation marks, is not a valid delimiter.
- Blanks can appear between keyword parameters, between values, and between statements. Blanks cannot appear within a keyword value unless the value is enclosed in single quotation marks.

Syntax format of IFAPRDxx

There are two kinds of statements in IFAPRDxx: WHEN and PRODUCT. The WHEN statement defines a system environment; all PRODUCT statements that follow identify products running on the system that the preceding WHEN statement defines.

The following diagram shows the syntax of the WHEN statement:

```
WHEN ([LPARNAME(l)]
      [SYSNAME(sn)]
      [SYSPLEX(sp)]
      [HWNAME(h)  ]
      [VMUSERID(v)])
```

As you use the diagram, consider:

- The WHEN parameter begins the WHEN statement.
- All parameters are optional.
- You must surround any parameters you specify with parentheses.

The following diagram shows the syntax of the PRODUCT statement:

```
PRODUCT [OWNER(o)]
        [NAME(n)]
        [FEATURENAME(fn)]
        [VERSION(v)]
        [RELEASE(r)]
        [MOD(m)]
        [ID(i)]
        STATE({ENABLED|DISABLED|NOTDEFINED})
```

As you use the diagram, consider:

- The PRODUCT parameter begins the PRODUCT statement.

- The STATE parameter is required; all other parameters are optional.

IBM-supplied default for IFAPRDxx

IBM supplies a tailored IFAPRD00 member, one that reflects the optional features that have been ordered.

Statements/parameters for IFAPRDxx

WHEN

Begins a WHEN statement, which specifies a system environment.

To match the system environment you specify on the WHEN statement with the actual system, MVS compares each specified parameter with the actual system. If all parameters match the actual system, the WHEN statement is considered to be true, and the system processes the PRODUCT statements that follow the WHEN statement. Otherwise, the WHEN statement is considered to be false, and the system ignores the subsequent PRODUCT statements.

When it compares a parameter with an actual system condition, MVS allows wildcard matching. WHEN statement parameters can include wildcard characters (* and ?) that allow a single parameter to match many different actual conditions. For example, SYSNAME(SY?) matches system names like SY1 or SYA but not SYS1. SYSNAME(S*) matches S1 or SYA or SYS1. If you omit a parameter, MVS treats the parameter as if you had specified an asterisk (*), and, because of wildcard matching, it always compares as true; it matches the actual system.

The initial state of the WHEN statement is true; that is, if PRODUCT statements appear before any WHEN statement, MVS processes these PRODUCT statements as if they followed a true WHEN statement.

When the SET PROD command or the PROD system parameter specifies more than one member, however, the WHEN state carries over from one member to the next. To avoid a false WHEN state from a preceding member, begin each member with a WHEN statement, even WHEN(), which is always true.

Syntax Error: When MVS finds a syntax error in a WHEN statement while it is building the policy, MVS ignores the WHEN statement and checks the following PRODUCT statements for syntax errors but does not add them to the policy.

LPARNAME(1)

Specifies the LPAR name, the name of a logical partition defined through HCD or IOCP. To match an actual LPAR name, specify the 1-8 character name, which can include wildcard characters (* and ?). To match any LPAR name or to match when the system is not running in a logical partition, specify LPARNAME(*) or omit the LPARNAME parameter. To match only when the system is not running in a logical partition, specify LPARNAME().

If you specify an LPAR name when the actual system is not running in a logical partition, the comparison uses an actual LPAR name of 8 blanks, and there is no match.

SYSNAME(sn)

Specifies the system name, which must be from 1 to 8 characters in length. For the WHEN statement to be true, the name must match the actual system name. The comparison allows wildcard matching; the name can contain wildcard characters (* and ?).

The default is SYSNAME(*), which matches any system name.

SYSPLEX(sp)
Specifies the sysplex name, which must be from 1 to 8 characters in length. For the WHEN statement to be true, the name must match the actual sysplex name. The comparison allows wildcard matching; the name can contain wildcard characters (* and ?).

The default is SYSPLEX(*), which matches any sysplex name.

HWNAME(h)
Specifies the hardware name, the name (identifier) of a central processor complex (CPC) defined to HCD. To match an actual hardware name, specify the 1-8 character name, which can include wildcard characters (* and ?). To match any hardware name or to match when the hardware name related to the actual system is not known, specify HWNAME(*) or omit the HWNAME parameter. To match only when the hardware name related to the actual system is not known, specify HWNAME().

If you specify a hardware name when the hardware name related to the actual system is not known, the comparison uses an actual hardware name of 8 blanks, and there is no match.

VMUSERID(v)
Specifies the user ID of a z/VM system under which the z/OS image is running as a guest. (For information about running z/OS as a z/VM guest, see *z/VM: Running Guest Operating Systems*.)

To match an actual VM user ID, specify the 1-8 character user ID, which can include wildcard characters (* and ?). To match any VM user ID or to match when the actual system is not running as a VM guest, specify VMUSERID(*) or omit the VMUSERID parameter. To match only when the actual system is not running as a VM guest, specify VMUSERID().

If you specify a VM user ID when the actual system is not running as a VM guest, the comparison uses an actual VM user ID of 8 blanks, and there is no match.

PRODUCT
Begins a PRODUCT statement. A PRODUCT statement defines a product and its enablement state.

When a product attempts to register, the system matches the input product against the policy statements.

When it compares an input request with a policy statement, MVS uses wildcard matching. PRODUCT statement parameters can include wildcard characters (* and ?) that allow a single parameter to match many different input requests. For example, NAME(NM?) matches product names like NM1 or NMA but not NAM1. NAME(N*) matches N1 or NMA or NAM1. If you omit a PRODUCT statement parameter, MVS treats the parameter as if you had specified an asterisk (*), and, because of wildcard matching, it always compares as true; it matches the input request.

It is thus possible that multiple policy statements might match the given input request. In that case, MVS uses the best match to determine whether or not the product is enabled, using the following rules:

1. An exact match is better than a wildcard match. There is no differentiation between two wildcard matches.
2. The parameters are processed in the following order: OWNER, ID, NAME, FEATURENAME, VERSION, RELEASE, and MOD. An exact match on a

parameter earlier in the list (such as OWNER) is better than a match on a parameter later in the list (such as NAME).

3. If, after applying the first two rules, more than one match remains, MVS uses the first match of those that still remain.

Note: When MVS is building the policy, it adds to the policy any PRODUCT statement that it finds after a valid WHEN statement. When there is a syntax error in a WHEN statement, MVS ignores the WHEN statement and checks the subsequent PRODUCT statements for syntax errors but does not add them to the policy.

OWNER(o)
Specifies the name of the product owner, such as 'IBM CORP'. The name must be 1 to 16 characters long.

The characters should be alphabetic, numeric, national (@, #, $), underscore (_), slash (/), hyphen (-), or period (.). The system allows wildcard matching; the name can contain wildcard characters (* and ?).

If the name includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in OWNER('lowercase').

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

The default is OWNER(*), which matches any product owner name.

NAME(n)
Specifies the product name. The name must be 1 to 16 characters long.

The characters should be alphabetic, numeric, national (@, #, $), underscore (_), slash (/), hyphen (-), or period (.). The system allows wildcard matching; the name can contain wildcard characters (* and ?).

If the name includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in NAME('lowercase').

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

The default is NAME(*), which matches any product name.

FEATURENAME(fn)
Specifies the name of a feature of the product. The name must be 1 to 16 characters long.

The characters should be alphabetic, numeric, national (@, #, $), underscore (_), slash (/), hyphen (-), or period (.). The system allows wildcard matching; the name can contain wildcard characters (* and ?).

If the name includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in FEATURENAME('lowercase').

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

The default is FEATURENAME(*), which matches any feature name.

FN is an accepted abbreviation of FEATURENAME.

VERSION(v)
Specifies the product version, where *v* must be 1 to 2 characters long.

The characters should be alphabetic or numeric. The system allows wildcard matching; the version can contain wildcard characters (* and ?).

If *v* includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in VERSION('lc').

The system performs all comparisons in upper case.

The default is VERSION(*), which matches any product version.

VER and VERS are accepted abbreviations of VERSION.

RELEASE(r)
Specifies the product release number. *r* must be 1 to 2 characters long.

The characters should be alphabetic or numeric. The system allows wildcard matching; the release number can contain wildcard characters (* and ?).

If *r* includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in RELEASE('1a').

The system performs all comparisons in upper case.

The default is RELEASE(*), which matches any release number.

REL is an accepted abbreviation of RELEASE.

MOD(m)
Specifies the product modification level. *m* must be 1 to 2 characters long.

The characters should be alphabetic or numeric. The system allows wildcard matching; the modification level can contain wildcard characters (* and ?).

If *m* includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in MOD('c2').

The system performs all comparisons in upper case.

The default is MOD(*), which matches any modification level.

ID(i)
Specifies the product identifier. The identifier must be 1 to 8 characters long.

The characters should be alphabetic, numeric, national (@, #, $), underscore (_), slash (/), hyphen (-), or period (.). The system allows wildcard matching; the identifier can contain wildcard characters (* and ?).

If the identifier includes lowercase alphabetic characters, embedded blanks, or other characters, enclose the parameter in single quotation marks, as in ID('lc34').

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

The default is ID(*), which matches any product identifier.

STATE({ENABLED|DISABLED|NOTDEFINED})
The state of the product. If you specify ENABLED or EN, the registering product can continue to run. If you specify DISABLED, DI, or DIS, the registering product is not to continue. When you specify NOTDEFINED or ND, the system removes from the policy any existing entry for the product.

Examples

Example 1: To enable the GDDM REXX feature of z/OS, first see *z/OS MVS Product Management* for information about enabling IBM products. Then, use the following PRODUCT statement:

```
PRODUCT OWNER('IBM CORP')
        NAME(z/OS)
        ID(5645-001)
        FEATURENAME(GDDM-REXX)
        STATE(ENABLED)
```

For a z/OS feature, do not specify VERSION, RELEASE, or MOD; either omit the parameter or specify an asterisk (*).

Example 2: To indicate that the subsequent PRODUCT statements apply only when the products are running on system S in sysplex SP, specify:

```
WHEN ( SYSNAME(S) SYSPLEX(SP) )
```

Example 3: To indicate that the state of product XXXX owned by YYY INC is to be disabled, specify:

```
PRODUCT OWNER('YYY INC') NAME(XXXX) STATE(DISABLED)
```

Specifying `OWNER(YYY_INC)` would have the same result.

Example 4: To remove product XXXX owned by YYY from the enablement policy, specify:

```
PRODUCT OWNER(YYY) NAME(XXXX) STATE(NOTDEFINED)
```

Example 5: To indicate that all products with names beginning with the letter I are to be enabled, specify:

```
PRODUCT NAME(I*) STATE(ENABLED)
```

Chapter 58. IFGPSEDI (enhanced data integrity)

Enhanced data integrity can prevent users from concurrently accessing a shared sequential data set on DASD for output or update processing, thus avoiding any resulting loss of data. You can activate enhanced data integrity by creating an IFGPSEDI parmlib member.

When you activate enhanced data integrity, you can request that multiple users no longer have concurrent output or update access to a sequential data set on DASD. Enhanced data integrity applies only to sequential data sets. Authorized applications can exclude their sequential data sets from enhanced data integrity protection. For information about how to exclude sequential data sets from enhanced data integrity, see *z/OS DFSMS Using Data Sets*.

If you do not need enhanced data integrity, do not create the IFGPSEDI member of SYS1.PARMLIB.

Activating Enhanced Data Integrity - at IPL and afterward — After you have created IFGPSEDI, you can activate enhanced data integrity in either of the following ways:

- At IPL, by specifying MODE(ENFORCE) or MODE(WARN) in the IFGPSEDI parmlib member
- After IPL, through the S IFGEDI command

Syntax rules for IFGPSEDI

The following syntax rules apply to IFGPSEDI

- If MODE is specified it must be the first keyword in the first record and start in column 1
- Multiple data set names can be included in a single 80 byte record, but the last data set name must be complete. It cannot be continued on to the next record.
- Blanks and commas can be used as delimiters between keywords, but blanks are not allowed between a keyword and the opening parenthesis.
- Comments may appear in columns 1-80 and must begin with "/*". After finding an opening comment delimiter, the rest of the line will be ignored. Therefore, comments may appear as the only entry in a line, or following keyword statement, but comments cannot span more than one line. Multiple line comments can be included as a series of one-line comments, each starting with the opening comment delimiter.
- Uppercase and lowercase letters can be used.
- Multiple lines are allowed, but every statement specification must be complete on a single line.
- It is not required to have a comma at the end of a statement in order to continue to the next line.
- Use columns 1-71. Do not use columns 72-80 for data; these columns are ignored.

Syntax examples

The following is an example of a complete IFGPSEDI member:

```
MODE(ENFORCE),DSN(AONE.ATWO),DSN(HIQUAL.*.LOWQUAL),DSN(DEFG.*),DSN(THIS.FILE.NAME)
```

Syntax format of IFGPSEDI

```
MODE{(WARN|ENFORCE|DISABLE)}[,DSN(datasetname)][,DSN(datasetname)]...
```

IBM-supplied default for IFGPSEDI

None.

Statement/parameters for IFGPSEDI

MODE{(WARN|ENFORCE|DISABLE)}

Use the MODE parameter to specify how you want the system to handle enhanced data integrity violations.

Specify MODE(WARN) to indicate that you want the system to issue a warning message IEC984I or IEC985I if a violation occurs. MODE(WARN) will allow processing of the data set to continue if a violation occurs.

Specify MODE(ENFORCE) to indicate that you want the system to issue an ABEND 213 with reason code FD if an attempt to open a sequential data set that is already open for output occurs.

Specify MODE(DISABLE) to end enhanced data integrity support without having to re-IPL. You can modify IFGPSEDI to specify MODE(DISABLE) and issue the command S IFGEDI.

DSN(datasetname)

Specifies the name of a data set that should be excluded from enhanced data integrity support. The data set name can either be a discrete name (meaning that the entire data set name is specified) or a partially qualified name. To specify a partially qualified name, use an asterisk, "*", or a percent sign, "%" anywhere within the data set name.

Specifies the name of a data set that should be excluded from enhanced data integrity support. The data set name can either be a discrete name (meaning that the entire data set name is specified) or a partially qualified name. To specify a partially qualified name, use an asterisk, "*", or a percent sign, "%" anywhere within the data set name. If two or more consecutive asterisks are specified, they must be immediately preceded by a period or left parenthesis, ex.DSN(**).

Use a percent sign as a generic character for a single position. For example TEST% would include only names with exactly five characters, such as TESTA, TEST9 or TESTZ.

Use an asterisk as a generic character for zero or more characters. For example TEST* would include names with only one qualifier such as TEST, TESTX and TEST1000. Use a double asterisk as a generic character for any number of qualifiers. For example, SYS1.** would include a data set name with any number of qualifiers as long as the first qualifier is SYS1 (for example, data sets SYS1.TEST, SYS1.LIST.XP and SYS1.MEMO.EMPL.A would be included).

You can specify any number of data set name parameters. However, a high number of exclusions may affect performance, so you should keep the number to a minimum. Note that by including data set names in the IFGPSEDI exclude list you will exclude all data sets with that same name. There can be multiple data sets on different volumes with the same name that are not SMS managed. If the name you specify matches a name in the exclude list, then all data sets by that name will bypass EDI processing.

Default: None

Chapter 59. IGDSMSxx (storage management subsystem definition)

IGDSMSxx contains the parameters that initialize the Storage Management Subsystem (SMS) and specify the names of the active control data set (ACDS) and the communications data set (COMMDS).

Parameter in IEASYSxx

The SMS parameter in IEASYSxx identifies the IGDSMSxx member from which the storage management subsystem (SMS) will obtain its options when the system is initialized for partitioned data set extended (PDSE) support, as follows:

SMS {*xx***}**
The two alphanumeric characters, *xx*, are appended to IGDSMS to name the member. The nucleus initialization program (NIP) saves the name until SMS is initialized. If initialization of the PDSE fails, the IGDSMS*xx* PARMLIB member is selected from the IEFSSN*xx* PARMLIB member, using the ID=*xx* keyword.

Defining SMS through the IEFSSNxx member

You can start SMS only after you define it to MVS as a valid subsystem. Do this by adding a record for the SMS subsystem to parmlib member IEFSSNxx. IEFSSNxx defines how MVS is to initialize the SMS address space.

Use keyword parameters to code the IEFSSN*xx* member.

Figure 20 shows the syntax of the keywords that you can use to define SMS in IEFSSNxx.

```
SUBSYS SUBNAME(SMS)
[INITRTN(IGDSSIIN)[INITPARM('ID=yy,PROMPT=NO     ')]]
                 [                      YES     ]]
                 [                      DISPLAY ]]
```

Figure 20. Keyword format of the SMS record in IEFSSNxx

Figure 21 shows the positional format of the SMS definition in IEFSSNxx (for users of positional parameters).

```
 SMS[,[IGDSSIIN][,'[ID=yy][,PROMPT={YES    }']]]
                                   {DISPLAY}
                                   {NO     }
```

Figure 21. Positional format of the SMS record in IEFSSNxx

This record differs somewhat from the one discussed in Appendix A, "IEFSSNxx (subsystem definitions) - positional parameter form," on page 791; do not use the optional parameters PRIMARY or NOSTART.

Tip: Place the SMS record before the primary subsystem (JES2 or JES3) record in IEFSSN*xx* to start SMS before starting the primary subsystem.

The fields within the SMS record are as follows:

SMS Identifies the subsystem as SMS.

IGDSSIIN
Identifies the SMS subsystem initialization routine. If you include this field, SMS is automatically started at IPL. If you omit this field, SMS as defined to MVS as a valid subsystem, but is not automatically started at IPL.

ID=*yy* Specifies the two-character suffix of the IGDSMSxx member to be used to start SMS in either of the following special cases:

- The SMS parameter of IEASYSxx does not specify a valid IGDSMSxx member and default member IGDSMS00 does not exist.
- A system error prevents the initialization of system functions that manage PDSEs (for example, the PDSE address space fails to start).

To avoid confusion, specify the same value on the SMS keyword in IEASYSxx and on the ID keyword in IEFSSNxx.

If you specify both ID and PROMPT, enclose them in one pair of single quotation marks and separate them with a comma.

Default: 00

PROMPT={YES | DISPLAY | NO}
Specifies the amount of control the operator is to have over the SMS initialization parameters. You can specify one of the following values for PROMPT.

DISPLAY
The system displays only the parameters of the IGDSMSxx member. The operator cannot change these parameters.

NO The system does not display the parameters or allow the operator to change them.

YES The system prompts the operator (through a write-to-operator-with-reply (WTOR) message) to change the parameters specified in the IGDSMSxx member. The system displays the current status of parameters before it issues the WTOR. The effect of any change lasts only for the duration of the IPL; the operator's action, if any, does not change the contents of the IGDSMSxx member.

If you specify both ID and PROMPT, enclose them in one pair of single quotation marks and separate them with a comma.

Default: NO

Example of an SMS record in IEFSSNxx

Figure 22 on page 539 shows an example of an SMS record in IEFSSNxx. This record:

- Defines SMS as a valid subsystem of MVS.
- Causes SMS to be started automatically at IPL.
- Identifies IGDSMS60 as the IGDSMSxx member that contains the SMS options.
- Specifies that SMS is to prompt the operator for changes to the SMS parameter options.

```
SUBSYS SUBNAME(SMS) INITRTN(IGDSSIIN)
       INITPARM('ID=60,PROMPT=YES')
```

Figure 22. Example: SMS record in IEFSSNxx

For more information about defining subsystems through IEFSSNxx, see Appendix A, "IEFSSNxx (subsystem definitions) - positional parameter form," on page 791.

Starting SMS - at IPL and afterward

After you have created IGDSMSxx and defined control data sets for SMS to use, you can start SMS in either of the following ways:

- At IPL, by having specified IGDSSIIN in the SMS record in IEFSSNxx.
- After IPL, through the SET SMS=*xx* (or T SMS=*xx*) command, where *xx* identifies the particular IGDSMSxx member that contains the SMS initialization parameters.

For more information about using the MVS SET SMS command, see *z/OS MVS System Commands*.

Specifying SMS parameters through SETSMS and SET SMS

After you have activated an SMS configuration, you can use the SET SMS or SETSMS operator commands to specify SMS parameters, as follows:

- Use the SET SMS command to start SMS if it is not started at IPL or to change SMS parameters if SMS is already running. The SMS parameters are taken from the IGDSMSxx member in SYS1.PARMLIB.
- Use the SETSMS command when SMS is already running to change a single SMS parameter. You enter the parameter to be changed on the command line.

Note: Some parameters in the IGDSMSxx parmlib member are read at IPL time, but do not have any effect unless it is the first time the parameters are specified for a new configuration. As indicated in the section "Changing IGDSMSxx Parameters to Support the Coupling Facility" in*z/OS DFSMSdfp Storage Administration*, you must issue the SETSMS command for any changes to take effect. In particular, the values used by VSAM RLS for the following keywords are remembered from last initialization, but not overridden by the IGDSMSxx specifications at IPL time unless the SETSMS command is issued.

- CF_TIME
- DEADLOCK_DETECTION
- RLS_MAXCFFEATURELEVEL
- RLS_MAX_POOL_SIZE
- RLSABOVETHEBARMAXPOOLSIZE
- RLSFIXEDPOOLSIZE
- SMF_TIME

For more information about using SETSMS and SET SMS, see *z/OS MVS System Commands*.

Syntax rules for IGDSMSxx

You can separate the keywords in IGDSMSxx with blanks or commas. You do not need to specify continuation characters for records that span multiple lines.

Comments may appear in columns 1-71. They must begin with /* and end with */. Comments are allowed between parameters.

Syntax format of IGDSMSxx

The following diagram shows the syntax format of the keywords that you code in IGDSMSxx.

```
SMS    ACDS(dsname) COMMDS(dsname)
       [ACSDEFAULTS({YES|NO})]
       [AKP(nnn[,nnn[...,nnn]])]
       [ASID({asid|*[(TRACE|T|VOLSELMSG|V)[,asid|*(TRACE|T|VOLSELMSG|V)]]})]
       [BLOCKTOKENSIZE({REQUIRE|NOREQUIRE})]
       [BREAKPOINTVALUE(0-65520)]
       [CACHETIME({nnn|3600})]
       [CA_RECLAIM(NONE|{DATACLAS|DATACLASS})]
       [CF_TIME({nnnnn|3600})]
       [CICSVR_INIT({YES|NO})]
       [CICSVR_BACKOUT_CONTROL({backout control string})]
       [CICSVR_DSNAME_PREFIX({DWW.|dsname prefix})]
       [CICSVR_GENERAL_CONTROL({general control string})]
       [CICSVR_GRPNAME_SUFFIX({grpname suffix})]
       [CICSVR_RCDS_PREFIX({rcds prefix})]
       [CICSVR_UNDOLOG_CONTROL({undolog control string})]
       [CICSVR_UNDOLOG_PREFIX({undolog prefix})]
       [CICSVR_ZZVALUE_PARM({zzvalue parm})]
       [COMPRESS({TAILORED|GENERIC|ZEDC_R|ZEDC_P})]
       [DB2SSID(ssid)]
       [DEADLOCK_DETECTION({iiii|15,kkkk|4})]
       [DESELECT({event[,event][,...]|ALL})]
       [DINTERVAL({nnn|150})]
       [DSNAME({dsname|*})]
       [DSNTYPE({LIBRARY|PDS|HFS})]
       [DSSTIMEOUT({nnnn|0})]
       [FAST_VOLSEL(ON|OFF)]
       [GDS_RECLAIM(YES|NO)]
       [HONOR_DSNTYPE_PDSE (YES|NO)]
       [INTERVAL({nnn|15})]
       [JOBNAME({jobname|*
       [(TRACE|T|VOLSELMSG|V)[,jobname|*(TRACE|T|VOLSELMSG|V)]]})]
       [LOG_OF_LOGS(logstream)]
       [MAXGENS_LIMIT(maximum-generations)]
       [MAXLOCKS({max|0},{incr|0})]
       [OAMPROC(procname)]
       [OAMTASK(taskid)]
       [OVRD_EXPDT({YES|NO})]
       [PDSE_BMFTIME|BMFTIME(nnn)]
       [PDSE_BUFFER_BEYOND_CLOSE (YES|NO)]
       [PDSE_DIRECTORY_STORAGE (nnn|2G)]
       [PDSE_HSP_SIZE|HSP_SIZE(nnn)]
       [PDSE_LRUCYCLES|LRUCYCLES(nnn)]
       [PDSE_LRUTIME|LRUTIME(nnn)]
       [PDSE_MONITOR({YES|DISPLAY|DUMPNEXT|NO}[,interval[,duration]])]
       [PDSE_RESTARTABLE_AS(YES|NO)]
       [PDSE_VERSION({1|2})]
       [PDSE1_BMFTIME(nnn)]
       [PDSE1_BUFFER_BEYOND_CLOSE (YES|NO)]
       [PDSE1_DIRECTORY_STORAGE (nnn|2G)]
       [PDSE1_HSP_SIZE(nnn|0)]
       [PDSE1_LRUCYCLES(nnn|15)]
       [PDSE1_LRUTIME(nnn|60)]
       [PDSE1_MONITOR({YES|DISPLAY|DUMPNEXT|NO}[,interval[,duration]])]
       [PDSESHARING({NORMAL|EXTENDED})]
       [PDSE_SYSEVENT_DONTSWAP({NO|YES})]
       [PS_EXT_VERSION({1|2})]
       [QTIMEOUT({nnn|300})]
       [REVERIFY({YES|NO})]
       [RLSINIT({NO|YES})]
       [RLS_MAX_POOL_SIZE({nnnn|100})]
       [RLS_MAXCFFEATURELEVEL({Z|A})]
       [RLSTMOUT({nnn|0})]
```

```
[RLSABOVETHEBARMAXPOOLSIZE({(ALL,size)|(sysname1,size1
        [;sysname2,size2[...;sysname32,size32]])})]
[RLSFIXEDPOOLSIZE({(ALL,size)|(sysname1,size1
        [;sysname2,size2[...;sysname32,size32]])})]
[SAM_USE_HPF(YES|NO)]
[SELECT({event[,event][...]|ALL})]
[SIZE(nnn{K|M})]

[SMF_TIME(YES|NO)]
[STEPNAME({stepname|*})]
[SUPPRESS_DRMSGS (YES|NO)]
[SUPPRESS_SMSMSG (YES|NO,IGD17054I,IGD17227I,IGD17395I)]
[SYSNAME(sysname1[,sysname2[...,sysname32]])] [SYSTEMS({32|8})]
[TRACE({OFF|ON})]
[TRACEEXIT(user_trace_exit)]
[TVSNAME(nnn1[,nnn2[...,nnn32]])]
[TV_START_TYPE({WARM|COLD}[,{WARM|COLD}[...,{WARM|COLD}]])]
[TYPE({ALL|ERROR[(TRACE|T|VOLSELMSG|V)[,ALL|ERROR(TRACE|T|VOLSELMSG|V)]]})]
[USEEAV({YES|NO})]
[USE_RESOWNER({YES|NO})]
[VOLSELMSG({ON|OFF|0|nnnnn|ALL})]
```

IBM-supplied default for IGDSMSxx

If you do not define an IGDSMSxx member, the system will attempt to use member IGDSMS00 by default. IBM does not supply this member; your installation must create it.

Required keywords for IGDSMSxx

You must code the following keywords; there are no defaults.

SMS
Specifies that this record is for SMS.

ACDS(dsname)
Specifies the name of the active control data set (ACDS). If you omit *dsname*, the system will prompt the operator for a value. For information about the ACDS, see *z/OS DFSMSdfp Storage Administration*.

COMMDS(dsname)
Specifies the name of the communications data set (COMMDS). If you omit *dsname*, the system will prompt the operator for a value. For information about the COMMDS, see *z/OS DFSMSdfp Storage Administration*.

Optional keywords for IGDSMSxx

The following keywords are optional; defaults, if any, are noted.

ACSDEFAULTS(YES | NO)
Specifies whether SMS initializes the following automatic class selection (ACS) routine variables from an additional call to RACF:

- &APPLIC
- &DEF_DATACLAS
- &DEF_MGMTCLAS
- &DEF_STORCLAS

Specify **YES** to request that RACF or a functional equivalent give SMS the values. Because SMS must set these variables every time a data set is created,

specifying **NO** reduces the overhead of using RACF. If you specify **NO**, these variables will have no values associated with them.

The ACSDEFAULTS keyword is not applicable for OAM.

Default: NO

AKP({nnn[,nnn[...,nnn]] |1000})
Specifies the activity keypoint trigger value, which is the number of logging operations between keypoints. Up to 32 activity keypoint values can be specified. AKP values must be specified in the same order as DFSMStvs instance names. Specify a value from 200 to 65535.

You can allow some values to default while specifying others. For example, in `AKP(800,,3000)`, the value for the DFSMStvs instance in the second position defaults to 1000, and sets the values for the first and third DFSMStvs instances to 800 and 3000 respectively.

If AKP is specified with only the TVSNAME parameter without an identifier, AKP applies to the system on which the PARMLIB member is being read. TVSNAME must be specified with the AKP parameter.

Activity keypointing is the process of accounting for all the log records in the undo and shunt logs that are involved in active units of recovery. Activity keypointing enables DFSMStvs to delete records that are no longer involved in active units of recovery. It also enables DFSMStvs to update the undo log to optimize its restart performance related to reading the log stream.

Default: 1000

ASID(asid|*[(TRACE|T|VOLSELMSG|V)[,asid|*(TRACE|T|VOLSELMSG|V)]])
Specifies whether SMS is to limit tracing (TRACE(ON)) or issue volume selection messages (VOLSELMSG(ON)) to a specific address space (*asid*) or all address spaces *.

asid|*
Limit tracing and volume selection analysis messages for a certain address space, or all address spaces (*). You can enter up to 4 digits for the ASID keyword. If you leave off the leading zeros, they are inserted.

[(TRACE|T|VOLSELMSG|V)[,asid|*(TRACE|T|VOLSELMSG|V)]]
These are optional sub-parameters. TRACE|T or VOLSELMSG|V associated with the first sub-parameter specifies whether the required value, asid or *, specified in the first sub-parameter applies to TRACE or VOLSELMSG facility. The second sub-parameter is optional and can be used to specify another value and facility after the first sub-parameter is specified. When none of these optional sub-parameters are specified, the value specified in the first sub-parameter applies to both TRACE and VOLSELMSG. For example, if you want to have ASID 0010 for the SMS TRACE facility, and ASID 0020 for the VOLSELMSG facility, you could code: ASID(10(TRACE),20(VOLSELMSG)).

Default: *

BLOCKTOKENSIZE(REQUIRE|NOREQUIRE)
Requires every open for a large format data set to use BLOCKTOKENSIZE=LARGE on the DCBE macro, indicating that the application has made needed changes to support the data sets. The exception to this is if the data set contains no more than 65535 tracks on each volume and is open for input using EXCP, BSAM or QSAM, or for update using BSAM or QSAM.

In z/OS V1R8, IBM changed the default from BLOCKTOKENSIZE(REQUIRE) to (NOREQUIRE). If you want BLOCKTOKENSIZE(REQUIRE), then you must explicitly specify it.

Default: NOREQUIRE

NOREQUIRE
Allows applications to access large format data sets under more conditions without having to specify BLOCKTOKENSIZE=LARGE on the DCBE macro.

When NOREQUIRE is in effect, programs that use the following access methods can open large format data sets without specifying BLOCKTOKENSIZE=LARGE on the DCBE macro:

- QSAM or BSAM without the NOTE or POINT macros.
- BSAM with the NOTE or POINT macros (MACRF=xP) only if the data set has 65535 or fewer tracks on the volume and the OPEN option is INPUT or UPDAT.
- EXCP (MACRF=E) and only when the data set has 65535 or fewer tracks on the volume and the OPEN option is INPUT.

BREAKPOINTVALUE (0-65520)
Specifies the disk space request (primary or secondary), expressed in the number of cylinders (0-65520), where the system should prefer the cylinder-managed space (CMS) on an extended address volume (EAV).

BREAKPOINTVALUE is applicable only to data sets that are eligible for extended addressing space (EAS). Data sets that are not EAS-eligible must reside in the track-managed region of a volume.

When a disk space request is equal to or greater than the value of BREAKPOINTVALUE , the system uses the cylinder-managed space for that extent. If not enough CMS is available, the system uses track-managed space (TMS) or both CMS and TMS. If the requested disk space is less than the BREAKPOINTVALUE, the system uses TMS. If not enough TMS is available, the system uses CMS or both of TMS and CMS.

BREAKPOINTVALUE is an optional field. If it is not specified at the storage group level, the system uses the default value that is specified in the IGDSMSxx member of SYS1.PARMLIB. If BREAKPOINTVALUE is not specified, a value of 10 cylinders is used instead.

CACHETIME({nnnnn}) {3600 }
Specifies the number of seconds between recording SMF records for device cache use. The CACHETIME parameter applies only to the volumes behind an IBM 3990 Storage Control with cache unit. You can specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds), and the default is 3600 (one hour). For information about the control unit cache summary recorded in SMF record type 42, see *z/OS MVS System Management Facilities (SMF)* .

Default: 3600

CA_RECLAIM(NONE|{DATACLAS|DATACLASS})
Specifies whether or not to use CA reclaim for KSDSs according to the CA Reclaim attribute in their data classes.

- NONE indicates that none of the KSDSs will be using CA reclaim, regardless of the data class specification. The default is NONE. If CA_RECLAIM is not specified in SYS1.PARMLIB, the CA_RECLAIM(NONE) default will be in effect, which means the default is CA reclaim disabled.
- DATACLAS|DATACLASS indicates that the SMS-managed KSDSs and non-SMS-managed KSDSs will go by the data class specification of CA

Reclaim=Y|N at define time or subsequent ALTER setting. Only the resultant catalog setting of CA Reclaim=Y will do CA reclaim; CA Reclaim=N will not do CA reclaim. It is the way to specify some KSDSs not to use CA reclaim.

Default: NONE

CF_TIME({nnnnn}) {3600}
Specifies the interval (in seconds) for recording SMF record 42 (subtypes 15, 16, 17, 18) for the SMSVSAM address space's use of the coupling facility.

If you record these subtypes, you can use CF_TIME to synchronize SMF type 42 data with SMF and RMF data intervals.

Specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds). The default is 3600 (one hour).

The SMF_TIME keyword, if set to *YES*, overrides the CF_TIME keyword.

In a sysplex, the first system that is initialized with an IGDSMSxx member having a valid CF_TIME specification determines the CF_TIME value for the other systems in the sysplex. You can change this value through the SETSMS or SET SMS commands.

Default: 3600

CICSVR_INIT(YES|NO)
This parameter indicates whether the CICSVR server address space should be started during an IPL:

- YES indicates that the CICSVR server address space should be started at system initialization.
- NO indicates that the CICSVR start should be deferred. If you specify CICSVR_INIT(NO) you will have to issue a SETSMS CICSVR_INIT(YES) command before you can activate the CICSVR server.

Note: Every system in a sysplex must have an IGDSMSxx setup for the CICSVR server address space.

Default: NO

CICSVR_BACKOUT_CONTROL(*cicsvr_backout_control_string*)
Specifies the CICSVR batch backout control string that is used to control CICSVR batch backout. The *cicsvr_backout_control_string* is up to 17 characters long. The valid characters are alphanumeric (A-Z, 0-9) and national ($, @, #) characters, blanks, and comma.

Default: a blank character string

For detailed information referring to CICSVR, see the documentation available at the following URL: `http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp`

CICSVR_DSNAME_PREFIX(DWW|*dsname prefix*)
Defines a prefix for all CICSVR required data sets. This prefix is used for the data sets that must be pre-allocated before you can start the CICSVR server address space (for example, data sets that are allocated to the DD name, DWWMSG).

Note: CICSVR_DSNAME_PREFIX is required if you specify CICSVR_INIT(YES). This defines a prefix for all CICSVR—required data sets excluding RCDS data sets, DWWCON1, DWWCON2, and DWWCON3.

Default: DWW

CICSVR_GENERAL_CONTROL(*cicsvr_general_control_string***)**
Specifies the CICSVR general control string that is used to general control CICSVR. For example, CICSVR general control string and service functions can be used to invoke a CICSVR scavenger or to display the current setting of all CICSVR control strings.

The *cicsvr_general_control_string* is up to 17 characters long. The valid characters are alphanumeric (A-Z, 0–9) and national ($, @, And #) characters, blanks and comma.

Default: a blank string that is interpreted as SUBMIT NONE

For detailed information referring to CICSVR, see the documentation available at the following URL: `http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp`

CICSVR_GRPNAME_SUFFIX(*cicsvr_grpname_suffix***)**
Defines the CICSVR XCF group name suffix that the CICSVR server address space will use to create a unique XCF group name for the sysplex and connect to the XCF group. All the systems that share the same suffix and are active will belong to the same CICSVR XCF group. The *cicsvr_grpname_suffix* is up to four characters long. The valid characters are alphanumeric (A-Z, 0–9) and national ($, @, And #) characters.

Default: PROD

CICSVR_RCDS_PREFIX(*cicsvr_rcds_prefix***)**
Defines a prefix of the CICSVR Recovery Control Data set (RCDS) name that the CICSVR server address space will use to allocate the RCDS to the CICSVR server. The *cicsvr_rcds_prefix* is up to 17 characters long. It follows the syntax rules for data set names. The prefix can contain up to two qualifiers separated by a period.

Default: DWW

CICSVR_UNDOLOG_CONTROL(*cicsvr_undolog_control_string***)**
Specifies the CICSVR UNDO logging control string. The *cicsvr_undolog_control_string* is up to 17 characters long. The valid characters are alphanumeric (A-Z, 0–9) and national ($, @, And #) characters, blanks and comma.

Default: a blank string that is interpreted as ENABLE CONT

For detailed information referring to CICSVR, see the documentation available at the following URL: `http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp`.

CICSVR_UNDOLOG_PREFIX(*cicsvr_undolog_prefix***)**
Defines a CICSVR UNDO log name prefix that the CICSVR server address space will use to determine the log stream name that should be written to by CICSVR UNDO logging. The *cicsvr_undolog_prefix* is up to eight characters long. It follows the syntax rules for data set names.

Default: DWW

CICSVR_ZZVALUE_PARM(*zzvalue_parm_string***)**
Specifies the ZZVALUE parameter string that is used to control CICSVR debugging and diagnostics. The *zzvalue_parm_string* must be up to 17 characters long, padded on the right with blanks if necessary. The valid characters are alphanumeric (A-Z, 0–9), national ($, @, And #) characters, blank, and comma.

Default: a blank characters string

COMPRESS({TAILORED|GENERIC|ZEDC_R|ZEDC_P})
Specifies the type of compression to be used for the data set.

TAILORED
Specifies that the data set is eligible for compression specifically tailored to the data set. A tailored dictionary is built, using the initial data written to the data set, and imbedded into the data set. The dictionary is issued to compress or expand data written to or read from the data set. This type of compression applies only to sequential data sets, not to VSAM KSDSs.

To convert an existing DBB-based compressed data set to use tailored compressions, you must set the COMPRESS parameter to TAILORED and copy the generic DBB-based data set to a new data set that meets compression requirements.

GENERIC
Specifies that the data set be compressed using generic Dictionary Building Block (DBB) compression. The dictionary is derived from a defined set of compression algorithms in data set SYS1.DBBLIB.

ZEDC_R
Specifies that the data set be compressed using zEnterprise data compression (zEDC). No separate dictionary is created, as zEDC compression hides the dictionary in the data stream. A new dictionary starts in each compression unit. The system can decompress the segment as is.

With this option (as opposed to ZEDC_P), the system will fail the allocation request if the zEDC function is not supported by the system, or if the minimum allocation amount requirement is not met.

ZEDC_P
Specifies that the data set be compressed using zEDC compression, as with the ZEDC_R option above. But here, the system will *not* fail the allocation request, but rather create either a tailored compressed data set if the zEDC function is not supported by the system, or create a non-compressed extended format data set if the minimum allocation amount requirement is not met.

Note:

1. Use tailored compression only when all systems in the SMS complex have been converted to DFSMS/MVS 1.4 or later, and when there is no need to revert to a prior release level for local recovery or remote recovery with Aggregate Backup and Recovery Support (ABARS). The date of general availability for DFSMS/MVS 1.4 was in 1997.
2. To convert an existing DBB-based (generic) compressed data set to use tailored compression, or to convert from either generic or tailored to zEDC compression, first set the COMPRESS parameter to TAILORED, ZEDC_R, or ZEDC_P in the IGDSMSxx parmlib member. Use IEBGENER, ICEGENER, REPRO, or any QSAM or BSAM application to copy the data set to a new data set that meets compression requirements.
3. Note that the zEDC feature must be installed on all systems before using zEDC compression. For more information on zEDC compression, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Default: GENERIC

DB2SSID(ssid)
Specifies the name of the DB2 subsystem which is used by Object Access Method (OAM) for object storage. *ssid* can be from one to four characters. There is no default.

If your installation does not use OAM for object storage, do not specify this parameter.

DEADLOCK_DETECTION(nnnn|15,kkkk|4)
This keyword specifies the intervals for local and global deadlock detection.

nnnn specifies the local system's deadlock detection interval (in seconds). Specify *nnnn* as a one to four digit numeric value in the range 1-9999.

kkkk specifies the number of local deadlock cycles that must expire before the sysplex performs global deadlock detection. Specify *kkkk* as a one to four digit numeric value in the range 1-9999.

In a sysplex, the first system that is initialized with an IGDSMSxx member having a valid DEADLOCK_DETECTION specification will determine the DEADLOCK_DETECTION value for the other systems in the sysplex. You can change this value through the SETSMS or SET SMS commands.

Default: 15,4

DESELECT({event[,event][,...]|ALL})
Deletes items from the list of events and services to be traced (if SMS tracing is active). DESELECT has no default. If you specify events that conflict in SELECT and DESELECT, the keyword that appears last has final authority. The events that you can specify on SELECT and DESELECT are:

MODULE
SMS module entry or exit

SMSSJF
SMS/SJF interfaces

SMSSSI
SMS/SSI interfaces

ACSINT
ACS services interfaces

OPCMD
Operator commands

CONFC
Configuration changes

CDSC Control data set changes

CONFS
SMS configuration services

MSG SMS message services

ERR SMS error recovery and recording services

CONFR
Return data from an active configuration

CONFA
Activate a new configuration

ACSPRO
Perform ACS processing

IDAX SMS interpreter/dynamic allocation

DISP SMS disposition processing exit

CATG SMS catalog services

VOLREF
SMS VOLREF services

SCHEDP
SMS scheduling services (prelocated catalog orientation)

SCHEDS
SMS scheduling services (system select)

VTOCL
SMS VTOC/data set services (allocate existing data set)

VTOCD
SMS VTOC/data set services (delete existing data set)

VTOCR
SMS VTOC/data set services (rename existing data set)

VTOCC
SMS VTOC/data set services (allocate new data set)

VTOCA
SMS VTOC/data set services (add a volume to a data set)

RCD SMS recording services or SMS fast VTOC/VVDS access

DCF SMS device control facility

DPN SMS device pool name select subsystem interface

TVR SMS tape volume record update facility

DSTACK
Trace execution of SMS data set stacking

DEBUG
Debug service

ALL All of the above options

DINTERVAL({nnn}) {150}
Specifies the interval (in seconds) that SMS waits between reading device statistics from the 3990-3 control unit (applicable only if the 3990-3 is installed and has at least one SMS-managed volume). Specify a value from 1 to 999 (16 minutes, 39 seconds).

Default: 150

DSNAME(DSNAME|*)
Limits the number of issued volume selection analysis messages activated by VOLSEGMSG(ON) that are issued to the hardcopy and job logs for a certain data set name or for all data set names. For a VSAM data set, this is the cluster's entry name. Specify "*" to issue volume selection analysis messages for all data set names.

Default: *

DSNTYPE(LIBRARY|PDS|HFS)
Specifies the installation default for data sets allocated with directory space but without a data set type specified. If DSNTYPE is PDS, the default is a partitioned data set format; if DSNTYPE is LIBRARY, the default is a PDSE

(partitioned data set extended) format. If DSNTYPE is HFS, the default is a hierarchal file-system format. For more information about data set types, see *z/OS DFSMS Using Data Sets*.

Default: PDS

DSSTIMEOUT({nnn}) (0)
Specifies the number of seconds that the dss component of DFMSMS waits during backup processing for quiesce data set requests to complete. Specify a value from zero to 65536 seconds (which is more than 18 hours). If you specify a value between 1 and 299 seconds, the system uses a value of 300 seconds (which equals 5 minutes).

The value specified in the DSSTIMEOUT parameter value is activated when the first instance of the SMSVSAM address becomes active in the sysplex. All subsequent SMSVSAM instances will use the same value.

You can alter the DSSTIMEOUT value dynamically in the following ways:

- Using the SETSMS DSSTIMEOUT(*nnnn*) command
- By adding or updating the DSSTIMEOUT parameter in IGDSMSxx parmlib member and then activating it with the SET SMS=xx command

Default: 0

FAST_VOLSEL(ON|OFF)
Specifies whether to use the “fast” approach during SMS volume selection.

For striping allocations, fast volume selection is automatically activated regardless of the specification of the FAST_VOLSEL parameter.

If you specify ON, SMS first selects volumes typically until DADSM rejects 100 volumes for insufficient free space. SMS issues message IGD17294I to indicate that 'fast' volume selection has been entered and then excludes volumes that do not have sufficient free space in the volume statistics. This 'fast' approach can inadvertently exclude volumes that have sufficient free space but for which SMS volume statistics indicates that they do not. SMS volume statistics can occur for the following events:

- The VTOC index is broken.
- OEM products bypass CVAF processing.
- In an SMSplex when the SMS synchronization time interval has not yet been driven to update the SMS configuration with the most current space statistics. These statistics are based on updates that can occur on another system in the SMSplex

If you specify OFF, SMS uses the 'normal' approach to select volumes.

Default: OFF

GDS_RECLAIM(YES|NO)
Specifies whether the system will reclaim generation data sets. This is an optional keyword.

If you specify YES, the system will reclaim generation data sets.

If you specify NO, GDS reclaim processing will not be done. This means that if a new generation is created by a job but does not get rolled in, then you must manually:

- delete the generation,
- rename it to a different name, or
- roll it in using Access Method Services

If you do not take one of these manual steps, then on a subsequent attempt to create a new generation, SMS will detect the existence of a duplicate data set and will fail the allocation.

For complete information about how reclaim processing works, see Reclaiming Generation Data Sets in *z/OS DFSMS Using Data Sets*.

Default: YES

HONOR_DSNTYPE_PDSE (YES|NO)
Specifies whether DSNTYPE of LIBRARY or HFS will be honored during the data set creation, regardless of what DSORG is specified and regardless of whether directory blocks have been specified. If YES is specified, then DSNTYPE of LIBRARY or HFS will be honored even if the DSORG is not set to PO and there are no directory blocks specified. If NO is specified, DSNTYPE of LIBRARY or HFS will be honored only when the DSORG is set to PO or directory blocks are specified; otherwise, a physical sequential data set will be created.

Default: NO

INTERVAL({nnn}) {15 }
Specifies the synchronization interval of the system, which is the number of seconds between system checks of the COMMDS for information about SMS configuration changes from other systems in the SMS complex. You can specify a value from 1 to 999 (16 minutes, 39 seconds).

If your installation has multiple systems simultaneously trying to update the SMS configuration and you have your INTERVAL value set too low, you may experience contention. The larger the number of SMSplex members and the larger the configuration, the more contention you may experience. SMS makes several attempts to complete the configuration updates, but after a pre-defined number of attempts have been blocked by contention, the system issues the 6040 configuration services reason code. This is more liked to happen in installations with multiple SMSplex members (8 or more) and large SMS configurations (50,000 or more volumes). To reduce the amount of contention and 6040 reason codes you experience, set your INTERVAL value higher.

Default: 15

JOBNAME(jobname|*[(TRACE|T|VOLSELMSG|V)[,jobname|*(TRACE|T|VOLSELMSG|V)]])
Specifies whether SMS is to limit tracing (TRACE) and volume selection analysis messages (VOLSELMSG(ON)) for a certain job (*jobname*), or permit tracing and volume selection analysis messages on all jobs. This keyword supports objects or tape libraries.

jobname|*
Limit tracing and volume selection analysis messages for a certain job, or all jobs (*). Specify "*" to issue for all jobs.

[(TRACE|T|VOLSELMSG|V)[,jobname|*(TRACE|T|VOLSELMSG|V)]]
These are optional sub-parameters. TRACE|T or VOLSELMSG|V associated with the first sub-parameter specifies whether the required value, jobname or *, specified in the first sub-parameter applies to TRACE or VOLSELMSG facility. The second sub-parameter is optional and can be used to specify another value and facility after the first sub-parameter is specified. When none of these optional sub-parameters are specified, the value specified in the first sub-parameter applies to both TRACE and VOLSELMSG. For example, if you want to have all jobs for the SMS TRACE facility, and a particular job, JOB111, for the VOLSELMSG facility, you could code: JOBNAME(*(TRACE),JOB111(VOLSELMSG)).

Default: *

LOG_OF_LOGS(logstream)
Specifies the log stream that is to be used as the log of logs. This log contains copies of tie-up and file-close records written to forward recovery logs and is used by forward recovery products. If this parameter is not specified, no log of logs is used. This parameter is unique to each system in the sysplex. As each instance of DFSMStvs starts, it uses the log of logs name that is found in its member of SYS1.PARMLIB. If you run test and production systems within the same sysplex, use a separate PARMLIB member for the test system and specify a different log of logs in it.

If you specify the LOG_OF_LOGS parameter, you must also specify the TVSNAME(*nnn*) parameter.

Default: There is no default.

MAXGENS_LIMIT(0-2000000000)
Specifies an upper limit for the MAXGENS parameter on the DD statement in JCL. MAXGENS specifies the number of generations for version 2 PDSEs.

For information about the MAXGENS parameter, see *z/OS MVS JCL Reference*.

Default: The default is 0.

MAXLOCKS({max|0},{incr|0})
Specifies a pair of values in the range of 0 to 999999. The two values are the maximum number of unique lock requests that a single unit of recovery can make and an increment value. Once the maximum number of unique lock requests is reached, warning messages are issued every time the number of unique lock requests over and above the maximum increases by a multiple of the increment. When the maximum number is reached, warning message IGW859I is issued to the system console, and message IGW10074I is issued to the job log. The messages include the name of the job that is holding the locks. This information will help you to determine whether the job should be canceled, in which case the unit of recovery will be backed out, and the locks will remain held until the backout completes. Specifying a value of 0 indicates that warning messages IGW859I and IGW10074I should not be issued.

If you specify the MAXLOCKS parameter, you must also specify the TVSNAME(*nnn*) parameter. This parameter applies across all systems.

Note:

1. Lock requests are considered unique if they lock different records within the base cluster. Repeated requests for the same base cluster records do not result in the count being increased.
2. Warning messages IGW859I and IGW10074I are not issued for units of recovery that are in backout. This is because a unit of recovery that is in backout cannot obtain locks on any additional records.
3. Messages IGW859I and IGW10074I are issued until the unit of recovery reaches commit. Once the unit of recovery reaches commit, no additional messages will be issued.
4. To avoid flooding the system console with messages, messages IGW859I and IGW10074I are issued by an asynchronous timer driven task that wakes up every 10 seconds. This means that the messages will not necessarily reflect the exact values specified for the maximum and the increment, but rather will reflect the values which represent the state of the unit of recovery at the time the task awakens.

5. MAXLOCKS takes into account the number of unique lock requests. It does not count the actual number of locks obtained. The number of locks requested will differ from the number of locks held when alternate indexes are used. If an update modifies alternate keys, a lock is obtained for the base record, for each old alternate key, and for each new alternate key. Therefore, if *n* alternate keys are modified, a single lock request can result in obtaining (2n+1) locks.

Some examples of how this parameter can be specified are:

MAXLOCKS(0,0)
Valid - messages IGW859I and IGW10074I will never be issued.

MAXLOCKS(,)
Valid - this is the equivalent of specifying MAXLOCKS(0,0); messages IGW859I and IGW10074I will never be issued.

MAXLOCKS(5000,0)
Valid - messages IGW859I and IGW10074I will be issued when the asynchronous task wakes up and a unit of recovery has made its 5000th lock request.

MAXLOCKS(0,2000)
Not valid

MAXLOCKS(4500,1000)
Valid - messages IGW859I and IGW10074I will be issued when the asynchronous task wakes up and a unit of recovery has made its 4500th lock request and again every 1000 unique lock requests thereafter.

MAXLOCKS(1000,2300)
Valid - messages IGW859I and IGW10074I will be issued when the asynchronous task wakes up and a unit of recovery has made its 1000th lock request and again every 2300 unique lock requests thereafter.

MAXLOCKS(3200,)
Valid - this is the equivalent of specifying MAXLOCKS(3200,0); messages IGW859I and IGW10074I will be issued when the asynchronous task wakes up and a unit of recovery has made its 3200th lock request.

MAXLOCKS(,2000)
Not valid

Default: The default for both values is 0

OAMPROC(procname)
Specifies the name of the procedure that is to start the OAM address space when SMS is initialized. You must specify this keyword if you want the OAM address space to be started during IPL. The procedure name can be from one to eight characters. There is no default.

OAMTASK(taskid)
Specifies the ID of the task that is to be used to start the OAM address space. OAMTASK is optional; if you specify it without an OAMPROC value, it is ignored. If you omit OAMTASK, the task ID defaults to the procedure name specified in OAMPROC. OAM keywords take effect only if you start SMS at IPL; otherwise the system ignores them. The task ID can be from one to eight characters.

OVRD_EXPDT(YES|NO)
Specifies whether the expiration date or retention period for an SMS-managed DASD data set is to be overridden when a user attempts to delete the data set in one of the following ways:

- DISP parameter on the JCL DD statement.
- Dynamic allocation.
- IEHPROGM utility.
- ISPF/PDF D or DEL line commands.

Specify YES to override an expiration date or retention period for an SMS-managed DASD data set. The data set will be deleted regardless of expiration dates or retention period and the system will not prompt for confirmation. Note that this is a system-level parameter that will effect all jobs running on the system. Exercise caution in using it, so that you do not accidentally delete needed data sets.

When you specify YES, ensure that all appropriate personnel at your installation know that specifying an expiration date or retention period for an SMS-managed DASD data set will not prevent the data set from being deleted.

IBM suggests that you specify YES when:

- Expiration dates or retention periods are not used for data sets, or are overridden by appropriate management classes.
- Data sets allocated to tape are redirected using tape mount and the retention periods and expiration dates for these data sets are not overridden by management classes.

Specify NO to honor the expiration date or retention periods.

Default: NO

PDSE_BMFTIME|BMFTIME(nnn)
With the introduction of a second PDSE address space, two sets of PDSE parameters exist depending on the release of the z/OS system. For z/OS 1.8, PDSE_BMFTIME applies. For z/OS 1.8 and later, BMFTIME applies and is associated with the SMSPDSE address space. For information about parameters for z/OS releases earlier than z/OS 1.8, see the appropriate version and release level of this book.

Specifies the number of seconds that SMS is to wait between recording SMF type 42 subtype 1 records for buffer manager facility (BMF) cache use and SMF type 42 subtype 6 interval records. You can specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds), and the default is 3600 (one hour).

For information about the buffer management statistics recorded in SMF record type 42, see *z/OS MVS System Management Facilities (SMF)*.

Default: 3600

PDSE_BUFFER_BEYOND_CLOSE (YES|NO)
For the SMSPDSE address space, specifies whether to keep cached directory and member data in storage beyond the last close on this system of a PDSE data set. If the NO option is selected or defaulted, the cached directory and member data of a PDSE is purged from the in-memory cache when the last close of the data set occurs. If you specify the YES option, the cached PDSE directory and member data are retained in the in-memory cache beyond the last close of the data set.

Default: NO

PDSE_DIRECTORY_STORAGE(nnn|2G)
Specifies the size in gigabytes or megabytes of 64-bit virtual storage that is used to cache PDSE directory buffers in the SMSPDSE address space. The size values are defined with nnnM for megabytes, or nnnG for gigabytes. For example, to request a 500 megabytes size for the PDSE server address space directory cache, specify PDSE_DIRECTORY_STORAGE(500M).

Default: 2 Gigabytes

Maximum: 16 Gigabytes

Minimum: 64 Megabytes

PDSE_HSP_SIZE|HSP_SIZE(nnn)
With the introduction of a second PDSE address space, two sets of PDSE parameters exist. For z/OS 1.8 and greater, HSP_SIZE is synonymous with PDSE_HSP_SIZE and is associated with the SMSPDSE address space. For information about parameters for z/OS releases earlier than z/OS 1.8, see the appropriate version and release level of this book.Specifies the size of the hiperspace in megabytes that is used for PDSE member caching for SMSPDSE.

You can use the HSP_SIZE parameter to request up to 2047 megabytes for the PDSE hiperspace. You can indicate that the hiperspace is not to be created by setting HSP_SIZE to 0. If the hiperspace is not created, the system will not cache PDSE members.

On systems that are running in z/Architecture mode, by default the PDSE hiperspace is not created and PDSE member caching is disabled.

The system uses the valid values for HSP_SIZE to create the PDSE hiperspace at IPL-time. The HSP_SIZE value remains in effect for the duration of the IPL.

If not enough of the appropriate storage is available to satisfy the HSP_SIZE value, the system uses some portion of the available storage (up to the full amount) for the PDSE hiperspace depending on the amount of caching activity in the system. If the available storage becomes full, the system stops caching PDSE members

Use the HSP_SIZE parameter with care. If you specify an HSP_SIZE value that is too low for normal PDSE hiperspace usage, you can degrade PDSE performance. If you specify a value for HSP_SIZE that is too large, and there is contention for storage on the system, you can degrade performance of other components or applications in the system.

To determine the current HSP_SIZE value of the PDSE hiperspace, use the DISPLAY SMS,OPTIONS command, or review the messages that are written to syslog when SMS is started.

You can examine SMF type 42, subtype 1, records to evaluate the effectiveness of a particular HSP_SIZE value.

PDSE Caching statistics at the data set level are available in SMF type 14 and 15 records. You can also display PDSE Caching details using the DISPLAY SMS,PDSE,HSPSTATS command.

Default: 0 for z/Architecture mode, 256 for non-z/Architecture mode

PDSE_LRUCYCLES|LRUCYCLES(nnn)
LRUCYCLES specifies the maximum number of times (5 to 240 cycles) that the least recently used (LRU) routine passes over inactive buffers before making them available for reuse. While this parameter sets the **maximum** value, the LRU algorithm dynamically changes the **actual** number of times it passes over inactive buffers.

LRUCYCLES is related to LRUTIME. A change to the LRUCYCLES value introduced by this parameter takes effect on the next execution of the LRU routine. Use the default value. In some very high data rate situations, you can tune this value. Monitor the SMF 42 type 1 record to determine the amount of caching activity in the BMF data space. See *z/OS MVS System Management Facilities (SMF)* for information about the buffer management statistics recorded in SMF record type 42.

Note: With the introduction of a second PDSE address space in z/OS 1.6, two sets of PDSE parameters exist. For z/OS 1.6 and greater, LRUCYCLES applies and is associated with the SMSPDSE address space. For information about parameters for z/OS releases earlier than z/OS 1.6, see the appropriate version and release level of this book.

Default: 15 for z/Architecture mode, 240 for non-z/Architecture mode

PDSE_LRUTIME|LRUTIME(nnn)
With the introduction of a second PDSE address space in z/OS 1.6, now two sets of PDSE parameters exist. For information about parameters for z/OS releases earlier than z/OS 1.6, see the appropriate version and release level of this book.

LRUTIME specifies the number of seconds (5 to 60) that the system waits between calls to the LRU (least recently used) routine. The LRU routine releases inactive buffers in the data space that are used to cache PDSE (partitioned data set extended) directory data and in the hiperspace used to cache PDSE member data.

LRUTIME is related to LRUCYCLES. A change to the LRUTIME value takes effect on the next execution of the LRU routine. Use the default value. In some very high data rate situations you can tune this value. Monitor the SMF 42 type 1 record to determine the amount of caching activity in the data space. See *z/OS MVS System Management Facilities (SMF)* for information about the buffer management statistics recorded in SMF record type 42.

Default: 60 for z/Architecture mode, 15 for non-z/Architecture mode

PDSE_MONITOR({YES|DISPLAY|DUMPNEXT|NO}[,interval[,duration]])
Specifies how the processing for the PDSE monitor should be started or modified for the non-restartable PDSE address space, SMSPDSE.

YES
Turns on monitor processing.

DISPLAY
Turns on monitor processing and causes any possible error messages to be displayed to the console.

DUMPNEXT
Turns on monitor processing and causes a dump to be taken for the next error detected. Only one dump is taken.

NO Turns off monitor processing.

If the PDSE parameter is omitted, the monitor is started with default values for 60 seconds for interval and 15 seconds for duration.

interval
Specifies the number of seconds between successive scans of the monitor. If

the PDSE_MONITOR keyword is specified, but interval is omitted, the interval remains unchanged, except at IPL time when the interval is set to 60.

duration
Specifies the number of seconds a possible error condition must exist before it is treated as an error. If the PDSE_MONITOR keyword is specified, but duration is omitted, the duration remains unchanged, except at IPL time when the duration is set to 15.

Default: YES

PDSE_RESTARTABLE_AS(YES|NO)
Specifies whether PDSE initialization during IPL NIP processing brings up a second restartable PDSE address space.

If you specify PDSE_RESTARTABLE_AS(YES), with PDSESHARING(EXTENDED), and EXTENDED sharing is not used, processing does not allow PDSE initialization to create a second restartable PDSE address space.

The PDSE_RESTARTABLE_AS specification is set in the IGDSMSxx member of SYS1.PARMLIB. This value cannot be changed through an operator command.

Default: NO

Note: The PDSE parameters must be specified in the IEASYSxx SMS=xx rather than in IEFSSNxx.

PDSE_VERSION({1|2})
Specifies the version number to be used as a default for data sets that are allocated with a DSNTYPE of LIBRARY.

The PDSE_VERSION specification is set in the IGDSMSxx member of SYS1.PARMLIB. This value cannot be changed through an operator command.

Default: 1

PDSE1_BMFTIME(nnn)
Specifies the number of seconds that SMS is to wait between recording SMF type 42 subtype 1 records for buffer manager facility (BMF) cache use and SMF type 42 subtype 6 interval records. You can specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds). For information about the buffer management statistics recorded in SMF record type 42, see *z/OS MVS System Management Facilities (SMF)* .

Default: 3600 (one hour)

Note: With the introduction of a second PDSE address space in z/OS 1.6, two sets of PDSE parameters now exist. For z/OS 1.6 and later, PDSE1_BMFTIME is associated with the restartable SMSPDSE1 address space. See the PDSE_RESTARTABLE_AS(YES) parameter for information about the SMSPDSE1 address space.

PDSE1_BUFFER_BEYOND_CLOSE (YES|NO)

For the SMSPDSE1 address space, specifies whether to keep cached directory and member data in storage beyond the last close on this system of a PDSE data set. If the NO option is selected or defaulted, the cached directory and member data of a PDSE is purged from the in-memory cache when the last

close of the data set occurs. If you specify the YES option, the cached PDSE directory and member data are retained in the in-memory cache beyond the last close of the data set.

Default: NO

PDSE1_DIRECTORY_STORAGE(nnn|2G)
Specifies the size in gigabytes or megabytes of 64-bit virtual storage that is used to cache PDSE directory buffers in the SMSPDSE1 address space. The size values are defined with nnnM for megabytes, or nnnG for gigabytes. For example, to request a 10 gigabytes size for the PDSE1 server address space directory cache, specify PDSE1_DIRECTORY_STORAGE(10G).

Default: 2 Gigabytes

Maximum: 16 Gigabytes

Minimum: 64 Megabytes

Note: The PDSE1_DIRECTORY_STORAGE amount is not a hard limit. This amount can be exceeded during periods of heavy PDSE processing.

PDSE1_HSP_SIZE(nnn|0)
PDSE1_HSP_SIZE parameter specifies the size of the hiperspace in megabytes that is used for PDSE member caching for SMSPDSE1.

You can use the PDSE1_HSP_SIZE parameter to request up to 2047 megabytes for the PDSE1 hiperspace. Or, you can indicate that the hiperspace is not to be created by setting PDSE1_HSP_SIZE to 0. If the hiperspace is not created, the system will not cache PDSE members.

On systems that are running in z/Architecture mode, by default the PDSE1 hiperspace is not created and PDSE member caching is disabled.

If you specify a valid value for PDSE1_HSP_SIZE, the system uses it to create the PDSE1 hiperspace at IPL-time. The PDSE1_HSP_SIZE value may be changed by a SET SMS=xx command, and then the new value will be used if the SMSPDSE1 address space is restarted.

If not enough of the appropriate storage is available to satisfy the PDSE1_HSP_SIZE value, the system uses some portion of the available storage (up to the full amount) for the PDSE1 hiperspace, depending on the amount of caching activity in the system. The system stops caching PDSE members if the available storage becomes full.

Use the PDSE1_HSP_SIZE parameter with care. If you specify an PDSE1_HSP_SIZE value that is too low for normal PDSE hiperspace usage, you can degrade PDSE performance. If you specify a value for PDSE1_HSP_SIZE that is too large, and there is contention for storage on the system, you can degrade performance of other components or applications in the system.

To determine the current PDSE1_HSP_SIZE value of the PDSE1 hiperspace, use the DISPLAY SMS,OPTIONS command, or review the messages that are written to syslog when SMS is started.

You can examine SMF type 42, subtype 1, records to evaluate the effectiveness of a particular HSP_SIZE value.

PDSE Caching statistics at the data set level are available in SMF type 14 and 15 records. You can also display PDSE Caching details using the DISPLAY SMS,PDSE1,HSPSTATS command.

Note: With the introduction of a second PDSE address space in z/OS 1.6, two sets of PDSE parameters now exist. For z/OS 1.6 and later, PDSE1_HSP_SIZE is associated with the restartable SMSPDSE1 address space. See the PDSE_RESTARTABLE_AS(YES) parameter for information about the SMSPDSE1 address space.

Default: 0

PDSE1_LRUCYCLES(nnn|15)
PDSE1_LRUCYCLES specifies the **maximum** number of times (5 to 240 cycles) that the least recently used (LRU) routine will pass over inactive buffers before making them available for reuse. The LRU algorithm will dynamically change the **actual** number of times it passes over inactive buffers.

PDSE1_LRUCYCLES is related to PDSE_LRUTIME. A change to the PDSE1_LRUCYCLES value takes effect on the next execution of the LRU routine. Use the default value. In some very high data rate situations, you can tune this value. Monitor the SMF 42 type 1 record to determine the amount of caching activity in the BMF data space. See *z/OS MVS System Management Facilities (SMF)* for information about the buffer management statistics recorded in SMF record type 42.

Note: With the introduction of a second PDSE address space in z/OS 1.6, now two sets of PDSE parameters now exist. For z/OS 1.6 and later, PDSE1_LRUCYCLES is associated with the restartable SMSPDSE1 address space. See the PDSE_RESTARTABLE_AS(YES) parameter for information about the SMSPDSE1 address space.

Default: 15 for z/Architecture mode, 240 for non-z/Architecture mode

PDSE1_LRUTIME(nnn|60)
PDSE1_LRUTIME specifies the number of seconds (5 to 60) that the system waits between calls to the LRU (least recently used) routine. The LRU routine releases inactive buffers in the data space that are used to cache PDSE (partitioned data set extended) directory data and in the hiperspace used to cache PDSE member data.

PDSE1_LRUTIME is related to PDSE1_LRUCYCLES. A change to the PDSE1_LRUTIME value takes effect on the next execution of the LRU routine. Use the default value. In some very high data rate situations, you change this value for tuning purposes. Monitor the SMF type 1 record to determine the amount of caching activity in the data space. See *z/OS MVS System Management Facilities (SMF)* for information about the buffer management statistics recorded in the SMF type 42 record.

Note: With the introduction of a second PDSE address space in z/OS 1.6, two sets of PDSE parameters now exist. For z/OS 1.6 and later, PDSE1_LRUTIME is associated with the restartable SMSPDSE1 address space. See the RESTARTABLE_PDSE_AS(YES) parameter for information about the SMSPDSE1 address space.

Default: 60

PDSE1_MONITOR({YES|DISPLAY|DUMPNEXT|NO}[,interval[,duration]])
Specifies how the processing for the PDSE1 monitor is to be started or modified for the restartable PDSE address space, SMSPDSE1.

YES
Turns on monitor processing.

DISPLAY
Turns on monitor processing and causes any possible error messages to be displayed to the console.

DUMPNEXT
Turns on monitor processing and causes a dump to be taken for the next error detected. Only one dump is taken.

NO Turns off monitor processing.

If the PDSE1 parameter is omitted, and the restartable PDSE address space is created, the monitor is started with default values for 60 seconds for interval and 15 seconds for duration.

interval
Specifies the number of seconds between successive scans of the monitor. If the PDSE1_MONITOR keyword is specified, but interval is omitted, the interval remains unchanged, except at IPL time when the interval is set to 60.

duration
Specifies the number of seconds a possible error condition must exist before it is treated as an error. If the PDSE1_MONITOR keyword is specified, but duration is omitted, the duration remains unchanged, except at IPL time when the duration is set to 15.

Default: YES

PDSESHARING(NORMAL|EXTENDED)
Specifies how PDSEs can be shared across systems in a sysplex. NORMAL allows users to share read access to PDSEs across systems in the sysplex. EXTENDED allows users to share read and write access to PDSEs across systems in the sysplex.

All systems that share PDSEs must use the same sharing protocol, either NORMAL or EXTENDED. The first system in the sysplex to IPL will determine which sharing protocol is used. If EXTENDED has been established as the protocol and a system that is not able to run with EXTENDED PDSE sharing joins the sysplex, that system will not be able to use the PDSEs.

For more information about sharing PDSEs across a sysplex, see *z/OS MVS Setting Up a Sysplex*.

Default: NORMAL

PDSE_SYSEVENT_DONTSWAP(NO|YES)
Specifies if the task that enters the SMSPDSE or SMSPDSE1 address spaces should be placed in run DONTSWAP in order to prevent the task from being swapped out while holding internal PDSE latches or locks. Specifying this parameter can delay the system from being swapped out but prevents PDSE processing from being delayed by a swapped address space. In addition, it causes the SYSEVENT ENQHOLD to be issued against a latch holder to insure that the holder completes its processing and releases the latch.

For more information about sharing PDSEs across a sysplex, see *z/OS MVS Setting Up a Sysplex*.

Default: NO

PS_EXT_VERSION(1|2)
This parameter indicates the format in which the system should create sequential extended format data sets. The default is PS_EXT_VERSION(1) to

denote the version 1 format. That means that if the sequential extended format data set is not striped and has multiple volumes, FlashCopy® cannot process it.

DFSMSdss must use conventional I/O to copy this type of data set. When the option is PS_EXT_VERSION(2), any new sequential extended format data sets created will be in version 2 format. This will allow FlashCopy to copy the data set much more efficiently if it has multiple volumes and is not striped. The system saves this version number in the field DFASEFVR in the DFA, which is mapped by the IHADFA macro as described in *z/OS DFSMSdfp Advanced Services*.

Each data set's catalog entry will indicate whether the data set is version 1 or version 2. You can display the data set version via IDCAMS LISTCAT and DCOLLECT.

Default: 1

QTIMEOUT({*nnn*|300})
Specifies the quiesce exit timeout value, in seconds. The quiesce timeout value specifies the amount of time the DFSMStvs quiesce exits allow to elapse before concluding that a quiesce cannot be completed successfully. Only one quiesce timeout value can be specified. The first instance of DFSMStvs that is brought up within the sysplex determines the value. Subsequent SFSMStvs instances use the value established by the first system, regardless of what can be specified in their members of SYS1.PARMLIB. Specify a value between 60 and 3600.

If you specify the QTIMEOUT parameter, you must also specify the TVSNAME(*nnn*) parameter. This parameter applies across all systems.

Default: 300

REVERIFY(YES|NO)
Specifies whether SMS is to check a user's authority to allocate a new data set and use storage or management class at job interpretation time or at both job interpretation time and execution time. If you want SMS to check the authority at both times, code YES; NO directs SMS to check only at job interpretation time.

Default: NO

RLSINIT({NO|YES})
Specifies whether you want the SMSVSAM address space started as part of system initialization or through the V SMS,SMSVSAM,ACTIVE command.

Specify YES if you want the SMSVSAM address space started as part of system initialization or through the command. This value applies only to the system accessed by the parmlib member and is acted upon when SMSVSAM is next started.

Specify NO if you do not want to start SMSVSAM during system initialization or through the command. If you specify NO, you must take the following steps to start SMSVSAM later:

1. Change RLSINIT to YES
2. Issue the SET SMS=xx command to activate the parmlib member you updated with RLSINIT(YES)
3. IPL or issue the V SMS,SMSVSAM,ACTIVE command to start the SMSVSAM address space

Default: NO

RLS_MAX_POOL_SIZE({nnnn|100})
Specifies the maximum size in megabytes of the SMSVSAM local buffer pool. SMSVSAM attempts to not exceed the buffer pool size you specify, although more storage might be temporarily used. Because SMSVSAM manages buffer pool space dynamically, this value does not set a static size for the buffer pool.

Use SMF 42, subtype 19 records to help you determine the maximum size of the SMSVSAM local buffer pool.

You can specify a two to four-digit numeric value, with 10 as the minimum value. If you specify a value less than 10, the field is set to 10. If you specify a value greater than 1500, SMSVSAM assumes there is no maximum limit. IBM suggests that you limit the size of the local buffer pool.

Default: 100

RLS_MAXCFFEATURELEVEL({Z|A})
Specify the method that VSAM RLS should use to determine the size of the data that is placed in the CF cache structure. You can use the RLS_MAXCFFEATURELEVEL keyword to limit the connect level when the sysplex has a mixed level of releases and maintenance. If you do not specify a value, or if you specify Z, then only VSAM RLS data that has a Control Interval (CI) value of 4K or less is placed in the CF cache structure. If you specify A, caching proceeds using the RLSCFCACHE keyword characteristics that are specified in the SMS data class that is defined for the VSAM sphere.

RLS_MAXCFFEATURELEVEL is a sysplex wide value. The first system activated in the sysplex will set the value; all other systems will use the value set by the first system.

Default: Z

Note:

1. If A is specified for the RLS_MAXCFFEATURELEVEL parameter, systems lower than V1R3 will not be able to connect to the CF cache structure.
2. If a lower-level system is the first system activated in the sysplex, RLS_MaxCfFeatureLevel defaults to Z, and all systems will be able to connect to the CF cache structure.

RLSTMOUT({nnn|0})
Specifies the maximum time, in seconds, that a VSAM RLS or DFSMStvs request is to wait for a required lock before the request is assumed to be in deadlock and ended with VSAM return code 8 and reason code 22(X'16'). Specify a value in seconds between 0 to 9999. A value of 0 means that the VSAM RLS or DFSMStvs request has no timeout value; the request waits for as long as necessary to obtain the required lock.

VSAM RLS detects deadlocks within VSAM and DFSMStvs. It cannot detect deadlocks across other resource managers, and uses the timeout value to determine when such deadlocks might have occurred. You can specify a global timeout value in the IGDSMSxx member of SYS1.PARMLIB, a step level timeout value on the JCL, or a timeout value on the RPL passed for each VSAM request.

For a particular VSAM RLS or DFSMStvs request, the value used for timeout is in the following order:

- The value specified in the RPL, if any
- The value specified in JCL at the step level, if any
- The value specified in the IGDSMSxx member of SYS1.PARMLIB, if any

RLSTMOUT is valid a parameter for either VSAM RLS or DFSMStvs. If you specify RLSTMOUT but do not specify the TVSNAME parameter, the value is used only by RLS. For DFSMStvs, the first instance of DFSMStvs brought up within the sysplex determines the value. Subsequent DFSMStvs instances use the value established by the first system, regardless of what might be specified in their members of SYS1.PARMLIB.

RLSTMOUT can be specified only once in a sysplex and applies across all systems in the sysplex.

Default: 0

RLSABOVETHEBARMAXPOOLSIZE({(ALL,size)})
RLSABOVETHEBARMAXPOOLSIZE({(sysname1,size1[...;sysname32,size32])})

Specifies the total size of the buffer management facility (BMF) buffer pool that resides above the 2-gigabyte bar for either of the following environments:

- All systems
- Each system referenced of the following parameter

Valid values are between 500MB and 2,000,000MB (2 Terabytes).

Default: 0

RLSFIXEDPOOLSIZE({(ALL,size)})
RLSFIXEDPOOLSIZE({(sysname1,size1[...;sysname32,size32])})

Specifies the amount of the total real storage, both above and below the 2 gigabyte bar, that will be permanently fixed (pinned) on either of the following environments:

- All systems
- Each system referenced in the parameter

Default: 0

SAM_USE_HPF({YES|NO})

Specifies whether you want SAM (BSAM and QSAM) and PAM to use HPF when it is available. This has an effect only if you enable HPF by specifying ZHPF=YES on the ZHPF statement in the IECIOSxx parmlib member.

- YES specifies that you want SAM and PAM to use HPF **when it is available and enabled**. If you specify or default to SAM_USE_HPF(YES) in either IGDSMSxx or on the SET SMS command, SMS sets on a bit in the DFA, DFASAMHPF.

  Note that because YES is the default, you will see SAM_USE_HPF=YES when you display SMS options for the system, even if it is not attached to an HPF-capable device.
- NO specifies that SAM and PAM should not use HPF. If you specify SAM_USE_HPF(NO) in either IGDSMSxx or on the SET SMS command, the DFASAMHPF bit in the DFA control block is set off. SAM and PAM do not use HPF

The following list are cases in which BAM does not use HPF even though IOS allows it:

- In a partitioned concatenation one or more of the data sets is a PDS that resides on a device that does not support HPF.
- An I/O error occurred with HPF. BAM switches to CCW channel programs and does not resume HPF for the currently open DCB.
- Certain temporary conditions cause BAM to switch to CCWs temporarily but resume HPF for later channel programs.

Default: YES

SELECT({event[,event][,...]|ALL})
Specifies one or more events or services that SMS is to trace (if SMS tracing is active). See the description of the DESELECT parameter for a list of valid events.

Default: ALL

Note: DESELECT(ALL) must precede the SELECT parameter if only specific trace options are to be traced.

SIZE(nnnnnnK|nnnM|nnnnnn|128K)
Specifies the size of the SMS trace table in bytes. If you specify a value of 0, no tracing is performed.

nnnnnnK specifies the size in kilobytes; the value can range from 0K to 255000K (255,000 kilobytes), and it is rounded up to the nearest 4K unit.

nnnM specifies the size in megabytes; the value can range from 0M to 255M (255 megabytes).

If you specify *nnnnnn* without a unit, the system assumes a unit of kilobytes.

Default: 128K

SMF_TIME(YES|NO)
SMF_TIME specifies whether DFSMS is to use SMF timing; that is, whether SMF type 42 records are to be created at the expiration of the SMF interval period, synchronized with SMF and RMF data intervals.

The following SMF record 42 subtypes are affected when you specify SMF_TIME(YES): 1, 2, 15, 16, 17, 18. If you record these subtypes, you can use SMF_TIME(YES) to synchronize SMF type 42 data with SMF and RMF data intervals.

Specifying SMF_TIME(YES) overrides the following IGDSMSxx parameters: CACHETIME, CF_TIME.

Default: YES

STEPNAME(STEPNAME|*)
Limits the number of volume selection analysis messages activated by VOLSEGMSG(ON) for a certain stepname or for all stepnames. Specify asterisk (*) to indicate *all* stepnames.

Default: *

SUPPRESS_DRMSGS (YES|NO)
Specifies whether SMS suppresses DELETE/RENAME messages issued to the hardcopy log and job log.

- Specifying YES suppresses DELETE/RENAME messages to the hardcopy log and job log.
- Specifying NO does not suppress DELETE/RENAME messages to the hardcopy log and job log.

Note that this parameter does not suppress callers from issuing the DELETE/RENAME messages. Some callers of SMS issue these messages and some do not. The SCOPE of this parameter is the entire system.

Default: NO

SUPPRESS_SMSMSG (YES|NO,IGD17054I,IGD17227I,IGD17395I)
Specifies whether SMS messages, IGD17054I, IGD17227I, and/or IGD17395I are to be issued or suppressed. The user must specify one or more of the

applicable messages in this parameter. If YES is specified, the messages specified are suppressed. If NO is specified, the messages specified are issued.

Default: NO

SYSNAME(sysname1[,sysname2[...,sysname32]])
Specifies the name of the system or systems on which DFSMStvs instances are to run. You can specify up to 32 system names, which must be in the same order as the DFSMStvs instance names. SMS examines the specified system names and compares them to the system names in the CVT. When it finds a match, SMS stores the value of the TVSNAME parameter in the matching position as the DFSMStvs instance name for the system.

Use the combination of SYSNAME and TVSNAME when the PARMLIB member is shared between systems. If no SYSNAME parameter is specified, the TVSNAME parameter applies to the system on which the PARMLIB member is read.

If you specify the SYSNAME parameter, you must also specify the TVSNAME(*nnn*) parameter. If you specify SYSNAME, but the system name is not specified, DFSMStvs is not started on the system.

Default: There is no default.

SYSTEMS({8|32})
This keyword specifies whether the system is running in compatibility mode (8-name limit) or 32-name mode.

SYSTEMS(8) specifies that a maximum of 8 system names, system group names, or both, can be specified for the SMS configuration. This value indicates that the system is running in compatibility mode and can share configurations (SCDSs or ACDSs) and COMMDS with systems that are running down-level releases of z/OS. Essentially, the system continues to operate as before DFSMS/MVS 1.3, whose date of general availability was in 1995.

SYSTEMS(32) specifies that a maximum of 32 system names, system group names, or both, can be specified for the SMS configuration. This value indicates that the system is not running in compatibility mode, and therefore the ACDS, SCDS and COMMDS cannot be shared with any systems that are running in compatibility mode.

Note:

1. When a system in an SMS complex begins running at a level with DFSMS/MVS Version 1.3 or later, the constructs in the SMS configuration are automatically expanded to prepare for the additional system information. This change occurs for all SMS configurations regardless of whether SMS 32-name support is used.
2. When you are using RESERVE or RELEASE on the volumes that contain shared SMS control data sets with other systems, the potential for deadlock increases dramatically. Even if the ACDS and COMMDS are allocated as the only data sets on separate volumes, deadlocks can occur. To avoid contention problems, you can use global resource serialization (GRS) to convert hardware reserves on the ACDS and COMMDS to SYSTEMS scope enqueues. Also, you should place resource name IGDCDSXS in the RESERVE conversion resource name list (RNL) as a generic entry. This can minimize contention delays and prevent deadlocks, which are sometimes associated with the VARY SMS command.

3. When you convert an SMS complex from compatibility mode to 32-name mode, the conversion is permanent. Be sure to copy your current SMS control data sets to data sets with different names before you begin the conversion.
4. You can do the conversion through the operator console or an ISMF panel. When you begin to convert, take the following steps:
   a. Set a date and time for the conversion when the allocation activity on the systems is low.
   b. Create an SCDS, ACDS, and COMMDS by copying existing SMS control data sets. Be sure to use the new formulas for data set space requirements. To provide for emergency fallback, do not convert your existing primary SCDS, ACDS, and COMMDS at the outset.
   c. On each system in the SMS complex, create an IGDSMSxx member in SYS1.PARMLIB that points to the new ACDS and COMMDS and specifies SYSTEMS(32).
   d. Immediately before conversion, issue SETSMS INTERVAL(999) on all systems involved in the conversion. This prevents a system from rereading the COMMDS and attempting to activate the new configuration before you have an opportunity to change the mode of that system.
   e. On each system, issue SET SMS=xx to restart SMS or re-IPL the system using the new IGDSMSxx member of SYS1.PARMLIB. This restarts SMS in 32-name mode using the new ACDS and COMMDS.

   Restarting SMS resets the INTERVAL parameter for the system. If INTERVAL is specified in the IGDSMSxx member of SYS1.PARMLIB, that value is used, otherwise, the default value is 15.

Default: 8

TRACE(ON|OFF)
Specifies whether SMS tracing is to be on or off.

Default: ON

TRACEEXIT(user-trace-exit)
Defines an installation exit for SMS tracing. The name of the exit must be 1 to 8 alphanumeric characters and it must be a valid load module. There is no default. For more information about the SMS tracing exit, see *z/OS DFSMS Installation Exits*.

TVSNAME(nnn1[,nnn2[...,nnn32]])
Specifies the identifier or identifiers that uniquely identify DFSMStvs instances that run in the sysplex. You can specify up to 32 identifiers, each of which must be a numeric value from 0 to 255. Each identifier must be unique within the sysplex. DFSMStvs uses this identifier as the last byte of its instance name (although it displays as three bytes). If the TVSNAME parameter is specified without the SYSNAME parameter, only a single value can be specified, and the name applies to the system on which the PARMLIB member is read. The numeric digits are appended to the characters IGWTV to form the DFSMStvs instance name.

You can use the TVSNAME parameter without the SYSNAME parameter when the PARMLIB member is not shared between systems. If no TVSNAME parameter is found, DFSMStvs processing is not available on the system.

Default: There is no default.

TV_START_TYPE({WARM|COLD}[,{WARM|COLD}[...,{WARM|COLD}]])
Specifies the type of start that each instance of DFSMStvs is to perform. Up to 32 TV_START_TYPE values can be specified. TV_START_TYPE values must be specified in the same order as DFSMStvs instance names. If WARM is specified, DFSMStvs reads its undo log and processes the information found in accordance with the information that RRS has about any outstanding units of recovery. If COLD is specified, DFSMStvs deletes any information remaining in the undo log and starts as if the log were empty. Use COLD only when the DFSMStvs undo log has been damaged.

Recommendation: Do not cold start DFSMStvs unless the DFSMStvs system logs have been damaged. After a cold start, data sets for which recovery was not completed could be left in a damaged state and must be recovered manually. If the data sets are forward recoverable, their forward recovery logs might also be damaged. IBM suggests that you manually recover the data sets (without using forward recovery), take back ups of them and any other data sets using the forward recovery log, then delete and redefine the forward recovery log.

You can allow some values to default and others to be specified. For example, specifying the keyword in the following way would cause the DFSMStvs instance in the second position to warm start, while the first and third DFSMStvs instances would cold start.

```
TV_START_TYPE(COLD,,COLD)
```

If both TVSNAME and SYSNAME parameters are also specified, TV_START_TYPE applies to the system specified on the SYSNAME parameter preceding or following it. If TV_START_TYPE is found with only the TVSNAME parameter, it applies to only the system on which the PARMLIB member is being read.

Default: The default is WARM.

TYPE(ERROR ALL[(TRACE|T|VOLSELMSG|V)[,ALL|ERROR(TRACE|T|VOLSELMSG|V)]])
Specifies how you want to trace events (TRACE) and issue volume selection analysis messages (VOLSELMSG).

ERROR
Specify ERROR to trace error events and issue volume selection analysis messages (VOLSELMSG(ON)) for allocations that have failed.

ALL
Specify ALL to trace all events and issue volume selection analysis messages (VOLSELMSG(ON)) for all allocations.

[(TRACE|T|VOLSELMSG|V)[,ALL|ERROR(TRACE|T|VOLSELMSG|V)]]
These are optional sub-parameters. TRACE|T or VOLSELMSG|V associated with the first sub-parameter specifies whether the required value, ALL or ERROR, specified in the first sub-parameter applies to the TRACE or VOLSELMSG facility. The second sub-parameter is optional and can be used to specify another value and facility after the first sub-parameter is specified. When none of these optional sub-parameters are specified, the value specified in the first sub-parameter applies to both TRACE and VOLSELMSG. For example, if you want to set a TYPE value of ERROR for the SMS TRACE facility, and a TYPE value of ALL for the VOLSELMSG facility, you could code:
`TYPE(ERROR(TRACE),ALL(VOLSELMSG))`.

If you specify VOLSELMSG(nnnnn), where nnnnn has a value greater than 0, with TYPE(ALL), you must also specify one of the following parameters to limit the number of detailed analysis messages:
- JOBNAME
- ASID
- STEPNAME
- DSNAME

Default: ERROR

USEEAV({YES|NO})
Specifies, at the system level, whether SMS can select an Extended Address Volume during volume selection processing. This check applies to new allocations and when data sets are extended to a new volume.

YES
This means that Extended Address Volumes can be used to allocate new data sets or to extend existing data sets to new volumes.

NO This is the default and means that SMS does not select any Extended Address Volumes during volume selection. Note that data sets might still exist on Extended Address Volumes in either the track-managed or cylinder-managed space of the volume.

When SMS is not active in the system, USEEAV is not available and the installation must use alternate means to control the usage of Extended Address Volumes.

Default: NO

USE_RESOWNER({YES|NO})
Indicates whether construct authorization checking is done using the RESOWNER value, which is based on the high-level qualifier of the data set name, or using the allocating user ID for the data set. If you specify NO, the RESOWNER value is not extracted and the allocating user ID is used.

Default: YES

VOLSELMSG({ON|OFF|0|nnnnn|ALL})
Allows you to control volume selection analysis messages issued when you create or extend an SMS-managed data set to a new volume. These analysis messages are written to the hardcopy log and the job log.

ON|OFF
Controls whether SMS volume selection analysis messages are to be issued.

Default: OFF

0|nnnnn|ALL
Controls whether SMS volume selection analysis messages are to be issued.

Default: 0

0 Indicates only summarized analysis messages are to be issued.

nnnnn
The number of volumes to be included in the message with a range of 0 to 65535.

ALL
Indicates that all volumes used for volume selection will be included in detailed analysis messages.

If you specify VOLSELMSG(nnnnn), where nnnnn has a value greater than 0,

with TYPE(ALL), you must also specify one of the following parameters to limit the number of detailed analysis messages:

- JOBNAME
- ASID
- STEPNAME
- DSNAME

When all volumes are to be included, volumes are listed by storage group. If only a subset of volumes is to be included, volumes are listed in volume selection preference order without an association to storage group. The system can issue an excessive number of analysis messages to the spool for the following conditions:

- The job or address space creates or extends many SMS-managed data sets
- Many volumes are to be included in the analysis messages.

Examples of the contents of IGDSMSxx

Assume that you want to define SMS with the following properties:

- An ACDS named SYS1.ACDS9.
- A COMMDS named SYS1.COMMDS.
- An interval of 15 seconds before synchronizing with any other SMS subsystems in the complex.
- SMF type 42 records are to be synchronized with the SMF and RMF intervals.

Figure 23 shows the statement that you would code in IGDSMSxx to request that SMS have these properties:

```
SMS ACDS(SYS1.ACDS9) COMMDS(SYS1.COMMDS) INTERVAL(15)
    SMF_TIME(YES)
```

Figure 23. Example: contents of IGDSMSxx

The next example defines a PARMLIB member that is specific to one system and sets up the following properties:

- An ACDS of SYS1.ACDS9
- A communications data set of SYS1.COMMDS
- An interval of 10 seconds before synchronizing with any other SMS subsystems in the complex
- A DFSMStvs name in the form of IGWTVnnn, for example, IGWTV001
- A quiesce timeout value of 240
- An activity keypoint (AKP) value of 2000
- No log of logs
- Requests DFSMStvs to perform a warm start

Figure 24 on page 570 shows the statement that you would code in IGDSMSxx to request that SMS have these properties.

```
SMS ACDS(SYS1.ACDS9) COMMDS(SYS1.COMMDS) INTERVAL(10)
  TVSNAME(1) QTIMEOUT(240) AKP(2000) TV_START_TYPE(WARM)
```

Figure 24. Example: contents of IGDSMSxx, example 2

The next example defines a PARMLIB member that applies to multiple systems and sets up the following properties:

- An ACDS of SYS1.ACDS10
- A communications data set of SYS1.COMMDS10
- An interval of 20 seconds before synchronizing with any other SMS subsystems in the complex
- Three instances of DFSMStvs name IGWTV001, IGWTV002, and IGWTV003 to be established on the systems named SYSTEM1, SYSTEM2, and SYSTEM3, respectively.
- The default quiesce timeout value (300)
- The default activity key point value (1000)
- A log of logs, using a log stream name IGWLOG.LOGLOGS
- Requests that DFSMStvs perform a cold start on all systems

Figure 25 shows the statement that you would code in IGDSMSxx to request that SMS have these properties.

```
SMS ACDS(SYS1.ACDS10) COMMDS(SYS1.COMMDS10) INTERVAL(20)
  LOG_OF_LOGS(IGWLOG.LOGLOGS)
  SYSNAME (SYSTEM1,SYSTEM2,SYSTEM3)
  TVSNAME( 1, 2, 3)
  TV_START_TYPE(COLD,COLD,COLD)
```

Figure 25. Example: contents of IGDSMSxx, example 3

Restrictions:

1. The DFSMStvs instance names must be unique.
2. After a cold start, any data sets for which recovery was not completed, are most likely left in a damaged state and must be recovered manually. If the data sets are forward recoverable, their forward recovery logs might also be damaged. IBM suggests that you manually recover the data sets (without using forward recovery), take back ups of them and any other data sets using the forward recovery log, then delete and redefine the forward recovery log.

The next example defines a PARMLIB member that applies to multiple systems and sets up the following properties:

- An ACDS of SYS1.ACDS10
- A communications data set of SYS1.COMMDS10
- An interval of 20 seconds before synchronizing with any other SMS subsystems in the complex
- RLSABOVETHEBARMAXPOOLSIZE value of 1024
- RLSFIXEDPOOLSIZE value of 1050

Figure 26 on page 571 shows the statement that you would code in IGDSMSxx to request that SMS have these properties.

```
SMS ACDS(SYS1.ACDS10) COMMDS(SYS1.COMMDS10) INTERVAL(20)
  RLSABOVETHEBARMAXPOOLSIZE(ALL,1024)
  RLSFIXEDPOOLSIZE(ALL,1050)
```

Figure 26. Example: contents of IGDSMSxx, example 4

Examples of specifying DFSMStvs parameters

Table 27 shows examples of ways that TVS parameters might be specified.

Table 27. Example: specifying DFSMStvs parameters

Example	Valid or not valid
TVSNAME(1)	Valid. Applies only to one system. All other parameters are defaulted.
TVSNAME(1) AKP(500)	Valid. Applies only to one system. All parameters other than AKP are defaulted.
TVSNAME(1) SYSNAME(SYS1) RLSTMOUT(15)	Valid. Applies only to one system. All parameters except RLSTMOUT are defaulted.
TVSNAME(1,2) SYSNAME(SYS1,SYS2) QTIMEOUT(500)	Valid. Applies to two systems. All parameters except QTIMEOUT are defaulted.
TVSNAME(1, 2, 3) SYSNAME(SYS1,SYS2,SYS3) TV_START_TYPE(WARM,COLD,WARM) LOG_OF_LOGS(LOG.OF.LOGS)	Valid. Each parameter specifies the correct number of values.
TVSNAME(1, 2, 3) SYSNAME(SYS1,SYS2,SYS3) TV_START_TYPE(COLD,,WARM) AKP(,700,900)	Valid. The second TV_START_TYPE value and the first AKP value are defaulted.
TVSNAME(1, 2, 3) SYSNAME(SYS1,SYS2,SYS3) TV_START_TYPE(WARM,COLD,)	Valid. The third TV_START_TYPE value is defaulted.
QTIMEOUT(500) AKP(500,200)	Not valid. The TVSNAME parameter is required when DFSMStvs parameters are specified.
TVSNAME(1,2) AKP(500,200) QTIMEOUT(500)	Not valid. The SYSNAME parameter is required when more than one TVSNAME value is specified.
TVSNAME(1, 2, 3) SYSNAME(SYS1,SYS2,SYS3) TV_START_TYPE(WARM,COLD)	Not valid. TVSNAME and SYSNAME specify three values while TV_START_TYPE specifies only two.
TVSNAME(1) AKP(500,200)	Not valid. The TVSNAME parameter specifies only one value while AKP specifies two.
TVSNAME(1, 2, 3) SYSNAME(SYS1,SYS2,SYS3) TV_START_TYPE(WARM,COLD,WARME)	Not valid. The right number of values is specified on each parameter, but WARME is not a valid value for TV_START_TYPE.

Chapter 60. IGGCATxx (DFSMS catalog configuration)

IGGCATxx allows you to define catalog system parameters and make permanent changes between IPLs. You can specify other catalog parameters in the LOADxx member of SYS1.PARMLIB and the SYSCATxx member of SYS1.NUCLEUS. The IGGCATxx parameters are processed both at IPL and when CAS is restarted. See The Intersection of SYSCATxx, LOADxx, IGGCATxx, and MODIFY CATALOG Command in *z/OS DFSMS Managing Catalogs*.

Use the command DISPLAY IPLINFO,CATALOG to display the catalog system parameter members currently in effect at a given time.

Parameter in IEASYSxx (or specified by the operator)

You must specify the current IGGCATxx member or members in the CATALOG=xx parameter in IEASYSxx parameter. The *xx* suffix can be any 2 alphanumeric characters or national characters (@,#,$). The default is 00 (zeros).

- If the system finds no CATALOG parameter, it uses default IGGCAT00. If the system finds no IGGCAT00 default member, default values will be used.
- If you specify multiple IGGCATxx members in the CATALOG=*xx* parameter, the system processes them in the order specified. When the same parameter appears in multiple members, the value specified in the last member processed, becomes the value in effect.
- If the system does not find a IGGCATxx member matching the CATALOG=xx specification, the system issues a message for each missing IGGCATxx member.

Syntax rules for IGGCATxx

Follow the rules in "General syntax rules for the creation of members" on page 9. The following rules also apply to the creation of IGGCATxx parmlib members:

- Input ends with the end of file (EOF). No explicit continuation syntax is required.
- Blanks in any combination constitute a delimiter.
- Delimiters are not required between parameters with values; the right parenthesis after the specified parameter value is sufficient.
- Comments can appear in columns 1-71 and must begin with "/*" and end with "*/". Comments are allowed between parameters.
- If the parser finds an error in any parameter, the system issues message IEC386W to display the text in error up to column 71. The parser will continue to parse any parameters on the next line. If the same parameter that was found to be in error on any earlier line is found and valid, the system uses that value for the parameter.
- If the parser finds an error in any parameter on a line, parsing for that line stops and no parameters from that line are used. The parser continues to parse any parameters on the next line.
- All parameters should start and finish on the same line. For example the following is not a valid syntax:

```
VVDSSPACE( 10, 14
             )
```

- The parameters do not have to start in column one, but they must fit entirely within columns 1 and 71. The parser ignores any text beyond column 71.
- You can specify any of the parameters multiple times, but the parser uses only occurrences of any parameters are allowed. The last valid value is used for the parameter. For example, the following is allowed; in this case, the system uses VVDSSPACE(14,14) as the final value for VVDSSPACE:

  ```
  VVDSSPACE(10,10)
  VVDSSPACE(14,14)
  ```

- You can use blanks between or before values in a parameter. For example:
  - A specification of VVDSSPACE(10, 14) is valid, and the system interprets it as VVDSSPACE(10,14).
  - A specification of NOTIFYEXTENT(55) is valid, and the system interprets is as NOTIFYEXTENT(55).
- Although you can have blanks between or before values in a parameter, you cannot have blanks between the digits in a single parameter value. For example:
  - If you intend a value of VVDSSPACE(10,14) but specify VVDSSPACE(1 0, 14) the parameter is not valid.
  - If you intend a value of NOTIFYEXTENT(66), but specify NOTIFYEXTENT(6 6), this parameter is not valid.
- If you specify one or more IGGCATxx suffixes, but the system cannot find all of the matching members, it issues a message for each member that is not found and processing continues with the members that the system did find. If the system does not find any IGGCATxx members, including the default IGGCAT00 member, it uses the default parameter values.
- When the same parameter appears in multiple IGGCATxx members, the value that is specified in the last member processed becomes the value in effect. The last valid specification in the last member is used for the particular parameter. If a particular parameter is not specified in any of the parmlib members, a default value will be used for the parameter
- If all instances of a specific parameter are invalid or the parameter is not specified at all in any of the IGGCATxx parmlib members during an IPL, then the default value is substituted for the parameter. During a restart of the Catalog address space, the value that existed before the restart is retained for the parameter.
- For readability, enter only one parameter per line.

Syntax format of IGGCATxx

```
resource(minutes,action)
ALIASLEVEL(n)
AUTOADD(ON|OFF)
CATMAX(nnnn)
DELFORCEWNG(YES|NO)
DELRECOVWNG(YES|NO)
DSNCHECK(YES|NO)
DUMP(ON|OFF)
DUMPON(rc,rsn,mod)
DUMPON(rc,rsn,mod,cnt)
EXTENDEDALIAS(YES|NO)
GDGFIFOENABLE(YES|NO)
NOTIFYEXTENT(percent)
SYMREC(YES|NO)
SYS%(ON|OFF)
TAPEHLQ(name)
TASKMAX(nn)
TASKMIN(nnn)
TASKTABLESIZE(nnn
UPDTFAIL(YES|NO)
VVDSSPACE(primary,secondary)
VVRCHECK(YES|NO)
```

Statements and parameters for IGGCATxx

resource(*minutes*,*action*)

Specifies whether resource contention monitoring for a resource is active or inactive for catalog requests and, if active, how long the system should allow a catalog request to wait before notifying the operator and (optionally) re-driving the request.

resource specifies the name of a resource and can be any of:

- ALLOCLCK, which is an internal CAS lock that serializes access to CAS allocation task responsible for most catalog allocation events.
- SYSIGGV2, which is used to serialize access to associated Catalog records.
- SYSZTIOT, which is used to control access to task input/output table resources.
- SYSZVVDS, which is used to serialize access to associated VVDS records. The SYSZVVDS reserve, along with the SYSIGGV2 reserve, provide a mechanism to facilitate cross system sharing of catalogs.

minutes is the wait time in minutes. Valid values range from 0 minutes (minimum) to 65535 minutes (maximum). Specifying 0 (zero) for *minutes* will make resource contention monitoring inactive for that resource. Any value from 1 through 4 will be reset to 5 minutes. Any value greater than 9999 will be reset to 9999 minutes.

action is the action to be taken when the specified wait time has been exceeded. *action* can be either N (for notify only) or R (for re-drive and notify). When re-drive is active, the first time a service task with an active resource passes the contention threshold, the service task is abended and the request is resubmitted to catalog for processing.

The default for all resources is active with threshold at 10 minutes and an action of notify only. These values determine the default for contention resources at IPL and post CAS RESTART. When no parmlib values exist at IPL, the resource is set to active with threshold of 10 minutes and action of notify

only. On CAS RESTART, in the absence of a parmlib value, the resource will retain whatever values it had prior to restart.

ALIASLEVEL(*n*)
Specifies the MLA search level. ALIASLEVEL has a default value of 1, with a minimum of 1 and a maximum of 4.

AUTOADD(ON|OFF)
Specifies whether the autoadd function is enabled when the first connection is made to the coupling facility by the catalog address space. The default is OFF.

CATMAX(*nnnn*)
specifies the maximum number of catalogs that can be opened concurrently in CAS. When the limit is exceeded, the least recently accessed catalog is closed, freeing the CAS storage it had occupied. Closed catalogs are not unallocated. They remain allocated, but in restart status with no CAS storage. If the new limit is less than the previous limit, all currently open catalogs are closed.

The minimum value is 1 and the maximum value is 9999. The number specified for *nnnn* is in decimal.

DELFORCEWNG(YES|NO)
specifies if you want the warning message IDC1997I or IDC1998I when attempting to use the DELETE VVDS RECOVERY or DELETE USERCATALOG FORCE command. The default is YES.

DELRECOVWNG(YES|NO)
specifies if you want the system to issue message IDC1999I if a DELETE UCAT RECOVERY command is attempted. The default is NO.

DSNCHECK(YES|NO)
specifies if you want to enable syntax checking on names being added to a catalog. The default is YES.

DUMP(ON|OFF)
Specifies whether or not dynamic dumps are available. The default is OFF.

DUMPON(*rc,rsn,mod[,cnt]*)
Specifies the return (*rc*) and reason codes (*rsn*) to take a dump on if a match is found in a valid catalog module (*mod*). The optional *cnt* value specifies how many times to skip before taking the dump. The default for *cnt* is 1.

EXTENDEDALIAS(YES|NO)
specifies if you want to enable the ability to create extension records for user catalog aliases on the current system. The default is NO.

GDGFIFOENABLE(YES|NO)
Specifies whether the GDG allocation order feature is enabled. The default is NO.

The GDG order feature allows the user to specify FIFO or LIFO (default) for the order in which the GDS list is returned for data set allocation when the GDG name is supplied on the DD statement.

NOTIFYEXTENT(*percent*)
Specifies the desired percentage threshold of the extents allocated for a catalog before the system issues message IEC361I to warn that the catalog is becoming full. The specified value is preserved across a CAS restart and is applied at IPL. Note that you can also change this value with a CAS restart.

The default is NOTIFYEXTENT(80). A percentage value of zero indicates that the monitoring for extent usage is suppressed. If a catalog exceeds 90%

utilization of the maximum extents, the system will issue message IEC361I even if the threshold has been set to zero.

SYMREC(YES|NO)
specifies if you want to enable SYMREC record creation. The default is YES.

SYS%(ON|OFF)
Specifies whether SYS% to SYS1% conversion is enabled. The default is OFF.

TAPEHLQ(*name*)
Specifies the high level qualifier of a tape volume catalog. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$) characters. The default is SYS1.

TASKMAX(*tasks*)
Specifies the Catalog address space (CAS) user service task upper limit, which is the maximum number of non-CAS concurrent service requests that can run at any given time.

- The value for TASKMAX cannot be **higher** than 90% of the CAS Service Task Table (TASKTABLESIZE) value specified in the SYS1.NUCLEUS(SYSCATxx) member. TASKMAX has a default value of 180, with a minimum of 24 and a maximum of 360.
- The value for TASKMAX cannot be **lower** than the Catalog address space service task lower limit specified in SYS1.NUCLEUS(SYSCATxx) or SYS1.PARMLIB(LOADxx) members. Default for the lower limit is 60.

You can display both the Catalog address space user service task upper limit and lower limit by issuing the `F CATALOG,REPORT` catalog command. You can use the MODIFY CATALOG command to decrease the TASKMAX value or restore it to 90% of the max number of concurrent catalog requests value specified in the SYSCATxx member. The new value specified on MODIFY CATALOG is then in effect until the next IPL.

TASKMIN(*mintasks*)
Specifies the lower limit on the number of catalog service tasks that can concurrently run. TASKMIN has a default value of 60, with a minimum of 24 and a maximum of 180.

TASKTABLESIZE(*maxtasks*)
Specifies the maximum possible tasks running at a particular time. This includes both catalog and non-catalog tasks. TASKTABLESIZE is an IPL only parameter. TASKTABLESIZE is ignored on a CAS restart (no error message is issued). TASKTABLESIZE has a default value of 200, with a minimum of 200 and a maximum of 400.

UPDTFAIL(YES|NO)
specifiesif you want the system to issue message IEC390I when a VSAM update request against a catalog has been abnormally terminated. This message is intended to alert the installation that potential catalog damage may have resulted from the incomplete request. The default is YES.

VVDSSPACE(*primary*,*secondary*)
Specifies the number of tracks for primary and secondary allocations that the Catalog Address Space (CAS) should use for an implicitly defined VVDS. The specified values are preserved across a CAS restart and are applied at IPL.

If a zero secondary value is specified on a VVDSPSACE(pri,sec) parameter and the primary value is valid, then on an IPL, a default of 10 tracks will be substituted for the secondary. On a restart of the Catalog Address Space the value that existed prior to the CAS restart is retained.

The default is VVDSSPACE(10,10).

VVRCHECK(YES|NO)
specifies if you want to enable enhanced VVR checking on VVDS I/O. The default is NO.

Chapter 61. IKJTSOxx (TSO/E commands and programs)

You can use IKJTSOxx to identify the commands and programs the system is to use. The IKJTSOxx member allows you to identify:

- Authorized commands and programs.
- Commands that a user cannot issue in the background.
- APF-authorized programs that users may call through the TSO/E service facility.

If your installation is using the virtual lookaside facility (VLF) and plans to use the TSO/E VLFNOTE command, add VLFNOTE to this parmlib member as an authorized command, using the AUTHCMD NAME parameter. See *z/OS MVS Programming: Authorized Assembler Services Guide* for information about when to use the VLFNOTE command.

IKJTSOxx also allows you to specify the defaults for the TSO/E ALLOCATE, SEND, RECEIVE, TRANSMIT, CONSOLE, and TEST commands.

By defining the SHR option on the ALLOCATE parameter, you can change the system default for the disposition of data sets from OLD to SHARE (SHR).

The TRANSREC statement allows you to specify the characteristics for the data to be transmitted or received.

The TEST parameter allows you to specify the installation-written test subcommands and the names of TSO/E commands that are to be allowed to execute under the TEST command. These commands are in addition to those allowed by default. Through the SEND statement, you can specify the data set into which the SEND command is to store messages. The system checks that named data set when the user issues the LISTBC command to retrieve stored messages. The SEND statement, with the MSGPROTECT keyword, also allows you to protect mail from being seen by users who do not meet security criteria. By itself, setting MSGPROTECT(ON) causes mail to be stored in a mail log named 'logname.userid', rather than 'userid.logname'. By combining MSGPROTECT(ON) with RACF 1.9, you can request that the system compare the level of security of the mail received with the level of security of the recipient.

For information about setting up RACF 1.9 to work with MSGPROTECT(ON), see *z/OS Security Server RACF Security Administrator's Guide*.

To use IKJTSOxx, copy the default from SYS1.SAMPLIB(IKJTSO00) to SYS1.PARMLIB(IKJTSOxx). You can then update the member to meet the needs of your installation.

Parameter in IEASYSxx (or specified by the operator)

IKJTSO=xx

The two-character identifier (xx) is appended to IKJTSO to identify the IKJTSOxx parmlib member. If you do not specify the IKJTSO=xx parameter, the system processes the IKJTSO00 parmlib member.

Selecting the IKJTSOxx member

If present, IKJTSO00 is used automatically during IPL. A different IKJTSOxx member can be selected during IPL by specifying IKJTSO=xx for the IPL parameters.

To select another IKJTSOxx member after IPL, you can issue the TSO/E command `PARMLIB UPDATE(xx)` or the system command `SET IKJTSO=xx`, where *xx* is the alphanumeric value to be appended to IKJTSO.

To list the values in the active IKJTSOxx parmlib member, you can issue either of the following commands:

```
DISPLAY IKJTSO

PARMLIB LIST
```

For example,

```
DISPLAY IKJTSO,ALLOCATE
              ,CONSOLE
              ,HELP
              ,LOGON
              ,NOTBKGND
              ,SEND
              ,TEST
              ,TRANSREC
              ,ALL
PARMLIB  LIST(ALLOCATE)
              (CONSOLE)
              (HELP)
(LOGON)
              (NOTBKGND)
              (SEND)
              (TEST)
              (TRANSREC)
              (ALL)
```

For more information about the DISPLAY command, see *z/OS MVS System Commands*. For more information about the PARMLIB command, see *z/OS TSO/E System Programming Command Reference*.

Syntax rules for IKJTSOxx

1. Comments may appear in columns 1-71 and must begin with a '/*' and end with '*/'.
2. Use a plus sign (+) or a dash (-) to continue a line. Any comments must appear to the left of the continuation character.
3. If the system encounters a syntax error in a IKJTSOxx statement, the default values are used for all the parameters on the statement. The system issues a message and processing continues.
4. Columns 1 through 72 may contain data.

IBM-supplied default for IKJTSOxx

IKJTSO00 is supplied and used automatically during IPL. To use another IKJTSOxx member, issue the SET IKJTSO=xx command or the PARMLIB UPDATE(xx) command after IPL. Either command will replace the current IKJTSOxx with the member specified in the command.

Statements/parameters for IKJTSOxx

ALLOCATE DEFAULT{(OLD)} | {(SHR)}
Allows you to specify the default value for the data sets required by a program. If you do not specify the ALLOCATE parameter, the system defaults to OLD.

AUTHCMD NAMES(cmd1,cmd2...)
Specifies the authorized TSO/E commands. ***cmd1,cmd2*** list the authorized commands. Each name can contain up to eight characters. See SYS1.SAMPLIB for the current list of authorized commands. If you are using VLF and plan to use the TSO/E VLFNOTE command, include VLFNOTE in the active IKJTSOxx member.

For the latest information about the commands, see the list of parmlib members in *z/OS V2R1 Program Directory*.

AUTHPGM NAMES(pgm1,pgm2...)
Specifies the authorized programs. ***pgm1,pgm2*** list the authorized programs. Each program name can contain up to eight characters.

For the latest information about the programs, see the list of parmlib members in *z/OS V2R1 Program Directory*.

AUTHTSF NAMES(name1,name2...)
Specifies the APF-authorized programs that may be called through the TSO service facility. The ***name1,name2*** identify the names of the programs. Each name can contain up to eight characters.

Note: Do not place programs from any IBM products in this table unless the owning product documentation specifically instructs you to do so. For example, do not put IDCAMS in AUTHTSF.

For the latest information about the programs, see the list of parmlib members in *z/OS V2R1 Program Directory*.

HELP language(dsname1[,dsname2,...])[,language(dsname1[,dsname2,...])]
Specifies message help texts for the specified languages.

language
Specifies the three character language code. See Table 30 on page 647 for the table of IBM-supported language codes.

dsname
Specifies the name of the data set that contains help text for the specified language. You can specify up to 255 help data sets for each language.

Default: ENU(SYS1.HELP)

LOGON
Specifies the system settings for the TSO/E LOGON command.

LOGONHERE(ON|OFF)
Specifies whether the RECONNECT option on the TSO/E LOGON panel will be honored even when the system does not detect a disconnected state and the user appears to be logged on. This allows users to reconnect their session from a new terminal without canceling their previous session first, similar to how the LOGONHERE option works under z/VM.

Default: ON

PASSPHRASE(ON|OFF)
Specifies whether the TSO/E logon panel allows users to enter up to 100 characters in the password field. If eight characters or less are entered, the input is treated as a password. If more than eight characters are entered, the input is treated as a passphrase. For more information about activating passphrase support, see *z/OS TSO/E Customization*.

Default: OFF

VERIFYAPPL(ON|OFF)
Specifies whether TSO/E determines an APPLID for the system the user is logging onto and passes that APPLID to RACF for verification during TSO/E logon. This option can be used to limit access to different systems that share the same RACF database. For information about setting up APPLIDs in RACF and implementing APPLID verification, see *z/OS TSO/E Customization*.

Default: OFF

NOTBKGND NAMES(cmd1,cmd2...)
Specifies the commands that may **not** be issued in the background. ***cmd1,cmd2*** list these commands. Each can contain up to eight characters. SAMPLIB contains the following commands:

OPERATOR	OPER
TERMINAL	TERM

CONSOLE
Specifies the installation's defaults for the TSO/E CONSOLE command.

INITUNUM(nnnn)
Specifies the initial number of **unsolicited** messages that can be queued to the extended MCS console session (established through the CONSOLE command) at any time. This number does not include messages sent by the TPUT service. For information about the TPUT service, see *z/OS TSO/E Programming Services*.

When the actual number of queued messages reaches 80% of the specified number, installation exit IKJCNX50 is invoked. When the actual number of queued messages reaches the specified number, installation exit IKJCNX64 is invoked. For information about these installation exits, see *z/OS TSO/E Customization*.

Default: 1000 (decimal)

INITSNUM(nnnn)
Specifies the initial number of **solicited** messages that can be queued to the extended MCS console session (established through the CONSOLE command) at any time. This number does not include messages sent by the TPUT service. For information about the TPUT service, see *z/OS TSO/E Programming Services*.

When the actual number of queued messages reaches 80% of the specified number, installation exit IKJCNX50 is invoked. When the actual number of queued messages reaches the specified number, installation exit IKJCNX64 is invoked. For information about these installation exits, see *z/OS TSO/E Customization*.

Default: 1000 (decimal)

MAXUNUM(nnnnn)
Specifies the maximum number of **unsolicited** messages that can be queued to the extended MCS console session (established through the CONSOLE command) at any time. The system records this value at initialization time; this value can only be changed through the PARMLIB UPDATE command for active extended MCS console sessions. This number does not include messages sent by the TPUT service. For information about the TPUT service, see *z/OS TSO/E Programming Services* .

This value is used by console activation processing and is passed to the TSO/E message capacity exits (IKJCNX50 and IKJCNX64).

Default: 10000 (decimal)

MAXSNUM(nnnnn)
Specifies the maximum number of **solicited** messages that can be queued to the extended MCS console session (established through the CONSOLE command) at any time. The system records this value at initialization time; this value can only be changed through the PARMLIB UPDATE command for active extended MCS console sessions. This number does not include messages sent by the TPUT service. For information about the TPUT service, see *z/OS TSO/E Programming Services*.

This value is used by console activation processing and is passed to the TSO/E message capacity exits (IKJCNX50 and IKJCNX64).

Default: 10000 (decimal)

PLATCMD{NAMES(cmd1,cmd2...)} | {NONE}
Specifies the commands (cmd1, cmd2,...) that will be executed on the TSO/E command/program invocation platform. These commands do not require task termination processing to clean up for them. Each command can contain up to eight characters. SAMPLIB contains the following commands:

ALLOCATE	ALLOC
ALTLIB	
ATTRIB	ATTR
EXEC	EX IKJEXC2
FREE	UNALLOC
PROFILE	PROF
SUBMIT	SUB
STATUS	ST

Default: NONE (no commands are eligible to run on the TSO/E command/program invocation platform).

For more information about the TSO/E command/program invocation platform, see *z/OS TSO/E Customization* .

PLATPGM{NAMES(pgm1,pgm2...)} | {NONE }
Specifies the programs (pgm1, pgm2,...) that are to be run on the TSO/E command/program invocation platform. These programs do not require task termination processing to clean up for them. Each program name can contain up to eight characters. SAMPLIB contains the following programs: IEFBR14 and IKJEFF76.

Default: NONE (no programs are to be run on the command/program invocation platform).

For more information about the TSO/E command/program invocation platform, see *z/OS TSO/E Customization*.

TEST TSOCMD(*cmd1,cmd2,cmd3.....*)
SUBCMD((*scmd1,load1*),(*scmd2,load2*)...)

The TEST parameter specifies that the following commands are authorized to be executed in a test environment. TSOCMD specifies that the following commands (***cmd1,cmd2...***) are installation-written TSO/E commands that are allowed to be executed under TEST. SUBCMD specifies that the following installation-written command can be invoked as a subcommand of TEST.

The value for ***scmd1*** is the command. The value for ***load1*** is the entry point for the program to be invoked as the subcommand. For each SUBCMD specified, you must include both the command and the program name for the command.

TRANSREC

TRANSREC allows you to specify the characteristics for the RECEIVE and TRANSMIT commands.

NODESMF{((*nodename1,smfid1*),(*nodename2,smfid2*),...)}
{((*,*))}

NODESMF specifies the correspondence between the system identifiers and the network node names.

nodename specifies the name of the network node. ***nodename*** must be the name of a node defined on the NJERMT JES3 initialization statement or on the NODE(xxxxxxxx) JES2 initialization statement.

The ***smfid*** specifies the system identifier for a particular processor, paired with a ***node-name***. ***smfid*** must be specified for each ***nodename***. ***smfid*** must match the system identifier defined for the processor on the SID parameter of the SMFPRMxx member.

, specifies that the nodename is to be retrieved dynamically from JES. This specification is recommended because it eliminates the need to specify static values for ***nodename*** and ***smfid***.

Default: (NODENAME, SMF)

Note: If you omit the ***smfid*** for the host node, TSO/E uses a value of eight question marks (????????) for the ***nodename*** associated with the transmitted data.

SPOOLCL(spoolclass)

Specifies the output class default. Use the SPOOLCL operand *class* to specify your installation's output class default.

This parameter applies to outgoing data only. Incoming data addressed to the issuer of RECEIVE or to the userid specified on the RECEIVE command by an authorized issuer of RECEIVE is eligible to be received regardless of its sysout class.

Value range: A-Z, 0-9, or *

Default: If you do not specify a different SPOOLCL operand, the default of 'B' is used.

CIPHER{(ALWAYS) | (YES) | (NO)}

CIPHER indicates the installation specification for controlling data encryption.

ALWAYS indicates that for every transmission, the data will be automatically encrypted.

YES indicates that encryption is a user option; this is the default.

NO indicates that encryption is not allowed on any transmission. (The specification of NO overrides the specification of the ENCIPHER operand on the TRANSMIT command and does not allow you to provide encryption through the TRANSMIT encryption exits, INMXZ03 or INMXZ03R.)

Default: YES

OUTWARN(n1,n2)

Allows the installation to specify the intervals at which a warning message is issued to a user who is transmitting a large file.

n1 specifies the number of records to be transmitted before the first warning message is issued to the user. The default is 10000 (decimal).

n2 specifies the number of records to be transmitted before second and subsequent warning messages are issued. The subsequent warning messages are issued each time this number of records is transmitted. The default is 5000 (decimal).

If you specify only one value with OUTWARN, the system uses that value as the first interval and uses the default (5000) for the second and subsequent intervals.

OUTLIM(n1)

OUTLIM specifies the maximum number of records a user can transmit before the transmission is terminated.

If n1 is less than or equal to 16777215, the system passes the value to JES as the OUTLIM value. If n1 is greater than 16777215, the system does not pass the value to JES. However, n1 still serves as the limit for the TRANSMIT command. *n1* should be greater than zero.

Note: The TRANSMIT command continues to work when the OUTLIM value is exceeded when writing to an OUTDDN or OUTDSN. The NOWARN option in the TRANSMIT command can be used to skip related warning messages. For more information about the NOWARN option, see *z/OS TSO/E Customization* and *z/OS TSO/E Command Reference*.

The TSO/E TRANSMIT command produces punched card output. Punched card output is limited by TSO/E or by JES, depending on which limit is lower. If the TSO/E limit is the lowest and is reached, the transmission is terminated and the following message is displayed at the user's terminal:

```
INMX032I   TRANSMIT command terminated.  Transmission limit of 'nn'
records exceeded.
```

If the JES limit is the lowest and is reached, the transmission is (one of the following):

- Allowed to continue.
- Abnormally ended.
- Abnormally ended with a dump.

IBM suggests that you set the TSO/E limit lower than the JES limit to allow the TSO/E user to receive messages that indicate whether the system successfully transmitted the data set.

For additional information about output limits, see:

- *z/OS MVS JCL Reference*
  - The OUTLIM DD statement

- *z/OS JES3 Initialization and Tuning Guide*
  - The OUTLIM parameter on the OUTSERV statement
- *z/OS JES2 Initialization and Tuning Guide*
  - The ESTPUN initialization statement
- *z/OS MVS Programming: Authorized Assembler Services Guide*
  - The dynamic allocation SYSOUT output limit specification DALOUTLM (key=X'001B')

Default: 30000 (decimal).

VIO(unitname)
VIO specifies the device type on which temporary space can be allocated for use by the TRANSMIT and RECEIVE commands. The ***unitname*** is the name of the device type and can be either an esoteric name (SYSDA) or a specific DASD device (for example, 3380).

If you do not specify VIO, the system defaults to the UNIT specification for the user in the UADS. If there is no UNIT specification, the system defaults to an installation-defined default or to the system default, SYSALLDA. (In each of the three preceding situations, IEBCOPY might fail.)

Note: IBM suggests that ***unitname*** be a device type that you designated as VIO at IPL. The use of VIO ensures the integrity of sensitive data.

LOGSEL(logselector)
LOGSEL specifies the default middle qualifier for the log data set name. (The name in the :LOGSEL tag in the control section of the NAMES data set takes precedence.) The ***logselector*** is 1-8 alphanumeric name of the middle qualifier. The first character must be alphabetic or special (#, @, or $). The names must be separated by a period.

LOGNAME(lognamesuffix)
LOGNAME specifies the default suffix qualifier for the log data set name. The following values take precedence over this parameter:

- LOGNAME operand of the TRANSMIT command.
- :LOGNAME tag in the control section of the NAMES data set.
- :LOGNAME tag in a nickname definition.

The ***lognamesuffix*** is the name of the suffix qualifier and must be 1-8 alphanumeric characters, beginning with an alphabetic or special (#, @, or $) character. The names must be separated by a period.

Default: In the absence of any explicit specification, the default log data set name is *userid*.LOG.MISC.

Maximum Length Restriction: TSO/E prefixes the name of the log data set with the user-specified *dsname-prefix* from the PROFILE command, so the name is equivalent to *prefix.logselector.logname-suffix*. The maximum length of the name is 44 characters, including the periods and the *prefix*.

USRCTL(name)
USRCTL is the name for the NAMES data set. The ***name*** must be 1-8 alphanumeric characters beginning with an alphabetic or special (#, @, or $) character. The names must be separated by a period.

Default: In the absence of any explicit specification, the default NAMES data set name is *userid*.NAMES.TEXT.

Maximum Length Restriction: TSO/E prefixes the name of the NAMES data set with the user-specified *dsname-prefix* from the PROFILE command. The maximum length of the name is 44 characters, including the periods and the *prefix*.

SYSCTL(datasetname)
SYSCTL specifies the name of an alternate NAMES data set. You can use this parameter, in conjunction with a routine you write, to provide a global standard set of nicknames within the installation. For example, the routine could manipulate the entries in a directory and store the resulting output — nicknames — in the SYSCTL data set. End users could then use the standard set of nicknames, instead of having to define their own nicknames on an individual basis.

The ***datasetname*** identifies the name of the data set and must be 1-8 alphanumeric characters, beginning with an alphabetic or special (#, @, or $) character. The names must be separated by periods. The total characters in the names including the periods cannot exceed 44 characters. TSO/E does not prefix the ***datasetname***.

SYSOUT(sysoutclass | *)
SYSOUT specifies the default SYSOUT class for messages that are written from utility programs, such as IEBCOPY. ***sysoutclass*** identifies the sysout class and can be A-Z, 0-9, or asterisk (*).

Default: SYSOUT(*) — the system writes the messages to the terminal.

DAPREFIX(TUPREFIX | USERID)
DAPREFIX specifies how the control and log data sets are to be prefixed for messages written by utility programs, such as IEBCOPY.

TUPREFIX indicates that the control and log data sets are to be prefixed with the PREFIX set in the User Profile Table. If the PROFILE NOPREFIX option is in effect, the log data set is 'LOG.MISC' and the control data set is 'NAMES.TEXT'. TUPREFIX is the default.

USERID indicates that the control and log data sets are to be prefixed with the user ID whenever the PROFILE NOPREFIX option is in effect. This prevents users from logging to 'LOG.MISC' and reading 'NAMES.TEXT' when PROFILE NOPREFIX is in effect.

SEND
Specifies the installation's defaults for the TSO/E SEND and LISTBC commands, and the OPERATOR SEND command. The defaults are shown here.

OPERSEND(ON| OFF),
OPERSEND specifies whether users who are authorized to use the OPERATOR command can issue the SEND subcommand to send messages or notes. For information about the OPERATOR and SEND commands, see *z/OS TSO/E Customization*.

USERSEND(ON | OFF),
USERSEND specifies whether users can issue the SEND command to send messages or notes to other terminal users. USERSEND is not valid for the SEND subcommand of the OPERATOR command. Use the OPERSEND parameter to specify whether authorized users of the OPERATOR command can use the SEND subcommand.

SAVE(ON | OFF),
SAVE specifies whether the SEND command processor and the OPERATOR SEND subcommand processor are to save messages in a log that the installation specifies.

USEBROD(ON | OFF)
USEBROD indicates whether messages intended for users who do not have individual user logs are to be stored in the SYS1.BRODCAST data set. (If the installation is not using individual user logs, the system ignores this operand.)

ON indicates that messages are to be stored in the SYS1.BRODCAST data set.

OFF indicates that messages are not to be saved and the user will not receive the messages.

ON is the default.

Table 28 on page 589 shows how LISTBC and SEND interpret the USEBROD and CHKBROD operands if the installation is using individual user logs. This figure applies to messages only; USEBROD and CHKBROD do not affect the processing of notices.

CHKBROD(ON | OFF)
CHKBROD indicates whether LISTBC processing is to check for messages in the broadcast data set and the user log and retrieve any messages found. LISTBC processing uses CHKBROD only when USEBROD is ON; that is, LISTBC processing ignores the broadcast data set when USEBROD is OFF.

ON indicates that the LISTBC processing is to check both the broadcast data set and the user log.

OFF indicates that the LISTBC processing is to check only the user log for messages.

OFF is the default.

Note:

1. CHKBROD applies only if you use user logs to store messages. It does not apply if you use only the broadcast data set to store messages. For more information about user logs, see *z/OS TSO/E Customization*.
2. If USEBROD is ON and the user log does not exist, LISTBC creates the user log and then checks SYS1.BRODCAST and retrieves any messages found. This checking and retrieval occur regardless of the CHKBROD setting.

LISTBC and SEND processing differ depending on the USEBROD and CHKBROD operands. If LOGNAME is set to 'SYS1.BRODCAST', the following conditions are true:

- The USEBROD and CHKBROD operands do not affect processing
- SEND and LISTBC use the broadcast data set for messages.

For the effects of USEBROD and CHKBROD on LISTBC and SEND processing when LOGNAME is not set to 'SYS1.BRODCAST', see Table 28 on page 589, which assumes the installation is using individual user logs.

Table 28. LISTBC and SEND Results Based on CHKBROD and USEBROD Settings when Installation is Using Individual User Logs

	USEBROD ON	USEBROD OFF
CHKBROD ON	• LISTBC: – The broadcast data set is always checked for messages, even when a user log exists. • SEND: – Messages are saved in the broadcast data set when no user log exists.	• LISTBC: – The broadcast data set is not checked for messages. • SEND: – Messages are not saved in the broadcast data set when no user logs exists. Message IKJ55058I is issued instead.
CHKBROD OFF	• LISTBC: – The broadcast data set is not checked for messages, except when LISTBC creates a new user log or when an existing user log cannot be allocated. • SEND: – Messages are saved in the broadcast data set when no user log exists.	• LISTBC: – The broadcast data set is not checked for messages. • SEND: – Messages are not saved in the broadcast data set when no user log exists. Message IKJ55058I is issued instead.

```
BROADCAST (DATASET(data-set-name) VOLUME(volume-name)
                                  TIMEOUT(time-out) switch-prompt)
```

Identifies the broadcast data set and the processing options to use when switching between broadcast data sets.

data-set-name
Specifies the fully qualified data set name. The use of quotes in the data set name is ignored; that is, 'SYS3.BRODCAST' is equal to SYS3.BRODCAST. DATASET is a required sub-keyword.

volume-name
Specifies the volume serial on which the broadcast data set resides. VOLUME is an optional sub-keyword.

time-out
Specifies the number of seconds a switch request will wait for resources before timing out. Valid values for TIMEOUT are integers in the range of 0 to 999, inclusive. TIMEOUT is an optional sub-keyword. The default value is 5 seconds.

switch-prompt
Specifies whether TSO/E should prompt before switching the broadcast data set. Valid values for *switch-prompt* are PROMPT and NOPROMPT. *switch-prompt* is an optional sub-keyword. The default value is PROMPT.

Note: When a PARMLIB UPDATE ROUTE command is issued, the *switch-prompt* setting is valid only when a switch is detected on the system. If no switch is detected, no prompt occurs, regardless of whether the command causes a broadcast data set switch on any systems that the settings are specified. If a separate prompt for each system is needed, consider using the MVS ROUTE *ALL,SET IKJTSO=xx command instead.

If the BROADCAST keyword is not specified, the default values are: SYS1.BRODCAST for *data-set-name*, no volume, five second *time-out*, and PROMPT for *switch-prompt*.

The data set must be:

- Formatted with the TSO/E SYNC command
- Cataloged or the volume serial must be specified on the BROADCAST keyword.

Note: TSO/E will use the name specified in the BROADCAST keyword or its default. Any entries for the SYSLBC DD name in MSTJCLxx will be disregarded by TSO/E.

MSGPROTECT(ON/OFF)

If you use user logs, use the MSGPROTECT operand to indicate whether the individual user log data set is security protected from the user. Set MSGPROTECT to one of the following values. The default value is OFF.

- ON — the individual user log is protected from the user and the messages (mail) within the individual user log can be viewed only if the user is logged on with the proper security label.

  If the MSGPROTECT operand is ON, the user log data set name is "logname.userid", where 'logname' is the data set name qualifier you specify on the LOGNAME operand of the SEND PARMLIB parameter and 'userid' is the user's TSO/E user ID. This naming convention protects the user log data set from the user when an installation has defined the RACF profile for LOGNAME.* UACC (NONE). Mail in the user log can be viewed by the user if the user is logged on at the proper security label by using the LISTBC command, or by requesting MAIL during LOGON.

  If RACF 1.9 or later is installed and your installation has set up security labels for your users using RACF, and the MSGPROTECT operand is ON, the security label of the sender is stored with the message in the user log data set. When the receiving user issues the LISTBC command to view messages, the security label of the receiving user is checked with the security label stored with the message for each message in the user log. The result of that check determines if LISTBC displays the message(s). If LISTBC does not display a message and the user is authorized at the security label of the message but is not currently logged on at the security label of the message, then the message remains in the user log. The user can log on to TSO/E with the proper security label and view the message at a later time. If LISTBC does not display a message and the user can never log on at the proper security label for the message (user is not authorized by RACF for that security label), the message gets deleted.

  Note: The MSGPROTECT operand should only be used with RACF 1.9 or later. If a lower level of RACF is installed on your system, MSGPROTECT will not protect messages. If you plan to use the MSGPROTECT operand with a lower level of RACF installed, each user log must be allocated by the installation.
- OFF — users can view their received mail without any security checking by the system.

  If MSGPROTECT is OFF, the user log data set name is "userid.logname," where 'userid' is the user's TSO/E userid and 'logname' is the data set name qualifier you specify on the LOGNAME operand. This naming convention allows the user to access the user log data set.

  If MSGPROTECT is OFF, the mail in the user log is not security protected from the user. The user can log on and get the messages by requesting MAIL or by issuing the LISTBC command.

If the MSGPROTECT value is switched from ON to OFF, messages that were left in the individual user logs on one setting cannot be retrieved until the settings are switched back.

Note: IBM suggests that installations create a user catalog and define an alias associated with this user catalog of LOGNAME. This prevents the master catalog from filling up with catalog entries for the new user logs.

LOGNAME(data-set-name/*)
LOGNAME identifies the log name of the data set where the system stores messages and notes.

To store messages (mail) in the broadcast data set, set LOGNAME to *. * is the preferred method to request that the mail be stored in the broadcast data set.

Note: For compatibility, the system accepts the specification of LOGNAME(SYS1.BRODCAST), and the user's mail is stored in the current broadcast data set. The BROADCAST keyword on the SEND statement is used to specify a broadcast data set name other than SYS1.BRODCAST.

To store messages in individual logs, set LOGNAME to the qualifier for the user log data set name. You can also specify a member name in parentheses.

The name of the user log data set depends on the setting of the MSGPROTECT parameter. If MSGPROTECT is off, the user log naming convention is as shown, where userid is the user's TSO/E user ID and LOGNAME is the qualifier you specify on the SEND PARMLIB parameter.

```
userid.LOGNAME
```

If MSGPROTECT is ON, the user log naming convention is

```
LOGNAME.userid
```

For example, suppose you set LOGNAME to *ulog.data*. If MSGPROTECT is OFF, the user log data set name is:

```
userid.ulog.data
```

If MSGPROTECT is ON, the user log data set name is:

```
ulog.data.userid
```

For more information about user logs, see *z/OS TSO/E Customization*.

SYSPLEXSHR(ON/OFF)
SYSPLEXSHR indicates whether the broadcast data set is shared outside systems in the sysplex. ON indicates that the broadcast data set is shared only among systems in the sysplex. OFF indicates that the broadcast data set is shared with systems outside the sysplex.

Note: IBM suggests that you set SYSPLEXSHR to the same value in every system's IKJTSOxx member. Doing so avoids confusion in your installation when modifying the the broadcast data set. A system with SYSPLEXSHR set to OFF does not communicate broadcast data set changes to systems with SYSPLEXSHR set to ON.

OPERSEWAIT(ON/OFF)
Indicates whether WAIT or NOWAIT is requested when an OPERATOR SEND command is issued without an explicit WAIT/NOWAIT operand. The default value is ON.

- ON - indicates you want the OPERATOR SEND command to be issued with WAIT. WAIT on OPERATOR SEND specifies that messages cannot be sent to any specified user until all users can receive them. That is, no user's terminal is busy (no user's output buffers are full).
- OFF - indicates you want the OPERATOR SEND command to be issued with NOWAIT. NOWAIT on OPERATOR SEND specifies that messages can be sent to any specified user whose terminal is not busy (output buffers not full).

USERLOGSIZE(primary-quantity,secondary-quantity,dir-block)
USERLOGSIZE specifies the amount of space allocated for the USERLOG data set if userlogs are used.

The ***primary-quantity*** specifies the number of tracks requested for the primary allocation of the user log data set. The default is 1 track.

The ***secondary-quantity*** specifies the number of tracks requested for secondary allocations of the user log. The default is 2 tracks.

The ***dir-block*** is an optional parameter that specifies the number of blocks requested for the directory if the user log is a partitioned data set. The default is 20 directory blocks.

Chapter 62. IOEPRMxx (z/OS Distributed File Service z/OS File System parameters)

IOEPRMxx contains the configuration options that control the z/OS Distributed File Service z/OS File System (zFS) environment and zFS aggregates to be attached at the start-up of zFS. The IOEPRMxx files are contained in the logical parmlib concatenation, defined on the PARMLIB parameter in LOADxx member. Alternatively, you can specify the z/OS File System (zFS) configuration options in the IOEFSPRM file, which is pointed to by the IOEZPRM DD statement in the ZFS PROC.

This section covers only concatenation and syntax rules. For information about parameters for the IOEPRMxx parmlib member and the IOEFSPRM file, see IOEFSPRM in *z/OS Distributed File Service zFS Administration*.

Specifying the IOEPRMxx concatenation

To concatenate data sets to the IOEPRMxx concatenation, you specify the suffixes of the IOEPRMxx data sets on the PARM option of the FILESYSTYPE TYPE(ZFS) statement in the BPXPRMxx parmlib member (see Chapter 13, "BPXPRMxx (z/OS UNIX System Services parameters)," on page 137). The PARM parameter is case sensitive; you must enter the string of suffixes in upper case. Figure 27 shows an example of a FILESYSTYPE TYPE(ZFS) statement.

```
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM)
ASNAME(ZFS,'SUB=MSTR')
PARM('PRM=(01,02,03)')
```

Figure 27. Example: FILESYSTYPE TYPE(ZFS) statement

You can specify any system symbol in the PARM parameter that resolves to two characters. For example, Figure 28 shows a FILESYSTYPE TYPE(ZFS) statement that uses system symbols.

```
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM)
ASNAME(ZFS,'SUB=MSTR')
PARM('PRM=(01,&SYSCLONE.)')
```

Figure 28. Example: FILESYSTYPE TYPE(ZFS) statement using system symbols

If symbol &SYSCLONE is equal to AB, this statement specifies that the system should search parmlib member IOEPRM01 first and IOEPRMAB second. You might use this specification if, for example, parmlib member IOEPRM01 contains common configuration options and IOEPRMAB contains configuration options specific to system AB. If the system does not find a particular parmlib member, the search for the configuration option continues with the next parmlib member.

You can specify up to 32 IOEPRMxx member suffixes in the PARM parameter. To specify 32 members, type member suffixes up to column 71 and then continue them in column 1 on the next line. For example, Figure 29 on page 594 shows an

FILESYSTYPE TYPE(ZFS) statement that specifies 32 IOEPRMxx members.

```
                                                                col 72
                                                                   |
                                                                   V
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS,'SUB=MSTR')
              PARM('PRM=(00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,
15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31)')
^
|
col 1
```

Figure 29. Example: FILESYSTYPE TYPE(ZFS) statement specifying 32 IOEPRMxx members

Syntax rules for IOEPRMxx

The following syntax rules apply to IOEPRMxx:

- Columns 73-80 are ignored in the IOEPRMxx member.
- Any line beginning with # or * is considered a comment.
- The text in the IOEFSPRM file is case insensitive. Any option or value can be upper or lower case.
- Blank lines are allowed.
- You should not have any sequence numbers in the IOEFSPRM file.
- If a configuration option is specified more than once, the first one found is taken.
- If a statement that has a default is omitted, the default is used.

Chapter 63. IPCSPRnn (Interactive Problem Control System)

An IPCSPRnn member contains parameters used during an IPCS session. These session parameters allow an installation to tailor IPCS sessions to its particular requirements. The parameters define the names of various data sets and default values to be used throughout that IPCS session. When you execute the IPCS command, IPCS initialization processes these parameters. Thus, you cannot modify or respecify them during an IPCS session.

The installation can have several IPCSPRnn members. You then specify the member that suits your needs for a particular IPCS session. See Chapter 11, "BLSCECT (formatting exits for dump and trace analysis)," on page 123 for information about IPCS parmlib member allocation.

Parameter in IEASYSxx (or specified by the operator)

None. The PARM(nn) keyword on the IPCS TSO command specifies the IPCSPRnn member that the system will use during that IPCS session.

Syntax rules for IPCSPRnn

- IPCS processes each statement as lines of TSO/E Command Language. Comments follow the rules of TSO/E Command Language and can be used wherever a delimiter can appear. Comments begin with /* and can end with */ on the same line. Additional options for ending comments are described within the TSO/E Command Language. For more details, see *z/OS TSO/E Command Reference* and *z/OS TSO/E User's Guide*.
- The IPCSPRnn member's suffix identifier can consist of numerals only. For example, IPCSPR01 is permitted, but IPCSPR1A is not.

IBM-supplied default for IPCSPRnn

The IPCS installation package includes the default member IPCSPR00.

Statements/parameters for IPCSPRnn

Multiple keywords are no longer functional: As of z/OS V1R11, the only functional keywords are LINELENGTH and PAGESIZE. The DSD, NODSD, PDR, NOPDR, PROBIDPREFIX, SYSTEM, GROUP, ADMINAUTHORITY, and DELETEAUTHORITY keywords will continue to be parsed but will ignored by IPCS when IPCSPRnn is processed.

DSD(datasetname)
Specifies the data set name of the IPCS data set directory. *datasetname* must be fully qualified and need not be in single quotation marks. IPCS appends nothing to either end of the specified name. This parameter is required. Without it, IPCS ends.

NODSD
Suppresses the use of problem and data management. This option requires the specification of the NOPDR option.

Default: NODSD

PDR(datasetname)
Specifies the data set name of the IPCS problem directory. *datasetname* must be fully qualified and need not be in single quotation marks. IPCS appends nothing to either end of the specified name. This parameter is required. Without it, IPCS ends.

NOPDR
Suppresses the use of problem and data set management. This option requires the specification of the NODSD option.

Default: NOPDR

PROBIDPREFIX(prefix)
Specifies the three-character value used to form a problem identifier where *prefix* is three alphanumeric characters. IPCS uses this prefix whenever an IPCS subcommand displays or prints a problem identifier. To form the complete problem identifier, IPCS concatenates a 5-digit decimal number to this prefix.

If the DSD(datasetname) and PDR(datasetname) parameters are specified, PROBIDPREFIX(prefix) is a required parameter. Without it, IPCS ends.

SYSTEM(system-id)
Specifies the default system identifier where system-id is one to eight alphanumeric characters. The ADDPROB subcommand of IPCS uses this value if you omit the SYSTEM keyword on that subcommand. This parameter is optional. The default is blank.

GROUP(group-id)
Specifies the default group identifier. *group-id* is one to eight alphanumeric characters. The ADDPROB subcommand of IPCS uses this value if you omit the GROUP keyword on the subcommand. This parameter is optional. The default group-id is blank.

ADMINAUTHORITY(userid-list)
Specifies the TSO/E user IDs of the persons that are to have IPCS administrative authority. *userid* is one to eight alphanumeric characters.

The owners of the specified TSO/E user IDs have the same authority that a problem owner has, but they have that authority over **all** problems in the problem directory.

Specifying administrators does not affect the privileges of problem owners. An owner can still modify and delete problems that he or she owns.

Specifying persons with administrative authority provides centralized control over all problems defined to IPCS. Administrators can assign and reassign problem owners, access a problem when the owner is unavailable, update problems when their attributes change, correct attributes that were incorrectly specified, and so forth.

This parameter is optional.

If not specified, no one has administrative authority for the installation.

If specified, it is a list whose entries are separated by one or more blanks, commas, or horizontal tabulation characters (X'05'). Each entry in the list can specify the user ID of an administrator or can specify an inclusive range of user IDs. The following example authorizes the person with user ID THEBOSS plus all persons with user IDs beginning with the letters ADMIN as administrators.

ADMINAUTHORITY(THEBOSS,ADMIN:ADMIN)

DELETEAUTHORITY(userid-list)
Specifies the TSO/E user IDs of the persons with IPCS delete authority. The owners of the specified TSO/E user IDs are the **only** persons who can delete problems. Specifying persons with delete authority diminishes the privileges of a problem owner and the person with administrative authority. They can modify problems but cannot delete them.

Specifying persons with delete authority provides centralized control over removing problems from IPCS. When a problem is deleted, all the accumulated information about it is lost. By designating persons who can delete problems, you can help prevent the inadvertent loss of such information.

This parameter is optional.

If specified, only the designated person can use the DELPROB subcommand of IPCS. Problem owners cannot use the DELPROB subcommand nor can the person designated with the ADMINAUTHORITY parameter (if the user ID specified is the different).

If this parameter is not specified, problem owners and the person specified with the ADMINAUTHORITY parameter can use the DELPROB subcommand.

If specified, it is a list whose entries are separated by one or more blanks, commas, or horizontal tabulation characters (X'05'). Each entry in the list can specify the user ID of an administrator or can designate an inclusive range of user IDs.

LINELENGTH(value)
Specifies the default record length (LRECL) for the IPCS print output data set. *value* is a two- or three- decimal digit from 83 through 255. This parameter is optional.

If you do not specify an LRECL for the print output data set and if it is not specified in the session parameters, the default is 137.

Value Range: 83-255

Default: 137

PAGESIZE(value)
Specifies the default number of lines per page for the IPCS print output data set. *value* is a decimal number ranging from 3 through 2^{31}-1. This parameter is optional. If you do not specify it, the default is 60.

PAGESIZE values should correspond with the number of lines that will fit on the forms typically used at your installation.

Default: 60

Chapter 64. IQPPRMxx (PCIE related parameters)

Use the IQPPRMxx parmlib member to specify parameters that are used for managing PCIE-related features, such as the zEDC Express® feature.

Use the SET IQP command to change the parameters while the system is active.

Parameter in IEASYSxx (or specified by the operator)

```
IQP={aa        }
    {(aa,bb...)}
```

This parameter identifies the IQPPRMxx parmlib member or members to be used for managing PCIE-related features, such as the zEDC Express feature. The two alphanumeric characters, represented by aa (or bb, and so on), are appended to IQPPRM to form the name of the IQPPRMxx member(s).

Syntax rules for IQPPRMxx

- Data is specified in columns 1-71. Columns 72-80 are ignored.
- You can enter one or more statements on a line or use multiple lines for one statement.
- The same statement can appear multiple times in a parmlib member. For example, a ZEDC statement with the MAXSEGMENTS parameter may be specified one line and another ZEDC statement with the DEFMINREQSIZE parameter may be specified on another line.
- Use blanks as delimiters. The system interprets multiple blanks as a single blank. You can use blanks between parameters and values. For example, both of the following parameter specifications are equally valid:
  - MAXSEGMENTS=10
  - MAXSEGMENTS = 10
- Comments can start in any column and can span lines. Comments must start with /* and end with */. Comments may appear in any place where a blank is accepted, except within quoted strings.
- Syntax errors cause the parameter to be ignored and processing to continue with the next parameter.
- If a parameter appears multiple times, the first valid instance is accepted, and subsequent occurrences are ignored.
- Enter values in uppercase, lowercase, or mixed case. The system converts the input to uppercase.

Statements/parameters for IQPPRMxx

ZEDC

Use the ZEDC statement to specify parameters for managing application requests that use zEDC Express features.

Parameters for the ZEDC statement:

MAXSEGMENTS=*nnn*
Specifies the maximum number of 16 MB storage areas (segments) to allow for problem state compression (deflation) and decompression (inflation) requests.

Value Range: 0, 4-64 (64 MB to 1 GB)

If zero is specified, problem state programs will not use zEDC Express features for compression and decompression requests.

If a non-zero value is specified, the system initially allocates and page fixes 4 segments for problem state requests. Additional segments are allocated and page fixed as needed up to the maximum number of segments. Segments that are no longer needed will be page freed and deallocated after a system determined amount of time.

If a SET IQP command is used to change the MAXSEGMENTS value to a lower value, the change is ignored and the original value remains in effect, because the maximum number of segments cannot be decreased dynamically. If a higher value is specified, the value is accepted.

Default: 4 (64 MB)

DEFMINREQSIZE=*nnnn*
Specifies the minimum size in kilobytes of the data to be compressed in order for the request to be eligible for compression.

Value Range: 4-9999 (4K to 9999K)

The request size must be greater than or equal to this size in order to be eligible for compression. Otherwise, software compression is used.

Default: 4 (4K)

INFMINREQSIZE=*nnnn*
Specifies the minimum size in kilobytes of the data to be decompressed in order for the request to be eligible for decompression.

Value Range: 4-9999 (4K to 9999K)

The request size must be greater than or equal to this size in order to be eligible for decompression. Otherwise, software decompression is used.

Default: 16 (16K)

Note: zlib is the problem state exploiter for zEDC. For more information about zlib, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Chapter 65. IVTPRM00 (Communication Storage Manager)

IVTPRM00 sets parameters for the Communication Storage Manager (CSM). IVTPRM00 is read during CSM initialization when the first IVTCSM REQUEST=CREATE_POOL macro is issued. (VTAM issues this macro when started.) The parameters can be changed without a re-IPL by editing the IVTPRM00 member and issuing the MODIFY CSM command with no command parameters specified.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax format of IVTPRM00

```
FIXED MAX(maxfix)
ECSA MAX(maxecsa)
POOL(bufsize,bufsource,initbuf,minfree,expbuf)
```

Restriction: FIXED MAX, ECSA MAX, and POOL must begin in column 1.

Syntax rules for IVTPRM00

System symbols can be used. For more information, see "Using MVS System Symbols" in the *z/OS Communications Server: SNA Network Implementation Guide*.

IBM-supplied defaults for IVTPRM00

The following are the IBM-supplied defaults for the CSM buffer pools.

bufsize	4K	16K	32K	60K	180K
initbuf	64	32	16	16	2
minfree	8	4	2	2	1
expbuf	16	8	4	4	2

Statements/parameters for IVTPRM00

FIXED MAX

Defines the maximum amount of storage dedicated to fixed CSM buffers.

maxfix

A decimal integer specifying the maximum bytes of fixed storage dedicated to CSM use.

Valid Range: 1024K to 30720M

Default Value: 100M

Note:

1. K indicates kilobytes, M indicates megabytes.

2. You must code only one blank between the keywords FIXED and MAX. If more than one blank appears between these keywords, the system ignores the statement as a comment and no syntax error message is generated. In this case, the system uses the default value of 100M.
3. The FIXED MAX statement must be completed one line.
4. No blanks should be coded between the keyword MAX and "(".

ECSA MAX
Defines the maximum amount of storage dedicated to ECSA CSM buffers.

maxecsa
A decimal integer specifying the maximum bytes of ECSA storage dedicated to CSM use.

Valid Range: 1024K to 2048M

Default Value: 100M

Note:

1. K indicates kilobytes, M indicates megabytes.
2. You must code only one blank between the keywords ECSA and MAX. If more than one blank appears between these keywords, the system ignores the statement as a comment and no syntax error message is generated. In this case, the system uses the default value of 100M.
3. The ECSA MAX statement must be completed one line.
4. No blanks should be coded between the keyword MAX and "(".

POOL
One POOL definition can be specified for each CSM buffer pool of a particular bufsize and bufsource combination.

bufsize
The size of the buffers in the pool to be created.

Valid Range: 4K, 16K, 32K, 60K, 180K

Default Value: None (valid range value required).

bufsource
The storage source from which buffers are allocated. The values are:

ECSA
Buffers are allocated from ECSA storage.

DSPACE
Buffers are allocated from data space storage.

initbuf
The initial number of buffers created in the pool when the first IVTCSM REQUEST=CREATE_POOL macro is issued by an application. If zero is specified, only the base pool structure is created and the pool is expanded on the first IVTSCSM REQUEST=GET_BUFFER based on the expbuf specification. The pool is not reduced below the level specified by either initbuf or expbuf, whichever is higher.

Valid Range: 0 - 9999

Default Value: IBM-supplied default unless overridden by a CREATE_POOL request.

minfree

The minimum number of free buffers allowed in the pool at any time. The storage pool is expanded the the value specified in expbuf if the number of free buffers falls below this limit.

Valid Range: 0 - 9999

Default Value: IBM-supplied default unless overridden by a CREATE_POOL request.

expbuf

The number of free buffers by which the pool is expanded when the free buffers fall below the minfree value.

Valid Range:

Bufsize	Expbuf Range
4K	1 - 256
16K	1 - 256
32K	1 - 128
60K	1 - 68
180K	1 - 22

Default Value: None.

Chapter 66. IXGCNFxx (system logger initialization parameters)

IXGCNFxx contains statements that can be used to control the following functions when z/OS system logger starts or is restarted on the initializing system within the sysplex.

1. Identify the tracing options when system logger initialized.
2. Specify the logger monitoring intervals for warning and action messages.
3. Specify the ZAI parameters for the z/OS IBM zAware log stream client socket communication and log data buffering details in order to send log stream data to the IBM zAware server.

Specifying the IXGCNFxx members

The installation can optionally create one or more IXGCNFxx members and place them in SYS1.PARMLIB. The following methods can be used to specify the current IXGCNFxx members:

1. The operator can enter the SET IXGCNF=xx command at any time after IPL.
2. You can specify the IXGCNF=xx parameter in the IEASYSxx parmlib member during IPL, and then select IEASYSxx in the LOADxx parmlib member or specify SYSP=xx in response to the SPECIFY SYSTEM PARAMETERS system message.
3. You can specify IXGCNF=xx in response to the SPECIFY SYSTEM PARAMETERS system message.

You can specify one or more current IXGCNFxx parmlib members during IPL. For example, a IXGCNF=(00,01) system parameter informs the system to use IXGCNF00 and IXGCNF01 as current members. The system processes these members in the order they are specified. For all options, the last parmlib member specifying the option is honored.

If any errors are encountered for the specified IXGCNFxx parmlib members, logger will issue an error message and then attempt to process remaining parameters for syntax. For a SET IXGCNF command, parameter updates are not done. At IPL, default values are substituted for all parameters. For example, assume a shared IXGCNFxx parmlib includes the MONITOR LSPRIMARY,CONSUMPTIONALERT(*value*) specification. Also assume one release is at z/OS V1R13 and the other is at z/OS V2R1, The shared IXGCNFxx parmlib member will be honored on the z/OS V2R1 release. However, the z/OS V1R13 release processing will issue an unrecognized keyword type error message, and use the system defaults for all the IXGCNFxx options.

As part of the IXGCNF parmlib processing, logger might issue a TRACE CT or a DISPLAY LOGGER command. For information about required SAF authority, see the topic on defining authorization for the system logger address space in *z/OS MVS Setting Up a Sysplex*.

Before changing options with SET IXGCNF=xx, consider issuing a D LOGGER,IXGCNF command to see what options are currently in affect. Combine the displayed options with any new request that shares the same option statement. Note, the SETLOGR system command can also be used to make changes to the

system logger (IXGLOGR) address space processing, and can be used to dynamically change some of the options provided in parmlib.

The following restrictions apply to the IXGCNFxx parmlib parameters when attempted to be updated as part of the SET IXGCNF=xx command processing.

1. ZAI SERVER and PORT changes are not honored if there are existing socket connections to the IBM zAware server. Disconnect the applicable log streams or quiesce any existing socket connection through the SETLOGR FORCE,ZAIQUIESCE command (use the ALL keyword or LSN= for each applicable log stream).
2. Reducing the LOGBUFMAX value below the current in-use storage buffers will cause system logger to put the LOGBUFFULL specification into effect.

Related members in parmlib

1. COUPLExx as z/OS system logger requires at least a base sysplex configuration in order to function.
2. CTILOG00 for information pertaining to system logger CTRACE options (see Ctncccxx). Also see SYSLOGR component trace in *z/OS MVS Diagnosis: Tools and Service Aids*.

Parameter in IEASYSxx (or supplied by the operator)

```
IXGCNF={aa        }
       {(aa,bb...)}
```

This parameter identifies the IXGCNFxx parmlib member(s) to be used when system logger starts or is restarted on the initializing system within the sysplex. Syntax IXGCNF=aa specifies a single member and syntax IXGCNF=(aa,bb,...) identifies groups of system logger initialization statements across several IXGCNFxx members. The installation can optionally create one or more members and place them in SYS1.PARMLIB. When multiple IXGCNFxx members are concatenated, the individual system logger options are merged with the last parmlib member option taking precedence.

Syntax rules for IXGCNFxx

The following syntax rules apply to IXGCNFxx:

1. Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
2. Comments may appear in columns 1-71 and must begin with "/*" and end with "*/". A comment can span lines and can appear anywhere except within a keyword or a specified value.
3. Blank lines are allowed anywhere in the member.
4. You can use both uppercase and lowercase letters; the system translates lowercase letters to uppercase letters before processing.
5. The system recognizes the end of a statement when it encounters either the beginning of the next valid statement or an end-of-file (EOF) indicator.
6. Use valid delimiters to separate keyword parameters. A valid delimiter is a blank. The system treats multiple blanks as one.

7. Blanks can appear between keyword parameters, between values, and between statements. Blanks cannot appear within a keyword value.
8. Delimiters are required between keywords.
9. If the system finds a syntax error in IXGCNFxx, the system issues an error message, and then attempts to continue processing the next keywords, statements, and parmlib members, for syntax only. No updates take place on error.

IBM-supplied defaults for IXGCNFxx

There is no default IXGCNFxx parmlib member. SYS1.SAMPLIB provides a sample parmlib member IXGCNFXX.

Syntax format of IXGCNFxx

```
[  CTRACE(parmlib_member_name)                    ]
   [                                              ]
[  MONITOR                                  ]
   [  OFFLOAD                                     ]
   [     WARNALLOC(initial-delay-interval)        ]
   [     ACTIONALLOC(secondary-delay-interval)    ]
   [     WARNRECALL(initial-delay-interval)       ]
   [     ACTIONRECALL(secondary-delay-interval)   ]
   [     LSPRIMARY                                          ]
   [     CONSUMPTIONALERT(ALLOW | SUPPRESS)      ]
   [  . . .                                          ]
[  ZAI                                      ]
   [    SERVER(NONE | host_name | IP_addr)        ]
   [    PORT(port_num)                            ]
   [    LOGBUFMAX(value)                          ]
   [    LOGBUFWARN(nn)                            ]
   [    LOGBUFFULL(MSG|QUIESCE)                   ]
```

Statements/parameters for IXGCNFxx

CTRACE(*parmlib_member_name***)**

Specifies the member of SYS1.PARMLIB that contains the tracing options to be used when the z/OS system logger component is initialized (IXGLOGR address space starts) or when a SET IXGCNF command is entered. You can specify a CTnLOGxx parmlib member name.

Use this statement to provide initial tracing options while system logger is starting and as an alternative approach to issuing a CTRACE command to specify tracing options other than those in default member CTILOG00. For information about coding the component trace member of SYS1.PARMLIB, see Chapter 26, "CTncccxx (component trace parameters)," on page 283.

Logger converts the input to a TRACE CT,ON,COMP=SYSLOGR,PARM=CTnLOGxx command. Look at the syslog for IEE538I or other error messages to verify the options specified in the *parmlib_member_name* are accepted. You can verify the logger tracing options in effect by using the D TRACE,COMP=SYSLOGR command.

For more information about viewing (D TRACE) the current system logger trace settings and specifying new options for the system logger (SYSLOGR) component trace, see *z/OS MVS System Commandsz/OS MVS System Commands* and *z/OS MVS Diagnosis: Tools and Service Aids*, respectively.

Value range: The 8 character member name must be in the format CTnLOGxx where:

n An alphanumeric character to specify the source of the member. IBM-supplied members will use "I".

xx Any two characters.

Default: CTILOGOO

MONITOR
Provide system logger monitoring specifications.

OFFLOAD
Specify the initial and secondary delay intervals for monitoring system logger log stream offload activity. See Offload and Service Task Monitoring in *z/OS MVS Setting Up a Sysplex* for more information about the logger offload monitoring.

WARNALLOC(*initial-delay-interval***)**
Specifies the amount of time, in seconds, allowed before the log stream offload monitor will start issuing warning message indicating an offload is delayed waiting for a data set allocation or deletion request to complete.

If the data set allocation and/or deletion request does not complete within this interval, system logger issues message IXG310I to warn the operator of the log stream offload being delayed. If this value is not less than the ACTIONALLOC secondary-delay-interval, then logger will not issue the IXG310I warning message as part of its log stream offload monitoring.

D LOGGER,IXGCNF[,MONITOR] displays system logger monitoring settings.

Value range: Must be a decimal number from 5 to 86400 seconds (meaning 5 seconds to 24 hours, respectively).

Default: 30 (30 seconds, 1/2 minute)

ACTIONALLOC(*secondary-delay-interval***)**
Specifies the amount of time, in seconds, allowed before the log stream offload monitor will issue messages to allow possible action be taken by the installation because of an offload being delayed waiting for a data set allocation or deletion request to complete.

If the data set allocation and/or deletion request does not complete within this interval, system logger issues messages IXG311I identifying the log stream and data set resources, and logger also issues WTOR message IXG312E to allow action to be taken if necessary.

D LOGGER,IXGCNF[,MONITOR] displays system logger monitoring settings

Value range: Must be a decimal number from 5 to 86400 seconds (meaning 5 seconds to 24 hours, respectively).

Default: 60 (60 seconds, 1 minute)

WARNRECALL(*initial-delay-interval***)**
Specifies the amount of time, in seconds, allowed before the log stream offload monitor will start issuing warning message indicating an offload is delayed waiting for a migrated data set recall request to complete.

If the migrated data set recall request does not complete within this interval, system logger issues message IXG310I to warn the operator of the log stream offload being delayed.

If this value is not less than the ACTIONRECALL secondary-delay-interval, then logger will not issue the IXG310I warning message as part of its log stream offload monitoring.

D LOGGER,IXGCNF[,MONITOR] displays system logger monitoring settings.

Value range: Must be a decimal number from 5 to 86400 seconds (meaning 5 seconds to 24 hours, respectively).

Default: 60 (60 seconds, 1 minute)

ACTIONRECALL(*secondary-delay-interval***)**
Specifies the amount of time, in seconds, allowed before the log stream offload monitor will issue messages to allow possible action be taken by the installation because of an offload being delayed waiting for a migrated data set recall request to complete.

If the migrated data set recall request does not complete within this interval, system logger issues messages IXG311I identifying the log stream and data set resources, and logger also issues WTOR message IXG312E to allow action to be taken if necessary.

D LOGGER,IXGCNF[,MONITOR] displays system logger monitoring settings.

Value range: Must be a decimal number from 5 to 86400 seconds (meaning 5 seconds to 24 hours, respectively).

Default: 120 (120 seconds, 2 minutes)

LSPRIMARY
Specify the log stream primary (interim) storage monitoring.

CONSUMPTIONALERT(ALLOW | SUPPRESS)
Specifies whether log streams defined with WARNPRIMARY(YES) will issue log stream primary storage consumption alert messages on this system when the imminent threshold is exceeded or an entry threshold occurs, or for any log stream when a primary storage full consumption point is reached.

ALLOW indicates that the primary storage consumption and offload messages will be issued if the log stream is defined with the correct parameters.

SUPPRESS indicates that this system will suppress any alerts or messages, regardless of the log stream definition.

See*z/OS MVS Setting Up a Sysplex*.

Display LOGGER,IXGCNF[,MONITOR] system logger monitoring settings.

Default: ALLOW

ZAI
Specify the definitions for the z/OS IBM zAware log stream client socket communication and log data buffering details in order to send log stream data to the IBM zAware server.

By default, no communication with the IBM zAware server will occur unless it is explicitly requested and the z/OS IBM zAware log stream client requirements are established.

See Preparing for z/OS IBM zAware log stream client usage in *z/OS MVS Setting Up a Sysplex* for more details on establishing and using z/OS IBM zAware log stream clients.

SERVER({NONE | *host_name* | *IP_addr*})
Specifies the host name (as defined by domain name system, DNS) or TCP/IP

location where the "IBM System z Advanced Workload Analysis Reporter" (z/OS IBM zAware) server is running and will receive the z/OS IBM zAware log stream client data.

NONE
Indicates no IBM zAware server is desired and does not indicate a server name. Therefore, no z/OS IBM zAware log stream client is to be established by system logger.

host_name
Specifies the canonical host name, as defined by DNS, where the IBM zAware server is located:

- Consist of alphabetic letters (A-Z), numeric digits (0-9), hyphens(-) and periods(.),
- Consist of 1 to multiple sections (labels) of 1-63 characters separated by a period(.),
- Alphabetic characters are case insensitive,
- Each section must start and end with either an alphabetic character (A-Z) or numeric digit (0-9). For example: WWW.IBM.COM

Note: The DNS server does not need to be up and running when the IXGCNFxx parmlib member is processed as the *host_name* and will be resolved when the z/OS IBM zAware log stream client needs to make a connection to the IBM zAware server (see more details below).

IP_addr
Specifies an IP address for the IBM zAware server location. Can be either IPv4 or IPv6 address format.

IPv4_address must be specified as a dotted decimal quad: ddd.ddd.ddd.ddd format, that is:

- Must have 3 dots(.) and the rest of the characters must be decimal numbers (0-9),
- Must consist of 4 sections of 1-3 decimal digits separated by dots(.),
- The decimal digits must represent numbers between 0 and 255.
- Leading (non-significant) zeros for each number can either be specified or suppressed.
- Minimum length is 7 characters, maximum length is 15 characters.

For example: 192.0.2.255

IPv6_address must be specified with colons as delimiters, that is:

- Must have 2 to 7 colons (:) and contain hexadecimal numbers (0-9, A-F, a-f),
- Consist of up to 8 sections of 0-4 hexadecimal numbers separated by a colon(:).
- Leading (non-significant) zeros for each number can either be specified or suppressed.
- When all 8 sections are not specified, then zero compression via double colons must be used and can only appear once to designate the zero compression range.
- IP address cannot start or end with a colon unless part of a double-colon (zero compression range).
- Section specified prior to a double colon (zero compression range) correspond to left-justified sections 1 through nth specified section, and

sections specified after the double colon correspond to right-justified sections 8 down to number of sections following double colon.

- Minimum length is 2 characters, maximum length is 39 characters.

For example:

SERVER(2001:db8:85a3::8a2e:370:7334)
has 3 sections before and 3 sections after the double colon, so it would yield 16 byte (4 word) value of 20010DB8 85A30000 00008A2E 03707334.

SERVER(2001:db8:85a3::8a2e:370)
has 3 sections before and 2 sections after the double colon, so it would yield 16 byte (4 word) value of 20010DB8 85A30000 00000000 8A2E0370.

Note: When specifying an IPv6_address, all the routers in the path of the socket connection between the z/OS IBM zAware client and IBM zAware server must also support IPv6 format addresses, otherwise the socket connection can fail when attempted.

When a value other than NONE is provided, the z/OS system logger component will establish a socket connection for communication with the IBM zAware server on behalf of a z/OS IBM zAware log stream client. By default, no communication with the IBM zAware server will occur unless it is explicitly requested and thez/OS IBM zAware log stream client requirements are established.

D LOGGER,IXGCNF[,ZAI] and D LOGGER,STatus,ZAI commands display the details on the system logger z/OS IBM zAware log stream client information.

Value range: Up to 64 characters, consisting of alphabetic characters (A-Z), numeric digits (0-9), periods (.), colons (:), and hyphens (-).

- Considered an IPv6 address if there are colons (:).
- Considered and IPv4 address if there are numeric digits (0-9) and periods(.), but no alphabetic characters (A-Z) or hyphens(-).
- Otherwise is considered a host_name, unless NONE meaning no IBM zAware server. See format details above.

Default: None

Default port number for ZAI server is 2001.

PORT(*port_num*)
Identifies the port number associated with the SERVER specification for the IBM zAware server, as shown in the following examples:

- PORT(3801)
- PORT(10120)

D LOGGER,IXGCNF[,ZAI] and D LOGGER,STatus,ZAI commands display the details on the system logger z/OS IBM zAware log stream client information.

Value range: 1 to 65535 (up to 5 digits).

Default: 2001

LOGBUFMAX(*value*)
Identifies the maximum amount of storage buffers (in gigabytes) to be used by system logger for managing z/OS IBM zAware log stream client data being sent to the IBM zAware server. Allows clients ability to set a threshold (for

their environment) on the amount of (pageable, above the 2 gigabyte bar) storage that could be utilized on their z/OS system in the case of spiking or inhibited data flow to the IBM zAware server.

If the maximum for these storage buffers are used, then any new log stream data needing to be sent to the IBM zAware server will be lost. See LOGBUFFULL specification on how system logger will handle the z/OS IBM zAware log stream client data when a storage buffer full condition occurs.

D LOGGER,IXGCNF[,ZAI] and D LOGGER,STatus,ZAI commands display the details on the system logger z/OS IBM zAware log stream client buffer usage.

Value range: 1-99 (in gigabytes)

Default: 2 (2 gigabytes)

LOGBUFWARN(*nn*)
Specifies the z/OS IBM zAware log stream client manager buffer warning level percentage when system logger will start issuing message IXG375E.

When the z/OS IBM zAware log stream client manager is using this percent of overall buffer space (see LOGBUFMAX), message IXG375E will be issued. As the buffer pool is expanded/extended by the next incremental amount, system logger will issue an updated instance of IXG375E with the new buffer storage percent in use indication. As in-use system logger z/OS IBM zAware log stream client buffers are no longer needed, the buffers will be released and message IXG375E will be issued for the new percent in use.

When the overall percentage of buffers in use is at least 5% below the LOGBUFWARN value, system logger will DOM message IXG375E.

D LOGGER,IXGCNF[,ZAI] and D LOGGER,STatus,ZAI commands display the details on the system logger z/OS IBM zAware log stream client buffer usage.

Value range: 10-90 (10% to 90%)

Default: 75 (75% of LOGBUFMAX value)

LOGBUFFULL({MSG|QUIESCE})
Specifies the action to be taken by system logger when the z/OS IBM zAware log stream client manager buffers become full (see LOGBUFMAX).

MSG indicates system logger should continue attempting to send log data to the IBM zAware server and keep a count of the number of log blocks that could not be buffered and were lost. When buffers become available, then message IXG383I will be issued indicating that not all the log data was sent to the IBM zAware server.

QUIESCE indicates that when system logger reaches the full amount of buffers then disconnect the socket connection to the IBM zAware server. The socket connection will remain disconnected until either a SETLOGR command (for example, SETLOGR FORCE,ZAICONNECT,ALL) or SET IXGCNF command that has specifications to connect to the IBM zAware server. No z/OS IBM zAware log stream client data will be maintained while in the quiesced state, meaning the buffers holding the log data for this purpose will be released (freed). Logger will issue message IXG382I for each log stream that is quiesced.

D LOGGER,IXGCNF[,ZAI] and D LOGGER,STatus,ZAI commands display the details on the system logger z/OS IBM zAware log stream client buffer usage.

Default: MSG

Examples of IXGCNFxx member

The following example shows a IXGCNFxx parmlib member that requests that up to one z/OS IBM zAware log stream client can be established (for example,. for SYSPLEX.OPERLOG) to communicate with a IBM zAware server at host domain name system "plpsc.pok.ibm.com" via port # 10120. System logger will manage up to 1 gigabytes as the log stream data buffering limit with warning messages to occur when 80% in use and to issue a different message if all the log data buffers become full.

```
ZAI  SERVER(plpsc.pok.ibm.com) PORT(10120)
  LOGBUFMAX(1) LOGBUFWARN(80) LOGBUFFULL(MSG)
```

Chapter 67. LNKLSTxx (LNKLST concatenation)

Use the LNKLSTxx member of parmlib to specify the program libraries that are to be concatenated to SYS1.LINKLIB to form the LNKLST concatenation. In addition to the data sets you specify in LNKLSTxx, the system automatically concatenates data sets SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE and SYS1.SIEAMIGE to SYS1.LINKLIB. You can create any number of LNKLSTxx members.

Using PROGxx to define LNKLST concatenations

Instead of using LNKLSTxx to specify the LNKLST concatenation, consider using PROGxx.

If you specify both PROG=xx for a member with a LNKLST ACTIVATE statement and LNK=xx, the system ignores LNK=xx at IPL and issues message CSV487I `LNK IPL PARAMETER HAS BEEN IGNORED. LNKLST SET` *`lnklstname`* `IS BEING USED`. The LNKLST set refers to the data sets that are defined to the LNKLST concatenation through LNKLST statements in PROGxx. See “Using the LNKLST statement” on page 701.

Using LNKLSTxx

If you use LNKLSTxx instead of PROGxx, during IPL, the system opens and concatenates each data set in the order it was listed, starting with the first-specified LNKLSTxx member. The system creates a data extent block (DEB) that describes the data sets concatenated to SYS1.LINKLIB and their extents. The extents remain in effect for the duration of the IPL.

After this processing completes, library lookaside (LLA) is started. LLA manages the LNKLST data sets and can be used to control updates to them. For more information about SYS1.LINKLIB and other data set concatenations, see:

- “Concatenating data sets to the LNKLST concatenation” on page 703
- “APF authorization for LNKLST data sets” on page 704
- “Cataloging LNKLST data sets” on page 704
- “Modifying the contents of LNKLST data sets” on page 705.

Parameter in IEASYSxx (or supplied by the operator)

```
LNK={aa             }
    {(aa,bb,...[,L])}
    {(,L)           }
```

The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so forth.), are appended to LNKLST to identify one or more LNKLSTxx members of parmlib.

If the L option is specified, the names of the data sets that are concatenated to SYS1.LINKLIB are displayed at the operator's console as the data sets are opened. The list is preceded by message: `IEA331I LINK LIBRARY CONCATENATION SYS1.LINKLIB`

Syntax rules for LNKLSTxx

The following rules apply to the creation of LNKLSTxx:

- On each record, use commas to separate the names of data sets.
- To indicate that a record is to be continued, place a comma followed by at least one blank after the last data set name on a record.
- If a data set is cataloged in a user catalog, but not in the system master catalog, you must also specify the volume serial (VOLSER) of the volume on which the data set resides. Specify the VOLSER in parentheses, immediately after the data set name.
- If you do not use the corresponding SYSLIB statement in PROGxx and you specify SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, or SYS1.SIEAMIGE in LNKLSTxx (perhaps to change the system's default processing), you must also specify the VOLSER of the volume on which the data set resides. Otherwise, the system ignores the specification.
- On a line, data entered after the last data set name and the optional comma continuation character is treated as a comment and ignored.
- Data records entered after the last data line are ignored as if they are comments.

Syntax format of LNKLSTxx

```
IEASYSxx:  ...,LNK=(nn,nn,nn,...)
LNKLSTxx: {data-set-name        },{data-set-name        },...
          {data-set-name(volser)} {data-set-name(volser)}
```

Syntax example of LNKLSTxx

```
IEASYSxx:   ...,LNK=(00,01,02,03)
LNKLST00:   SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB,
LNKLST01:   SYS1.LINKLIB,DBLUE.U30LIB(U30PAK),SYS2.U30LIB
LNKLST02:   SYS1.AUXLIB,SYS1.JES3
LNKLST03:   SYS1.TEST
```

As a result of these specifications, the following data sets, in the order specified, are concatenated to SYS1.LINKLIB (after SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE and SYS1.SIEAMIGE):

- SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB,DBLUE.U30LIB,
- SYS2.U30LIB,SYS1.AUXLIB,SYS1.JES3,SYS1.TEST

In the LNKLST01 parmlib member in this example, note the following:

- The specification of SYS1.LINKLIB is ignored.
- DBLUE.U30LIB is a user-cataloged data set on VOLSER U30PAK.

IBM-supplied default for LNKLSTxx

IBM does not supply a default LNKLSTxx member. By default, the system concatenates data sets SYS1.MIGLIB and SYS1.CSSLIB to SYS1.LINKLIB.

IBM-supplied sample for LNKLSTxx

IBM provides a sample LNKLSTxx member in SYS1.SAMPLIB. The sample member is named LNKLST00.

Chapter 68. LOADxx (system configuration data sets)

The LOADxx member specifies:

- Information about your I/O configuration.
- An alternate nucleus ID.
- The architecture level of the nucleus.
- The NUCLSTxx member that you use to add and delete modules from the nucleus region at IPL-time.
- Information about the master catalog.
- Information about the parmlib concatenation.
- The name of the sysplex (systems complex) that a system is participating in; it is also the substitution text for the &SYSPLEX system symbol.
- The IEASYMxx and IEASYSxx parmlib members that the system is to use.
- Additional parmlib data sets that the system will use to IPL. These data sets are concatenated ahead of SYS1.PARMLIB to make up the parmlib concatenation.
- Filtering keywords so you can use a single LOADxx member to define IPL parameters for multiple systems. The initial values of the filter keywords (HWNAME, LPARNAME and VMUSERID) are set at IPL to match the actual values of the system that is being IPLed. The LOADxx member can be segmented by these keywords.
- Facilities that are not to be used at this time because migration to another processor, z/OS release, or both is underway.

The LOADxx member is selected through the use of the LOAD parameter on the system control (SYSCTL) frame of the system console. For information about specifying the LOAD parameter, see *z/OS MVS System Commands*. If the operator does not select a LOADxx member on the system console, the system uses LOAD00.

Note: The system must have access to a LOADxx member.

Placement of LOADxx

You can place the LOADxx member in one of the following system data sets:

- SYSn.IPLPARM
- SYS1.PARMLIB

Consider placing LOADxx in the SYSn.IPLPARM data set. During IPL, the system looks for LOADxx in the following order:

1. SYS0.IPLPARM through SYS9.IPLPARM on the IODF volume.
2. SYS1.PARMLIB on the IODF volume.
3. SYS1.PARMLIB on the sysres volume.

Note: The IODF volume must be in the same subchannel set as the IPL device.

Do not create a SYSn.IPLPARM data set unless it contains the LOADxx member that is used to configure your system. When the system finds either SYSn.IPLPARM or SYS1.PARMLIB, it expects to find a LOADxx member in the data set. The search will stop with the first SYSn.IPLPARM or SYS1.PARMLIB data

set found. The system will use the LOADxx it finds within that data set as designated by the LOAD parameter. If the LOADxx member specified on the LOAD parameter is not in the data set, the system loads a wait state.

Note: Because the system data sets are searched in ordered sequence, use SYS0.IPLPARM for best IPL performance.

Copying LOADxx members

To copy LOADxx members to a backup volume, use either DFDSS (for SMS-managed volumes) or IEBCOPY. To ensure that the system will associate the copied LOADxx member with the proper IODF on a SYSn.IPLPARM volume, do not specify the IODF prefix in the LOADxx member. The system then will default the prefix of the IODF data set to the prefix of the SYSn.IPLPARM data set that contains the LOADxx member.

Filtering with LOADxx

The LOADxx filter keywords HWNAME, LPARNAME, and VMUSERID allow you to use a single LOADxx member to define IPL parameters for multiple systems. At IPL, the initial values of the keywords are set to match the actual values of the system being IPLed. The filter keywords are used to optionally change the IPL parameters.

Note: If a filter keyword is not applicable to a particular system, it is initialized to blanks at IPL. Subsequent specification of this filter keyword within LOADxx resets it.

The keywords control filtering in a hierarchy with HWNAME on the top and VMUSERID on the bottom. Figure 30 on page 621 shows this hierarchical relationship. The HWNAME parameter is used to specify the Control Processing Complex (CPC) name. HWNAME also sets LPARNAME and VMUSERID to their default values. The LPARNAME parameter, the next level in the hierarchy, is used to set the Logical Partition name. LPARNAME also sets VMUSERID to the default value. The value of HWNAME is unchanged. The lowest level of the hierarchy, the VMUSERID parameter, specifies the user ID of a z/VM system under which a z/OS image is running as a guest. Due to the hierarchical relationship of these keywords, the following rules apply to the specifications in your LOADxx member:

1. Keyword HWNAME is in effect from its occurrence to the next HWNAME statement in the member or to the end of the LOADxx member.
2. Keyword LPARNAME is in effect from its occurrence to the next HWNAME or LPARNAME statement or to the end of the LOADxx member.
3. Keyword VMUSERID is in effect from its occurrence to the next HWNAME, LPARNAME, or VMUSERID statement or to the end of the LOADxx member.

There is no way to explicitly indicate that you want the LPARNAME parameter or the VMUSERID parameter reset to their default values. If the system that is being IPLed is not running as LPAR mode or on VM, LPARNAME and VMUSERID might not be meaningful. This is a case when you might want to reset these values. As previously discussed, specifying the HWNAME parameter resets the LPARNAME and VMUSERID parameters to their default values. You can also reset just the VMUSERID parameter to its default value by specifying the LPARNAME parameter.

See “Filtering example” for examples of how to set the keywords.

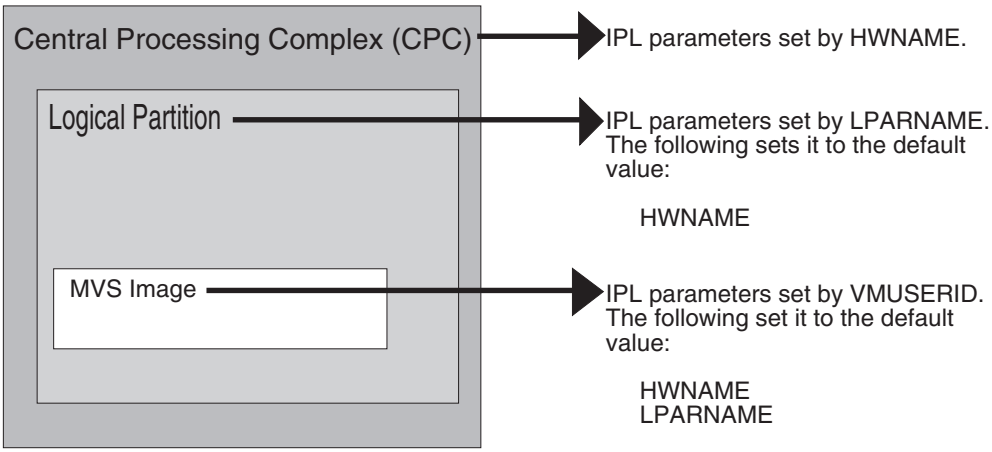

Figure 30. LOADxx filtering hierarchy

When another LOADxx statement (such as SYSCAT or IODF) is specified, the current HWNAME, LPARNAME, and VMUSERID filters are compared to the actual system values. If they all match, that LOADxx statement is accepted as applicable to the system that is being IPLed. See “Statements/parameters for LOADxx” on page 625 for specifics about each keyword.

Filtering example

Figure 31 illustrates uses of the HWNAME, VMUSERID, and LPARNAME filter parameters to segment LOADxx statements for various configurations. The contents of the LOADxx member are shown and then the IPL results are given.

```
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
NUCLEUS  1
NUCLST   00

*This segment applies to any machine in any logical partition (or
*not in LPAR mode).

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPARM  01
PARMLIB  ALLSYS.PARMLIB

*This segment applies to an IPL on VM user ID V1 on any LPAR (or not
*in LPAR mode), and on any hardware.
*Note that at this point neither HWNAME
*nor LPARNAME have been specified in the LOADxx member, so they
*will automatically match those of the system being IPLed.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
VMUSERID V1
IEASYM   V1
SYSPARM  V1

*This segment applies to any IPL on VM user ID V2 on any LPAR (or not
*in LPAR mode), and on any hardware.
*Note that at this point neither HWNAME
*nor LPARNAME have been specified in the LOADxx member, so they
*will automatically match those of the system being IPLed.
```

Figure 31. Example: using filter parameters to segment LOADxx statements (Part 1 of 2)

```
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
VMUSERID V2
IEASYM   V2
SYSPARM  V2
PARMLIB  V2.PARMLIB

*This segment applies to an IPL in logical partition L1 on any hardware
*and not under any VM user ID (or not under VM).
*Note that at this point HWNAME has not been specified in the LOADxx member,
*so it automatically matches the system being IPLed.
*Also note that specification of LPARNAME
*causes the VMUSERID filter to be reset to match the system being IPLed.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
LPARNAME L1
IEASYM   L1
SYSPARM  L1

*This segment applies to an IPL on machine H1 in any logical partition
*(or not in LPAR mode) and under any VM user ID (or not under VM).
*Note that specification of HWNAME
*causes the LPARNAME and VMUSERID filters to be reset to match the
*system being IPLed.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
LPARNAME L2
IEASYM   BB
PARMLIB  L2.PARMLIB

*This segment applies to an IPL on machine H1 in logical partition L2
*not running under VM.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
VMUSERID
IEASYM   CC
PARMLIB  NOT.VM.PARMLIB

*This segment applies to an IPL on machine H1 running in non-LPAR mode
*under any VM user ID (or not under VM).  The specification of
*LPARNAME causes the VMUSERID filter to be reset to match the system
*being IPLed.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
LPARNAME
IEASYM   DD
PARMLIB  NOT.LPAR.PARMLIB

*This segment applies to an IPL on machine H2 in any logical partition
*(or not in LPAR mode), under any VM user ID (or not under VM).
*The specification of HWNAME causes the LPARNAME
*and VMUSERID filters to be reset.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
HWNAME   H2
SYSPARM  H2
PARMLIB  H2.PARMLIB

*This is only *one* LOADxx.

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
HWNAME   H1
SYSPARM  H1
PARMLIB  H1.PARMLIB

*This segment applies to an IPL on machine H1 in logical partition L2
*not under any VM user ID (or not under VM).
```

Figure 32. Example: using filter parameters to segment LOADxx statements (Part 2 of 2)

Table 29 summarizes some of the IPL results produced by the examples in Figure 31 on page 621. Note that use of a second PARMLIB statement adds to the list of parmlibs. For any other statement, the previous value is replaced.

Table 29. IPL Results

IPL	LOADxx Members
IPL on hardware H1 with logical partition L0 not under z/VM.	SYSPARM H1 IEASYM not used PARMLIB ALLSYS.PARMLIB PARMLIB H1.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H1 with logical partition L1 under VMUSERID V1.	SYSPARM H1 IEASYM L1 PARMLIB ALLSYS.PARMLIB PARMLIB H1.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H1 with logical partition L2 under VMUSERID V0.	SYSPARM H1 IEASYM BB PARMLIB ALLSYS.PARMLIB PARMLIB H1.PARMLIB PARMLIB L2.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H1 with logical partition L2 not under z/VM.	SYSPARM H1 IEASYM CC PARMLIB ALLSYS.PARMLIB PARMLIB H1.PARMLIB PARMLIB L2.PARMLIB PARMLIB NOT.VM.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H1, not in LPAR mode, and not under VMUSERID V2.	SYSPARM H1 IEASYM DD PARMLIB ALLSYS.PARMLIB PARMLIB H1.PARMLIB PARMLIB NOT.LPAR.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H2, not in LPAR mode, and not under VMUSERID V1 or V2.	SYSPARM H2 IEASYM not used PARMLIB ALLSYS.PARMLIB PARMLIB H2.PARMLIB PARMLIB SYS1.PARMLIB
IPL on hardware H2 under VMUSERID V2 in logical partition L0.	SYSPARM H2 IEASYM V2 PARMLIB ALLSYS.PARMLIB PARMLIB V2.PARMLIB PARMLIB H2.PARMLIB PARMLIB SYS1.PARMLIB
IPL on unnamed hardware with logical partition L1 not under VMUSERID V1 or V2.	SYSPARM L1 IEASYM L1 PARMLIB ALLSYS.PARMLIB PARMLIB SYS1.PARMLIB

Parameter in IEASYSxx (or supplied by the operator)

None.

Support for system symbols

You can use LOADxx to define substitution text for the &SYSPLEX system symbol and point to other parmlib members that define system symbols. However, do not use those system symbols or any others in LOADxx. The system must complete processing of LOADxx to define substitution texts to the system symbols. Therefore, the system might not substitute text for system symbols that you use in LOADxx.

Syntax rules for LOADxx

The following syntax rules apply to LOADxx:

- Each record consists of 80 columns, although columns 73 through 80 are ignored.
- The fields are column-dependent as shown in "Statements/parameters for LOADxx" on page 625. Columns not shown to contain data must contain blanks. All data must be left-justified within the column ranges.
- Lines that begin with an asterisk in column 1 are comments.
- Blank lines are ignored.

Syntax format of LOADxx

LOADxx is a column-dependent parmlib member. An asterisk in column 1 denotes a comment. Parameters begin in column 1. Data begins in column 10. Columns 73-80 are ignored.

In the following syntax diagram, column grids are included as comments to help you place parameters and data in the correct columns.

```
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7
HWNAME   h1
LPARNAME l1
VMUSERID v1

*---+----1----+----2----+----3----+----4----+----5----+----6----+----7
ARCHLVL  a
DYNCPADD { nnnn | ENABLE | DISABLE}

IEASYM  [xx]
        [(xx,yy,zz,...,L)]
INITSQA  xxxxK yyyyK
         xxxxM yyyyM
IODF     xx hiqualif configid id y s
MACHMIG  x1,x2,...,xn
NUCLEUS  n
NUCLST   nn  y
PARMLIB  dsn                                          [volid]
                                                      [******]
                                                      [*MCAT*]
SYSCAT   volserxycsdsname                              hlqtvc
SYSPARM [xx]
        [(xx,yy,zz,...,L)]
SYSPLEX  plexname
```

IBM-supplied default for LOADxx

Although IBM provides no default member in SYS1.PARMLIB, you can create a sample LOADxx member using the JCL in the IPXLOADX member of SYS1.SAMPLIB. You can also use the SPPINST member of SYS1.SAMPLIB to create, update or list LOADxx members through an ISPF application.

For detailed information about setting up and using these tools, see the prolog of the IPXLOADX and Appendix B, “Symbolic Parmlib Parser,” on page 795 at the end of this information.

Statements/parameters for LOADxx

ARCHLVL

The ARCHLVL statement specifies the mode in which the operating system will run. In z/OS, the processor determines the appropriate z/OS architecture mode, and you do not need to specify the ARCHLVL parameter. It is recommended that you do not specify the ARCHLVL statement.

Note: Specifying the ARCHLVL parameter explicitly identifies the nucleus extension, IEANUCax, for the architecture level of your system. The system will use this extension along with the common base, IEANUC0x, to build the nucleus.

Column	Contents
1-7	ARCHLVL
10	A one-digit value to specify the architecture level for the nucleus extension. **Default:** 2 (run in z/Architecture mode). **Note:** If you specify a value for ARCHLVL that is not the default for the processor, message IEA368I will be issued during the IPL process, and the value you specified will be ignored.

DYNCPADD

Indicates if z/OS support for dynamic CPU/core addition is enabled.

Column	Contents
1-8	DYNCPADD

Column	Contents
10-16	{nnnn | ENABLE | DISABLE}

nnnn specifies if dynamic CPU/core addition functionality should be enabled such that *nnnn* CPUs/cores can be dynamically added to the image for the life of the IPL. *nnnn* must be a left-justified value without any leading zeros. Specifying 8 or 16 is acceptable syntax for *nnnn*; 16 is the default.

The number of CPUs/cores that can be dynamically added will be the minimum between:

- *nnnn* plus the highest CPU/core ID defined at IPL
- Highest CPU/core ID that the machine supports
- Highest CPU/core ID the z/OS release supports

If you specify *nnnn*, choose a value that is large enough to be able to dynamically add more CPUs/cores than you could envision for the life of the IPL, then add some extra and set that value in *nnnn*.

DISABLE
Specifies if dynamic CPU/core addition functionality should be disabled. If either the hardware or the z/OS release do not support dynamic CPU/core addition, dynamic CPU/core addition will be disabled.

ENABLE
Specifies if dynamic CPU/core addition functionality should be enabled. The hardware and the z/OS release must support dynamic CPU/core addition to be able to dynamically add a CPU/core after IPL. When you specify ENABLE, z/OS prepares to dynamically add the smaller of the following:

- Highest CPU/core ID the hardware supports
- Highest CPU/core ID the z/OS release supports.

IBM recommends specifying *nnnn* to avoid allocating CPU/core-related storage for CPUs/cores which will never be dynamically added for the life of the IPL.

Default: 16

Note: When PROCVIEW CPU is in effect, DYNCPADD applies to CPUs. When PROCVIEW CORE is in effect, DYNCPADD applies to cores.

HWNAME
Specifies the name of a central processor complex (CPC), as defined to hardware configuration definition (HCD). The HWNAME parameter is used as a filter to define value parameters for a specified processor.

This optional filter parameter identifies a segment of LOADxx that may contain LOADxx statements that will be used if the specified HWNAME matches the name of the processor where the LOADxx statement is running. When HWNAME is specified, it resets the LPARNAME and VMUSERID to their default values.

Note: Before the specification of this keyword, all statements up to this keyword pertain to any CPC. This keyword cannot be reset, and all subsequent statements pertain to the last specified HWNAME.

Column	Contents
1-6	HWNAME
10-17	A required hardware processor name as defined in the IODF. There is no default value.

IEASYM

The IEASYM statement identifies one or more suffixes of IEASYMxx members of SYS1.PARMLIB that take the following steps for one or more systems:

- Define static system symbols.
- Specify the IEASYSxx parmlib members that the system is to use.

To specify *one* IEASYMxx parmlib member, code IEASYM as follows:

Column	Contents
1-6	IEASYM
10-11	A 2-character suffix appended to "IEASYM" to select the member.

The following example shows an IEASYM statement that tells the system to use the IEASYM01 parmlib member:

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
IEASYM   01
*
```

To specify one IEASYMxx parmlib member and display a list of member names in message IEA009I, code IEASYM as follows:

Column	Contents
1-6	IEASYM
10	A left parenthesis.
11-12	A 2-character suffix appended to "IEASYM" to select the member.
13-14	The following characters: ,L
15	A right parenthesis.

The following example shows an IEASYM statement that tells the system to use the IEASYM01 parmlib member and display the names of those members in message IEA009I.

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
IEASYM   (01,L)
*
```

To specify *more than one* IEASYMxx parmlib member, code IEASYM as follows:

Column	Contents
1-6	IEASYM
10	A left parenthesis.
11-71	A list of 2-character suffixes appended to IEASYM to select the members. To display a list of member names in message IEA009I, specify ,L anywhere after the first suffix and enclose the values in parentheses, as shown in the example below.

The following example shows an IEASYM statement that tells the system to use the IEASYM01 and IEASYM02 parmlib members and display the names of those members in message IEA009I.

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
IEASYM   (01,02,L)
*
```

If you specify a list of IEASYMxx suffixes, the system processes them from left to right. The system uses the definitions in all IEASYMxx members for which suffixes are specified. If the system finds duplicate definitions, the last definition overrides any previous definitions.

Default: None. If you do not specify an IEASYM statement, the system does not process IEASYMxx during initialization.

INITSQA

The INITSQA statement allows additional SQA and ESQA storage to be reserved for IPL and NIP processing when the default amount is not sufficient. Note that increasing the minimum below the line SQA allocation must be done with caution to ensure that it does not affect the lower boundary of CSA storage. (See "SQA" on page 486 for more information about specifying SQA storage.) If a syntax error exists, the entire statement is ignored and message IEA368I is issued.

Column	Contents
1-7	INITSQA
10-14	The amount of SQA to be added to the default initial amount. If blank, no SQA is added. If non-blank, a 4-digit numeric value followed by a multiplier of K or M must be specified. Value Range: 0000 K to 2048 K or 0000 M to 0002 M **Default**: 0000 K
16-20	The amount of ESQA to be added to the default initial amount. If blank, no ESQA is added. If non-blank, a 4-digit numeric value followed by a multiplier of K or M must be specified. Value Range: 0000 K to 8192 K or 0000 M to 0016 M **Default**: 0000 K

IODF

The IODF statement identifies the I/O definition file that contains information about the I/O configuration defined to your system through the hardware configuration definition (HCD). To change the I/O configuration definition when the system is running, use the activate process. For information about the ACTIVATE command, see *z/OS MVS System Commands*. For information about using the HCD to change the I/O configuration definition, see *z/OS HCD User's Guide*.

Column	Contents
1-4	IODF

Column	Contents
10-11	IODF suffix. This suffix is appended to nnnnnnnn.IODF to form the name of the nnnnnnnn.IODFxx data set. **Value Range**: A two-digit hexadecimal number (X'00-FF'), asterisks ('**'), pluses ('++'), minuses ('--'), or equals ('==').

If you used HCD to create your I/O Configuration data set (IOCDS), you can specify asterisks, pluses, minuses, or equals for the IODF suffix. If you specify asterisks, pluses, minuses, or equals, z/OS uses the IODF name found in the hardware configuration token. This IODF represents the last I/O configuration in the channel subsystem which you activated during the last Power® On Reset (POR) or dynamic I/O configuration change. If you override descriptor fields 1 and 2 in the HCD panel with an invalid IODF name, z/OS treats IODF suffixes '**' and '++' as if no IODF suffix was specified in LOADxx (z/OS searches from X'00' to X'FF' for a production IODF). If an invalid IODF name is found and you specified '--', z/OS searches for a production IODF starting with a suffix of X'FF'and searches backwards to X'00'. If an invalid IODF name is found and '==' were specified, z/OS loads wait state X'0B1' reason code X'00B'.

If pluses ('++') are specified and a valid IODF name is found in the hardware token, but is not found in the search or cannot be used, IODF selection is made in the following order:

- Search for the next available IODF starting with the current IODF suffix number plus one and search up to X'FF'. If none is found, the search continues from X'00' up to the current IODF number minus one.
- Select the first IODF that has both a processor definition in the IODF that matches the currently active hardware definition, and a matching operating system configuration identifier.
- If no matching IODF token is found, select the first IODF that has a matching operating system configuration identifier.
- If no matching operating system configuration identifier is found, a wait state is loaded.

Column	Contents
	If minuses ('--') are specified and a valid IODF name is found in the hardware token, but is not found in the search or cannot be used, IODF selection is made in the following order: • Search for the next available IODF starting with the current IODF suffix number minus one and search down to X'00' If none is found, the search continues from X'FF' down to the current number plus one. • Select the first IODF that has both a processor definition in the IODF that matches the currently active hardware definition, and a matching operating system configuration identifier. • If no matching IODF token is found, select the first IODF that has a matching operating system configuration identifier. • If no matching operating system configuration identifier is found, a wait state is loaded. If equals ('==') are specified and a valid IODF name is found in the hardware token, but is not found in the search or cannot be used, the result is the same as if you specified a specific number for the IODF suffix (hilevqu.IODF01). **Default**: If not specified, all possible IODFs (00-FF) are searched. The IODF selection is made in the following order: • If no matching IODF token is found, select the first IODF that has a matching operating system configuration identifier. • If no matching operating system configuration identifier is found, a wait state is loaded. **Attention:** If you specify asterisks ('**'), pluses ('++'), minuses ('--'), or equals ('==') for an IODF that does not reside on the IODF volume, or if you omit the IODF parameter altogether, the resulting search can substantially increase the time it takes to IPL the system, especially if the IODF volume has a large VTOC. In this situation, the system enters a X'073' wait state, indicating that the IPL is waiting for an I/O interrupt. To correct the problem, either specify the correct IODF suffix in LOADxx, or move the required IODF to the IODF volume, then reIPL.
13-20	High-level qualifier for the IODF data set name. This qualifier is added to IODFxx to form the name of the nnnnnnnn.IODFxx data set. If equals ('========') are specified, z/OS attempts to extract the high-level qualifier from the hardware configuration token. If the token is not available, a wait state is loaded. Otherwise, z/OS uses this high-level qualifier for the IODF name. If the first character of the high-level qualifier found in the hardware configuration token is blanks (' '), a wait state is loaded. **Value Range**: 1 to 8 alphameric characters or "========". **Default**: If LOADxx resides in a SYSn.IPLPARM data set, and there is no high-level qualifier specified, the high-level qualifier defaults to the high-level qualifier of the SYSn.IPLPARM data set. If LOADxx resides in a SYS1.PARMLIB data set, and there is no high-level qualifier specified, a wait state is loaded.

Column	Contents
22-29	Operating system configuration identifier. This eight-character identifier is used to select the appropriate configuration from those configurations defined in the IODF. For a list of eligible operating system configurations, select the "Define Operating System Configurations" option on the primary HCD panel. **Default**: If there is only one operating system configuration identifier in nnnnnnnn.IODFxx, then that one will be used. If there is more than one identifier and the identifier is not specified, a wait state is loaded.
31-32	Eligible device table identifier **Default**: 00
34	To indicate that the system should load all of the device support modules for the devices defined in the IODF and all the device types that support dynamic processing, specify either Y or blank. To indicate that the system should load only the modules required for the devices defined in your IODF, specify any non-blank character other than Y. **Default**: Y

Column	Contents
36	Subchannel set indicator. Indicates the subchannel set IOS uses for normal base devices that have a special secondary device with the same address. The following values can be specified: **0** Indicates the normal base devices in subchannel set 0 are used for the IPL. **n** Indicates the special secondary devices in this subchannel set are used for the IPL. ***** Indicates the subchannel set of the IPL device is used for the IPL. If you are using a product to manage disk replication with HyperSwap capability (for example, GDPS or TPC-R), it is possible to have secondary devices attached to a subchannel set other than 0. Using such a configuration creates "special" pairs of devices consisting of one device in subchannel set 0 and a second device in an alternate subchannel set. Attaching secondary devices in an alternate subchannel set allows you to better utilize subchannel set 0 for devices that must be online and available for allocation. You must provide direction when an IPL is required so that z/OS brings the correct devices online when HyperSwap occurs (that is, when secondary devices become the devices in use, or in other words, the "active" subchannel set for a special pair of devices becomes the subchannel set to which the secondary devices are attached). On systems where special secondary devices are connected, if this value is not specified or is not valid (for example, not a 0, 1, 2, or *), the system will prompt the operator with message IEA111D to determine what subchannel set should be used. Specifying '*' allows z/OS to inherit the "active" subchannel set from the IPL devices (for IBM zEnterprise 196 (z196) or later processors). Specifying a number indicates that a specific subchannel set is to be used as the "active" subchannel set. Consult the documentation for your disk replication manager to determine whether special pairs are supported and if any restrictions exist before specifying this parameter. **Default**: None

LPARNAME

Specifies the name of a logical partition that is defined to a processor, which is one of the following:

- The partition name specified on the “Add Partition” panel in HCD (see *z/OS HCD User’s Guide* for more information).
- The partition name specified on the RESOURCE or CHPID statement that is input to the I/O configuration program (IOCP).

This optional filter parameter identifies a segment of LOADxx that may contain LOADxx statements that will be used if the specified HWNAME matches the name of the processor and the specified LPARNAME matches the actual LPAR logical partition in which z/OS is executing. When LPARNAME is specified, it resets VMUSERID to its default value. The LPARNAME parameter is used as a filter to define value parameters for a specified partition of a processor.

Note: If running under z/VM, you cannot filter using LPARNAME.

Column	Contents
1-8	LPARNAME
10-17	A required logical partition name as defined to LPAR. A blank entry indicates a z/OS image not running in LPAR mode. The default of matching the system being IPLed is set indirectly by specifying the HWNAME parameter.

MACHMIG

Identifies one or more facilities that you do not want z/OS to use at this time because migration to another processor, z/OS release, or both is underway. Code the MACHMIG statement as follows:

Column	Contents
1-7	MACHMIG
10-72	A list of facilities not to use. When more than one facility is listed, separate each from the previous by one or more blanks or commas. The following facilities can be specified in upper, lower, or mixed case: • EDAT2 (the hardware-based enhanced-DAT facility 2) • TX (the hardware-based transactional-execution facility) • RI (a hardware-based facility that is reserved for IBM use only(• VEF (the hardware-based vector extension facility)

Example: The following example shows a MACHMIG statement that tells the system not to use the transactional execution facility and the enhanced DAT facility 2.

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
MACHMIG  TX,EDAT2
*
```

Default: None. If you do not specify a MACHMIG statement, the system does not limit its exploitation of machine facilities.

MTLSHARE

The MTLSHARE statement enables a full-support MTL system to treat manual tape library defined devices as stand-alone devices when the Y(es) parameter is specified. Specifying Y also implies that the cartridge loader on MTL defined devices should not be indexed. For complete details please see *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries*.

Column	Contents
1-8	MTLSHARE
10	Y indicates that the system is to run in its coexistence mode, that is, treat MTL defined tape drives as stand-alone drives, and do not index the cartridge loaders on such drives. N indicates that the system is to run in its full function mode. MTL defined drives are treated as MTL resident drives, and cartridge loaders on such drives are indexed per MTL rules.

Default: N

NUCLEUS

The NUCLEUS statement identifies the IEANUC0x member of SYS1.NUCLEUS

that your system is to use. If the operator specified a nucleus identifier on the LOAD parameter, z/OS ignores the NUCLEUS statement.

Column	Contents
1-7	NUCLEUS
10	A one-digit suffix that is appended to "IEANUC0" to select a member of SYS1.NUCLEUS. **Default:** 1 **Note:** When you specify an alternate nucleus, the proper architectural extension of the nucleus must also exist. See page ARCHLVL for information about architectural extensions to the nucleus.

NUCLST

The NUCLST statement:

- Identifies the NUCLSTxx parmlib member that your system is to use.
- Specifies whether the system is to load a wait state if any of the INCLUDE statements in the NUCLSTxx member specify a member that cannot be found in SYS1.NUCLEUS.

The NUCLSTxx member must reside in the same data set as the LOADxx member. For information about coding the NUCLSTxx member, see Chapter 74, "NUCLSTxx (customizing the nucleus region)," on page 689.

Column	Contents
1-6	NUCLST
10-11	A two-character suffix appended to "NUCLST" to select a NUCLST member. **Default**: None
13	The character 'Y' indicates that a wait state is to be loaded if any of the INCLUDE statements in the NUCLSTxx member specify a member that cannot be found in SYS1.NUCLEUS. If any other character is specified, the system does not load a wait state. **Note:** Uppercase 'Y' is required. Lowercase 'y' indicates that a wait state is not to be loaded. **Default**: Do not load a wait state.

PARMLIB

Specifies a data set that will be included in the logical parmlib concatenation. The parmlib concatenation consists of up to 16 PARMLIB data sets and SYS1.PARMLIB. When there is more than one PARMLIB statement, the statements are concatenated and SYS1.PARMLIB, as cataloged in the Master Catalog, is added at the end of the concatenation, unless it was specified in a parmlib. See "Restrictions" on page 61 for restrictions on cataloging parmlib data sets when a system symbol is used. The parmlib concatenation is established during IPL and is used by Master Scheduler Initialization. Programs can access members in the logical parmlib concatenation using the IEFPRMLB macro (see *z/OS MVS Programming: Assembler Services Reference IAR-XCT*). Each additional PARMLIB statement adds another data set to the logical parmlib concatenation.

Column	Contents
1-7	PARMLIB
10-53	A required valid data set name. There is no default value.

Column	Contents
55-60	A optional valid volume name. If a volume name is specified, IPL processing attempts to locate the specified data set on the specified volume. If '******' or '&SYSR1' is specified, IPL processing attempts to locate the specified data set on the system residence volume. If '*MCAT*' is specified, IPL processing attempts to locate the specified data set on the master catalog volume. If nothing is specified, IPL processing attempts to locate the specified data set first in the master catalog and, if it is not located there, on the system residence volume.

Note: &SYSR1 is the only system symbol that can be specified in the logical parmlib concatenation.

Default: If you do not specify at least 1 PARMLIB statement, the parmlib concatenation will consist of only SYS1.PARMLIB and Master Scheduler processing will use the IEFPARM DD statement, if there is one in the Master JCL. If there are no parmlib statements in the parmlib concatenation and there is no IEFPARM DD statement, Master Scheduler processing will use SYS1.PARMLIB.

Example 1: The following example shows the definition of a parmlib concatenation consisting of MYDSN1.PARMLIB, MYDSN2.PARMLIB, MYDSN3.PARMLIB. SYS1.PARMLIB, as cataloged in the master catalog, will automatically be concatenated as the last data set in the parmlib concatenation.

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
PARMLIB  MYDSN1.PARMLIB
PARMLIB  MYDSN2.PARMLIB                              VOL123
PARMLIB  MYDSN3.PARMLIB                              VOL456
```

Example 2: The following example shows the definition of a parmlib concatenation consisting of MYDSN1.PARMLIB, MYDSN2.PARMLIB, SYS1.PARMLIB on volume VOL234, MYDSN3.PARMLIB and, additionally, SYS1.PARMLIB, as cataloged in the master catalog, which will be concatenated as the last data set in the parmlib concatenation.

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
PARMLIB  MYDSN1.PARMLIB
PARMLIB  MYDSN2.PARMLIB                              VOL123
PARMLIB  SYS1.PARMLIB                                VOL234
PARMLIB  MYDSN3.PARMLIB                              VOL456
```

Example 3: The following example shows the definition of a parmlib concatenation consisting of MYDSN1.PARMLIB, MYDSN2.PARMLIB, SYS1.PARMLIB as cataloged in the master catalog and MYDSN3.PARMLIB. SYS1.PARMLIB is not concatenated as the last data set in this case since it was already specified on a PARMLIB statement without a volume serial number and is, thus, the SYS1.PARMLIB cataloged in the master catalog.

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
PARMLIB  MYDSN1.PARMLIB
PARMLIB  MYDSN2.PARMLIB                              VOL123
PARMLIB  SYS1.PARMLIB
PARMLIB  MYDSN3.PARMLIB                              VOL456
```

In all of the above examples, if none of the data sets specified on the PARMLIB statements could be located, the parmlib concatenation would consist of only SYS1.PARMLIB.

PROCVIEW

Indicates the processor view for the duration of the IPL.

Column	Contents
1-8	PROCVIEW
10-21	{CORE | CPU | CORE,CPU_OK}

CORE Specifies z/OS should configure a processor view of core, where a core can have 1-n threads. A thread is comparable to the definition of a CPU in a pre-multithreading environment. The number of useable threads per core is limited by what is supported by the underlying hardware, the current z/OS release, and the specification of MT_XXXX_MODE keywords. See the description of these keywords in "Statements/parameters for IEAOPTxx" on page 398. If the underlying hardware does not support MT, a core is limited to one thread.

This parameter:

- Results in LOADxx DYNCPADD specifying the number of cores that can be dynamically added. For more information, see the DYNCPADD parameter described previously in this section.
- Requires DISPLAY M=CORE to display the core state. For more information, see the description of the DISPLAY M=CORE command in *z/OS MVS System Commands*.
- Requires the CONFIG CORE command to configure an entire core. For more information, see the description of the CONFIG CORE command in *z/OS MVS System Commands*.
- Limits the use of the MODE command. For more information, see the description of the MODE command in *z/OS MVS System Commands*.

For purposes of migration and coexistence, you can specify PROCVIEW CORE on systems that do not specifically support multithreading. For more information on the MT Facility, see *z/Architecture Principles of Operations*.

CPU Specifies z/OS should configure a traditional processor view of CPU and not exploit the MT Facility.

CORE,CPU_OK
Specifies z/OS should manage a processor as a core; see the previous description of the CORE option. However with CORE,CPU_OK the CPU parameter is accepted as an alias for applicable commands.

Note: When PROCVIEW CORE or PROCVIEW CORE,CPU_OK is requested and the underlying hardware supports MT, partitions are forced to run with IEAOPTxx HIPERDISPATCH=YES and are unable to switch into HIPERDISPATCH=NO.

Default: CPU

SYSCAT

Identifies the master catalog. The operator can override the value specified on this parameter, using the LOAD parameter on the system console with an

appropriate initialization message suppression indicator (IMSI). For more information, see the topic on loading the system software in *z/OS MVS System Commands*.

Column	Contents
1-6	SYSCAT
10-15	The volume serial of the device that contains the master catalog..
16	The character "1", unless SYS% to SYS1 conversion is active, in which case this will be a "2".
17	Alias name level of qualification. **Value Range**: 1 - 4 **Default**: 1
18-19	CAS service task lower limit. **Value Range**: X'18' - X'B4' **Default**: X'3C' If you want to specify the CAS service task lower limit, specify the value with EBCDIC characters, for instance, hexadecimal B4 is specified as C'B4' or X'C2F4'.
20-63	The 44-byte data set name of the master catalog.
64-71	The 1 to 8 character high-level qualifier of the tape volume catalog. **Default**: SYS1
72	Specify Y to enable AUTOADD when the catalog address space (CAS) makes the first connection to the coupling facility. The AUTOADD function enables coupling facility support of enhanced catalog sharing (ECS) for eligible catalogs.

Default: If you do not specify a SYSCAT statement, the system prompts the operator to specify the SYSCATxx member of SYS1.NUCLEUS.

SYSPARM

The SYSPARM statement identifies one or more IEASYSxx members of the parmlib concatenation that the system is to use (in addition to IEASYS00). To display the contents of IEASYSxx at the operator console when the system processes each member, specify ,L anywhere after the first suffix and enclose the values in parentheses. For example, specify (01,L) on SYSPARM to tell the system to process IEASYS01 and display the contents of that member at the operator console. The system ignores the SYSPARM statement if the operator specifies on the LOAD parameter that the system should prompt for system information. The operator can accomplish this by specifying an A, P, S, or T IMSI character on the LOAD parameter on the system console. For details about IMSI characters, see the topic on loading the system software in *z/OS MVS System Commands*.

Note: The suffixes of IEASYSxx members can also be specified:

- On the SYSPARM parameter in the IEASYMxx parmlib member.
- By the operator, in response to message IEA101A SPECIFY SYSTEM PARAMETERS.

See "Step 2. Determine where to specify system parameters" on page 42 for more information.

During system initialization, NIP first processes the IEASYS00 parmlib member to establish parameters. Then it determines, from the suffixes that are specified on the SYSPARM statement in LOADxx, the SYSPARM parameter in IEASYMxx, or by the operator, which IEASYSxx members are to be used. See "Step 2. Determine where to specify system parameters" on page 42 for a description of how the system determines which IEASYSxx members are to be used.

To specify one IEASYSxx parmlib member, code SYSPARM as follows:

Column	Contents
1-7	SYSPARM
10-11	The volume serial of the device that contains the master catalog.

The following example shows a SYSPARM statement that tells the system to use the IEASYS01 parmlib member:

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPARM  01
*
```

To specify one IEASYSxx parmlib member and display the contents of IEASYSxx at the operator console, code SYSPARM as follows:

Column	Contents
1-7	SYSPARM
10	A left parenthesis.
11-12	A 2-character suffix appended to "IEASYS" to select the member.
13-14	The following characters: ,L
15	A right parenthesis.

The following example shows an SYSPARM statement that tells the system to use the IEASYS01 parmlib member and display the contents of IEASYSxx at the operator console.

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPARM  (01,L)
*
```

To specify more than one IEASYSxx parmlib member, code SYSPARM as follows:

Column	Contents
1-7	SYSPARM
10	A left parenthesis.
11-71	A list of 2-character suffixes appended to IEASYS to select members of SYS1.PARMLIB. To display the contents of IEASYSxx at the operator console when the system processes each member, specify ,L anywhere after the first suffix and enclose the values in parentheses, as shown in the example below.

Default: If you do not specify a SYSPARM statement, the system prompts the operator to specify the IEASYSxx members of SYS1.PARMLIB.

The following example tells the system to use IEASYSxx members IEASYS01 and IEASYS02 and display their contents:

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPARM  (01,02,L)
*
```

SYSPLEX

The SYSPLEX statement specifies the name of the sysplex in which the system participates. It is also the substitution text for the &SYSPLEX system symbol. Specify a sysplex name that is different from the names of all the systems that participate in the sysplex.

Note: The sysplex name must match the name that is specified in both of the following places:

- The SYSPLEX parameter of the XCF couple data set format utility. See *z/OS MVS Setting Up a Sysplex* for information about the data set format utility.
- The SYSPLEX parameter in the COUPLExx parmlib member. LOADxx defines the substitution text for &SYSPLEX early in system initialization so other parmlib members can use it. Therefore, if you plan to use the &SYSPLEX system symbol in parmlib, specify the sysplex name in LOADxx. To ensure that the name in COUPLExx matches the one in LOADxx, specify the &SYSPLEX system symbol on the SYSPLEX parameter in COUPLExx. (See Chapter 24, "COUPLExx (cross-system coupling facility (XCF) parameters)," on page 263for more information.)

Value range:

Column	Contents
1-7	SYSPLEX
10-17	The sysplex name. It can consist of 1 to 8 characters, left-justified in the column. Valid characters are alphanumeric (A-Z and 0-9) and national (@,#,$).

Default: If you do not specify a SYSPLEX statement, the system uses the value LOCAL until it processes the COUPLExx parmlib member; then it uses the name that is specified in COUPLExx. In this case, the system substitutes LOCAL for the &SYSPLEX system symbol in all parmlib members that are processed before COUPLExx.

The following example specifies that OURWORLD is the sysplex name.

```
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPLEX  OURWORLD
*
```

VMUSERID

Specifies the user ID of a z/VM system under which a z/OS image is running as a guest. This optional filter parameter identifies a segment of LOADxx that may contain LOADxx statements that will be used if the specified HWNAME matches the name of the processor, the specified LPARNAME matches the actual LPAR logical partition in which z/OS is executing, and the specified VMUSERID matches the actual user ID of the z/VM guest machine in which z/OS is executing.

Column	Contents
1-8	VMUSERID

Column	Contents
10-17	A required user ID name as defined to z/VM. A blank indicates an image not running under VM. The default of matching the system being IPLed is set indirectly by specifying the HWNAME or LPARNAME parameter.

Example of parmlib concatenation

The example in Figure 33 creates a parmlib concatenation with five data sets.

- The first PARMLIB statement indicates that data set dsn1 is included in the parmlib concatenation.
- The second PARMLIB statement indicates that data set dsn2, residing on volume vol2, is included in the parmlib concatenation.
- The third PARMLIB statement indicates that data set dsn3, residing on the sysres volume, is included in the parmlib concatenation.
- The fourth PARMLIB statement indicates that data set dsn4, residing on the master catalog volume, is included in the parmlib concatenation.
- SYS1.PARMLIB is automatically added to the bottom of the parmlib concatenation.

```
*
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
PARMLIB  dsn1
PARMLIB  dsn2                                          vol2
PARMLIB  dsn3                                          ******
PARMLIB  dsn4                                          *MCAT*
*
*
SYSCAT   volserxycsdsname
*
*  The first SYSPARM parameter specifies one IEASYSxx parmlib member.
*  The second SYSPARM parameter specifies multiple IEASYSxx parmlib
*  members.  xx, yy, and zz are suffixes for IEASYS.
*
SYSPARM  xx
SYSPARM  (xx[,yy,zz....,L])
*
*  In the SYSPLEX parameter, plexname is the sysplex name.
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
SYSPLEX  plexname
*
*
*  VMUSERID v1 specifies that the LOADxx statements that follow the
*  VMUSERID statement are used only if the vm user ID being used is v1.
*
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
VMUSERID v1
*
************************************************************************
```

Figure 33. Example of parmlib concatenation (LOADxx)

Chapter 69. LPALSTxx (LPA library list)

Use the LPALSTxx member to add your installation's read-only reenterable user programs to the pageable link pack area (PLPA). Placing programs in the PLPA allows them to be shared among users of the system.

Use one or more LPALSTxx members to concatenate your installation's program library data sets to SYS1.LPALIB. The system uses this concatenation, which is referred to as the LPALST concatenation, to build the PLPA.

The modules in the data sets that you define in LPALSTxx must meet the same requirements as the modules defined in SYS1.LPALIB. For example, these modules must be reentrant and executable.

During NIP, the system opens and concatenates each data set specified in LPALSTxx in the order in which the data set names are listed, starting with the first-specified LPALSTxx member.

The LPALST concatenation can have up to 255 extents. If you specify more data sets than the concatenation can contain, the system truncates the LPALST concatenation and issues messages that indicate which data sets were not included in the concatenation.

If one or more LPALSTxx members exist, and the system can open the specified data sets successfully, the system uses the LPALST concatenation to build the PLPA (during cold starts and IPLs that include the CLPA option). If a library named in LPALSTxx cannot be opened, message IEA712I is issued for that library and the system continues to build the LPA list concatenation with the other libraries named in LPALSTxx. If no LPALSTxx members exist, or none of the libraries named in LPALSTxx can be opened, the system builds the PLPA from only the modules named in SYS1.LPALIB.

The data sets in the LPALST do not have to be APF-authorized. However, any module in the pageable link pack area is treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected any data set in the LPALST to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

Data sets to be concatenated to SYS1.LPALIB must be cataloged in the system master catalog or a user catalog identified in the LPALSTxx member. (The system does not check OS CVOLs and only checks user catalogs when a data set name and pack VOLSER are listed in LPALSTxx.)

You cannot use PDSEs in the LPALST concatenation.

To identify which LPALSTxx members the system is to use, specify the member suffixes on the LPA parameter in IEASYSxx or as a parameter entered during IPL.

Parameter in IEASYSxx (or supplied by the operator)

```
LPA=     {aa            }
         {(aa,bb,...[,L])}
```

The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, and so forth), are appended to LPALST to form the name of the LPALSTxx parmlib member(s). If the L option is specified, the system displays (at the operator's console) the names of the data sets successfully concatenated to SYS1.LPALIB.

The LPA parameter is only effective during cold starts, or during IPLs in which you specify the CLPA option. The LPA parameter does not apply to modules requested through the MLPA option.

Syntax rules for LPALSTxx

The following rules apply to the creation of LPALSTxx:

- On each record, place a string of data set names separated by commas.
- If a data set is not cataloged in the system master catalog but is cataloged in a user catalog, specify in parentheses immediately following the data set name the one to six-character VOLSER of the pack on which the data set resides.
- Indicate continuation by placing a comma followed by at least one blank after the last data set name on a record.
- Be careful not to specify the same data set name more than once in the LPALSTxx members. This applies to data sets with and without a VOLSER specified. The same data set name is concatenated as many times as it appears in all specified LPALSTxx members. Specifying the data set name more than once can cause additional processing during IPL, when the CLPA processes.
- If a module exists in more than one library in the concatenation, the first occurrence of the module is placed in the PLPA. Later occurrences are ignored.
- The LPALIB data set is always the first data set in the concatenation (see "Using the SYSLIB statement" on page 699). Unless overridden by a SYSLIB statement in PROGxx, the LPALST concatenation begins with SYS1.LPALIB. If you do not use SYSLIB and place the LPALIB data set name on any record in any LPALSTxx member, the name is ignored.
- Lines that begin with an asterisk in column 1 are comments.
- On a line, data entered after the last data set name and the optional comma continuation character are treated as a comment and ignored.

Syntax format of LPALSTxx

```
IEASYSxx:  ...,LPA=(nn,nn,nn,...)
LPALSTxx: {data-set-name        },{data-set-name        },...
          {data-set-name(volser)} {data-set-name(volser)}
```

Syntax example of LPALSTxx

The following applies to both examples: DBLUE.U30LIB is a user-cataloged data set on VOLSER U30PAK. All other data sets are cataloged in the system master catalog.

```
IEASYSxx:   ...,LPA=(00,01,02,03)
LPALST00:   SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB
LPALST01:   SYS1.U30LIB,SYS2.U30LIB,DBLUE.U30LIB(U30PAK),SYS1.LPALIB
LPALST02:   SYS1.AUXLIB,SYS1.JES3
LPALST03:   SYS1.TEST
```

- **Example 1**:

  PROGxx does not include a SYSLIB LPALIB statement. The following data sets are concatenated to SYS1.LPALIB:

  – SYS1.CMDLIB
  – SYS1.TSORTNS
  – SYS1.BTAMLIB
  – SYS1.U30LIB
  – SYS2.U30LIB
  – DBLUE.U30LIB
  – SYS1.AUXLIB
  – SYS1.JES3
  – SYS1.TEST

  Because the SYSLIB statement with the LPALIB option was not used in PROGxx, the specification of SYS1.LPALIB (in LPALST01) is ignored.

- **Example 2**:

  PROGxx includes the following:

  ```
  SYSLIB LPALIB(SYS2.LPALIB)
  ```

  The following data sets are concatenated to SYS2.LPALIB:

  – SYS1.CMDLIB
  – SYS1.TSORTNS
  – SYS1.BTAMLIB
  – SYS1.U30LIB
  – SYS2.U30LIB
  – DBLUE.U30LIB
  – SYS1.LPALIB
  – SYS1.AUXLIB
  – SYS1.JES3
  – SYS1.TEST

IBM-supplied default for LPALSTxx

There is no default LPALSTxx member. If the installation does not create an LPALSTxx member (and therefore no data sets are concatenated to SYS1.LPALIB), the system uses only SYS1.LPALIB to build the PLPA.

Statements/parameters for LPALSTxx

Not applicable.

Chapter 70. MMSLSTxx (MVS message service list)

MMSLSTxx contains information that the MVS message service (MMS) uses to define the languages that are available for message translation at your installation. Through the use of the LANGUAGE, DEFAULTS, and EXIT statements, you:

- Identify the languages available into which U.S. English messages can be translated.
- Specify the default language into which U.S. English messages can be translated.
- Specify the installation exits that get control either before or after the translation occurs.

Selecting an MMSLSTxx member

You can select a particular MMSLSTxx parmlib member in one of the following ways:

- Specifying the MMS keyword on the INIT statement in the CONSOLxx parmlib member. CONSOLxx is used at system initialization.
- Issuing the SET MMS command after system initialization.

Two alphanumeric characters are appended to MMSLST to form the name of the MMSLSTxx parmlib member. After initialization, you can issue the SET MMS command to change the MMSLSTxx member; however, this change is temporary. At the next IPL, the system uses the MMSLSTxx member specified in the CONSOLxx. For information about CONSOLxx, see Chapter 23, "CONSOLxx (console configuration definition)," on page 231. For descriptions of the SET command, see *z/OS MVS System Commands*.

Parameter in IEASYSxx

None.

Sample MMSLSTxx member

IBM provides a sample MMS member, CNLLSTXX, in SYS1.SAMPLIB.

Syntax rules for MMSLSTxx

These rules apply to the creation of MMSLSTxx:

- Each record is 80 characters long. The system ignores columns 72 through 80 and leading blanks.
- Begin comments with /* in any column. It is best to end the comment with a */. If, however, you do not include the end delimiter, the system recognizes the next statement.
- Specify one DEFAULTS statement only.
- Specify one or more LANGUAGE statements including U.S. English.
- Optionally, specify one or two EXIT statements.

Syntax format of MMSLSTxx

```
DEFAULTS  LANGCODE(langcode)

LANGUAGE  LANGCODE(langcode) DSN(langdsn) CONFIG(membername) [NAME(langname)]

[EXIT  NUMBER(exitnum)  ROUTINE(exitname)]
```

Syntax example for MMSLSTxx

```
DEFAULTS LANGCODE(FRC)
LANGUAGE LANGCODE(ENU) DSN(SYS1.MSG.NLSMSENU) CONFIG(CNLENU01) NAME(AMERICAN)
LANGUAGE LANGCODE(ENG) DSN(SYS1.MSG.NLSMSENG) CONFIG(CNLENG01)
LANGUAGE LANGCODE(FRC) DSN(SYS1.MSG.NLSMSFRC) CONFIG(CNLFRC01)
         NAME(FRENCH) NAME(FRANCAIS)
LANGUAGE LANGCODE(ESP) DSN(SYS1.MSG.NLSMSESP) CONFIG(CNLESP01)
EXIT NUMBER(1) ROUTINE(NLSEXIT1)
EXIT NUMBER(2) ROUTINE(NLSEXIT2)
```

IBM-supplied default for MMSLSTxx

None.

Statements/parameters for MMSLSTxx

DEFAULTS

Specifies the default language your installation uses for translation. If MVS system messages or application-generated messages are available in this language, the messages will be displayed to the operator in this specified language.

LANGCODE(langname)

Specifies the three-character code for the default system language. You must code a LANGUAGE statement corresponding to the default language specified here. For a list of valid language codes see Table 30 on page 647.

LANGUAGE

Specifies information pertaining to all languages your installation uses for translation, including the language specified on the DEFAULTS statement. You must code one language statement per language supported at your installation.

LANGCODE(langcode)

Specifies the three-character code for the language whose message skeletons are contained in the run-time message file identified by DSN(langdsn).

At least one LANGUAGE statement must have a LANGCODE(langcode) of "ENU" for U.S. English. For a list of valid language codes see Table 30 on page 647.

DSN(langdsn)

Specifies the data set name of the run-time message file containing the message skeletons for the specified language. See *z/OS MVS Programming: Assembler Services Guide* for information about creating run-time message files.

CONFIG(membername)
Specifies the member of SYS1.PARMLIB that contains information about the date and time formats of messages translated into the specified language. The name of the member must be in the form of CNLcccxx, where:

ccc The appropriate three-character language code.

xx Any two characters that uniquely identify the member.

See Chapter 18, "CNLcccxx (time and date format for translated messages)," on page 205 for information about coding the CNLcccxx parmlib member.

NAME(langname)
Specifies a 1-24 byte name by which the installation can optionally refer to the language specified through the LANGCODE parameter. Language names must be unique within any one MMSLSTxx member. You can specify more than one NAME parameter on a LANGUAGE statement.

The language can be a quoted string containing mixed case characters as well as *shift-in* and *shift-out* characters to delineate double-byte character set (DBCS) characters.

EXIT
Specifies an installation exit that is to get control either before or after the translation occurs. See *z/OS MVS Installation Exits* for more information about the MMS pre-processing and post-processing exits.

NUMBER(exitnum)
Specifies the number of the exit to get control. The possible values are 1 and 2. Exit 1 gets control before message translation occurs and exit 2 gets control after translation.

ROUTINE(exitname)
Specifies the 1 to 8 character name of the MMS pre-processing or post-processing installation exit.

Table 30. Language Codes

Code	Language Name	Country or Region
CHS	Simplified Chinese	China (PRC)
CHT	Traditional Chinese	Taiwan
DAN	Danish	Denmark
DEU	German	Germany
DES	Swiss German	Switzerland
ELL	Greek	Greece
ENG	UK English	United Kingdom
ENP	US English (upper case)	United States
ENU	US English (mixed case)	United States
ESP	Spanish	Spain
FIN	Finnish	Finland
FRA	French	France
FRB	Belgian French	Belgium
FRC	Canadian French	Canada
FRS	Swiss French	Switzerland

Table 30. Language Codes (continued)

Code	Language Name	Country or Region
ISL	Icelandic	Iceland
ITA	Italian	Italy
ITS	Swiss Italian	Switzerland
JPN	Japanese	Japan
KOR	Korean	Korea
NLD	Dutch	Netherlands
NLB	Belgian Dutch	Belgium
NOR	Norwegian	Norway
PTG	Portuguese	Portugal
PTB	Brazil Portuguese	Brazil
RMS	Rhaeto-Romanic	Switzerland
RUS	Russian	Russia
SVE	Swedish	Sweden
THA	Thai	Thailand
TRK	Turkish	Turkey

Chapter 71. MPFLSTxx (message processing facility list)

MPFLSTxx contains information that the message processing facility (MPF) uses to control:

- Message presentation

  On certain devices, messages can appear with highlighting, in color, or with added intensity.
- Message management

  Message management refers to message suppression, message retention, and message processing.
  - Message suppression

    A suppressed message does not appear at a console but is written to the hard-copy log.
  - Message retention

    The action message retention facility (AMRF) keeps action messages in a buffer area, allowing the operator to request that any action message not acted on be recalled to the screen. In MPFLSTxx, you can identify certain action messages that AMRF is not to retain.
  - Message processing

    Use MPFLSTxx to identify messages to be processed either using a message automation subsystem (for example, NetView® or using installation-written exits.
- Command Processing

  You can specify installation exits that will get control each time a command is issued.

This description of the MPFLSTxx parmlib member has the following order:

- General information including syntax rules and selecting MPFLSTxx members.
- Controlling message presentation using MPFLSTxx, in “Controlling message presentation through MPFLSTxx” on page 650.
- Controlling message management using MPFLSTxx, in “Controlling message management” on page 653.
- Controlling command processing using MPFLSTxx, in “Controlling command processing using MPFLSTxx” on page 665.

Parameter in IEASYSxx

None.

Syntax rules for MPFLSTxx

These rules apply to the creation of MPFLSTxx:

- Each record is 80 characters long. The system ignores columns 72 through 80 and leading blanks.
- Each message processing record can contain only one message identifier.
- The message processing record for msgid, .DEFAULT, or MPFHCF cannot have embedded blanks.

- The message processing record for .MSGCOLR and .MSGIDS must have a single blank between the statement and the operand. Embedded blanks are not allowed anyplace else on the message processing record.
- Begin comments with /* in any column. It is best to end the comment with a */. If, however, you do not include the end delimiter, the system recognizes the next statement.

Selecting MPFLSTxx members

You can select a particular MPFLSTxx parmlib member, or a concatenation of MPFLSTxx members (up to 39), in the following ways:

- Specifying the INIT statement with the MPF keyword in the CONSOLxx parmlib member. CONSOLxx is used at system initialization.
- Including the SET MPF command in the COMMNDxx parmlib used at system initialization.
- Issuing the SET MPF command after system initialization.

Two alphanumeric characters are appended to MPFLST to form the name of the MPFLSTxx parmlib member. Multiple members can be specified.

Note: Specify one or more MPFLSTxx parmlib members in **either** the CONSOLxx member **or** in the COMMNDxx member. Do not maintain this selection in both members. After initialization, you can issue the SET MPF command to change one or more MPFLSTxx members; however, this change is temporary. At the next IPL, the system uses the MPFLSTxx members specified in the CONSOLxx or COMMNDxx member. See the information about the CONSOLxx and COMMNDxx parmlib members.

If you issue the SET MPF=NO command, the system uses the IBM-supplied defaults for message presentation and for message management. For descriptions of the SET command, see *z/OS MVS System Commands*.

IBM-supplied MPFLSTxx member

None.

Controlling message presentation through MPFLSTxx

Message presentation refers to color, highlighting, and intensity attributes that the system uses when displaying messages on different areas of an operator console. You can specify these attributes on the .MSGCOLR statement in the MPFLSTxx member; however, the console must support the attributes you specify. On the .MSGCOLR statement, you can indicate the display attributes that the system is to use in one of the following ways:

- IBM-supplied defaults (DEFAULT).
- Attributes specified in the previously used MPFLSTxx member or concatenation of MPFLSTxx members (NOCHANGE).
- Color (*c*), highlighting (*h*), and intensity (*i*) attributes specified within MPFLSTxx for a message area (*msgarea*).

  For more information about these definitions, see Table 31 on page 652.

Syntax for controlling message presentation

The syntax for the **.MSGCOLR** statement is:

```
        {DEFAULT                                }[/*comments*/]
.MSGCOLR {NOCHANGE                              }
        {msgarea(c,h[,i])[,msgarea(c,h[,i])]...}
```

IBM-supplied defaults for .MSGCOLR

If you do not specify a .MSGCOLR statement, the system uses IBM-supplied defaults. See the following description on the .MSGCOLR statement for these defaults.

If you specify a .MSGCOLR statement with no operands, the system defaults to NOCHANGE. NOCHANGE indicates that the color, highlighting, and intensity attributes are to be the same as those specified in the previously used MPFLSTxx member (or concatenation of MPFLSTxx members).

Displaying the message presentation attributes for the current MPFLSTxx

You can use the DISPLAY MPF command to list the current color intensity and highlighting specifications for a specific console or for all consoles. For a description of the DISPLAY MPF command, see *z/OS MVS System Commands*.

MPFLSTxx parameters for controlling message presentation

.MSGCOLR

.MSGCOLR indicates the beginning of a statement that defines the display attributes for messages.

DEFAULT

DEFAULT causes the system to use the IBM-supplied color, highlighting, and intensity defaults. See the following description on *msgarea*. (Table 32 on page 652 lists the defaults.) If you need to specify a .MSGCOLR DEFAULT statement, specify only one.

If the first .MSGCOLR statement in the MPFLSTxx concatenation specifies DEFAULT, the system ignores any later .MSGCOLR statements, issues an error message, and continues processing any remaining statements.

NOCHANGE

NOCHANGE indicates that the color, highlighting, and intensity attributes are to be the same as those specified in the previously used concatenation of MPFLSTxx members. If you need to specify a .MSGCOLR NOCHANGE statement, specify only one. If the MPFLSTxx concatenation does not contain a .MSGCOLR statement, the system defaults to NOCHANGE.

If the first .MSGCOLR statement specifies NOCHANGE, the system ignores any later .MSGCOLR statements, issues an error message, and continues processing any remaining statements. The system uses the color, highlighting, and intensity attributes specified in the previously used concatenation of MPFLSTxx members.

msgarea(c,h[,i])

msgarea specifies the message area (message type or field) and contains the accepted values and the defaults for *msgarea*. This operand allows you to

specify the color (*c*), highlighting (*h*), and intensity (*i*) attributes for a message area. Table 31 summarizes the supported attributes.

Table 31. Attributes to specify for a message area

Color attributes (*c*)	Highlighting attributes (*h*)	Intensity attributes (*i*)
• R-Red • W-White • G-Green • B-Blue • P-Pink • Y-Yellow • T-Turquoise	• N-Normal (colored characters on a black background) • B-Blinking • R-Reverse video (black characters on a colored background) • U-Underscored characters	• N-Normal intensity • H-High intensity

If you specify a *msgarea*, you must specify the color and highlighting attributes. If you do not, or if the system detects an error in any specified value, it issues an error message, stops checking the statement in error, and continues processing any later statements. The intensity attribute, however, is optional. If you do not specify the intensity attribute, the system uses the IBM-supplied defaults.

If you specify multiple .MSGCOLR statements for the same *msgarea*, the system uses the last valid statements.

These attributes can be overridden by the extended message highlighting options in IEAVMXIT or the MPF exit routines. For more information about coding these exit routines, see *z/OS MVS Installation Exits*.

Table 32. Values for msgarea

msgarea Value	Meaning and Use	Defaults (c,h,i)
AUTOR	Specifies attributes for write-to-operator with reply (WTOR) messages that are being monitored by auto-reply processing.	(W,N,H)
ENTRYARA	Specifies attributes for the entry area.	(G,N,N)
EVETACTN	Specifies attributes for eventual action messages (issued with a descriptor code of 3).	(G,N,N)
GENMSG	Specifies attributes for general system messages (issued with a descriptor code other than 1, 2, 3 or 11).	(G,N,N)
IMEDACTN	Specifies attributes for immediate action messages (issued with a descriptor code of 2 or WTOR messages). **Note:** For a device with limited image capability (such as the 3290), the reverse video highlighting attribute applies only to the action characters (* or @) that precede the message identifier.	(W,N,H)
INSTRERR	Specifies attributes for error messages that appear in the instruction line.	(W,N,H)
OOLCNTL	Specifies attributes for the control line text of an out-of-line status display, not including the selector pen detectable fields on the control line.	(T,N,N)
OOLDATA	Specifies attributes for the data lines of an out-of-line status display.	(G,N,N)
OOLLABEL	Specifies attributes for the label lines of an out-of-line status display.	(T,N,N)
PPMSG	Specifies attributes for non-action messages issued by a problem program (issued with a descriptor code other than 1, 2, 3, or 11).	(G,N,N)

Table 32. Values for msgarea (continued)

msgarea Value	Meaning and Use	Defaults (c,h,i)
SELPEN	Specifies attributes for fields that are selector pen detectable, such as: • in the control line of an out-of-line display - – F – E • in the control line of an in-line display - – C	(B,N,N)
URGATTN	Specifies attributes for urgent attention messages (issued with a descriptor code of 1 for system failure or 11 for critical eventual action). **Note:** For a device with limited image capability (such as the 3290), the reverse video highlighting attribute applies only to the action characters (* or @) that precedes the message identifier.	(R,N,H)
WARNLGEN	Specifies attributes for the left half of the warning line for general messages such as: • IEE163I MODE=R • IEE163I MODE=RD • IEE161I WARNING-CON=N, DEL=Y	(B,N,N)
WARNRGEN	Specifies attributes for the right half of the warning line for general messages such as: • IEE160I UNVIEWABLE MESSAGE	(B,N,N)
WARNRURG	Specifies attributes for the right half of the warning line for urgent attention messages such as: • IEE159E MESSAGE WAITING	(R,B,H)
Note: If you specify EVETACTN, GENMSG, OOLDATA, OOLLABEL, or PPMSG for a device with limited image capability (such as the 3290), the system ignores the reverse video highlighting attribute.		

Controlling message management

MPFLSTxx allows you to specify how you want to process WTO/WTOR messages. **Message management** refers to message suppression, message retention, message automation eligibility, and the use of installation-supplied WTO/WTOR exit routines.

Message suppression means that the message is logged in the hard-copy log, but it does not appear at an MVS operator's console.

Message retention means that action messages are saved by the action message retention facility (AMRF) on the action message retention queue. Retention allows the operator to view the message later. If you choose not to retain a message, the system will not add it to the action message retention queue.

Message automation eligibility means that you are using an automation subsystem to process particular WTO/WTOR messages in a pre-determined way. The automation subsystem (such as NetView) allows the installation to program the processing for a particular message. For example, your installation may be using an automation subsystem to:

- Modify the text of a message.
- Re-route a message to a different operator's console or to a different system for processing.
- Record some information contained in the message text for future use.

- Respond to the message.

An automation subsystem can look at the message text and the message attributes and perform the programmed action. You can use MPFLSTxx to identify whether you want a message to be eligible for automation processing. The automation subsystem must then select and process the messages. You can also use MPFLSTxx to indicate information to pass to an automation subsystem. Messages that are eligible for automation processing can also be suppressed or retained. Message suppression and retention are functions separate from automation. Beginning with NetView Release 2, the NetView subsystem recognizes the message automation eligibility specifications in MPFLSTxx. For more details, see *Tivoli® NetView for z/OS Automation Guide* and *Tivoli NetView for z/OS Administration Reference.*

Your installation might have installation-written exit routines to handle WTO/WTOR messages. Through MPFLSTxx, you can specify which messages are to go to which exit routine. The exit routine can examine the message text and the message attributes and decide whether to suppress, retain, or take other actions on the message. The exit routine can also examine and modify the token.

Specifying message management

For the messages that you specify in MPFLSTxx, you can indicate that the system is to:

- Suppress specific messages or all messages that begin with a particular prefix. SUP(YES/NO) provides this option.
- Retain action messages (descriptor codes 1, 2, 3, 11) using the action message retention facility (AMRF). The RETAIN parameter allows you to selectively retain certain types of action messages (immediate, eventual, or critical eventual).
- Allow an automation subsystem, such as NetView, to respond to the WTO/WTOR message(s) that you specify. AUTO(YES/NO/token) allows you to identify whether a message is eligible for automation processing and to pass information (a token) to the automation subsystem.
- Pass control to an installation-supplied WTO/WTOR exit routine to process the message(s). The USEREXIT option allows you to identify the exit routine.
- Assign installation-supplied defaults for specific messages that are identified in the MPFLSTxx member. The .DEFAULT statement allows you to change the system-assigned message processing defaults for any message you specify.
- Assign installation-supplied defaults for messages that are not identified in the MPFLSTxx member. The .NO_ENTRY statement allows you to indicate how the system is to handle messages that are not specified in the MPFLSTxx member.
- Indicate or flag the beginning of particular messages that are written to the JES3 hard-copy log. The MPFHCF statement allows you to specify the character you want to use as an indicator.

Syntax for controlling message management

To control message management, MPFLSTxx recognizes one parameter, **msgid**, and four statements: (**.DEFAULT**, **.NO_ENTRY**, **MPFHCF**, and **.MSGIDS**)

- **msgid** allows you to specify a particular message id or prefix. This specification is also called the message processing record.
- **.DEFAULT** allows you to specify the defaults for groups of messages listed in the MPFLSTxx parmlib member.

- **.NO_ENTRY** allows you to specify the default processing you want for messages that are **NOT** identified in any of the active MPFLSTxx parmlib members. (Note that you must code the underscore (_) in the statement.)
- **MPFHCF** allows you to indicate or flag messages that JES3 writes to the hard-copy log. Note that this statement does not begin with a period (.).
- **.MSGIDS NOCHANGE** allows you to specify that the messages identifiers and the message processing are to be the same as those specified in the previously used concatenation of MPFLSTxx members.

The syntax of the **message processing record (msgid)** is:

```
msgid     [,AUTO  [(YES)  ]  ,RETAIN  [(YES)   ]  ,SUP [(YES)]]
          [       [(NO)   ]           [(NO)    ]       [(ALL)]]
          [       [(TOKEN)]           [(I,E,CE)]       [(NO) ]]
          [                                              ]
          [[,USEREXIT(exitname)][/*comments*/]           ]
```

The syntax of the **.DEFAULT** statement is:

```
.DEFAULT  [,AUTO  [(YES)  ]  ,RETAIN  [(YES)   ]  ,SUP [(YES)]]
          [       [(NO)   ]           [(NO)    ]       [(ALL)]]
          [       [(TOKEN)]           [(I,E,CE)]       [(NO) ]]
          [                                              ]
          [[,USEREXIT(exitname)][/*comments*/]           ]
```

The syntax of the **MPFHCF** statement is:

```
MPFHCF=[x/&]
```

The syntax for the **.MSGIDS** statement is:

```
.MSGIDS NOCHANGE
```

The syntax of the **.NO_ENTRY** statement is:

```
.NO_ENTRY [,AUTO  [(YES)  ] ,RETAIN [(YES)   ] ,SUP[(YES)]]
          [       [(NO)   ]         [(NO)    ]     [(ALL)]]
          [       [(TOKEN)]         [(I,E,CE)]     [(NO) ]]
          [                                          ]
          [[/*comments*/]                            ]
```

For descriptions of these statements and parameters, see “Statements/parameters for MPFLSTxx” on page 658.

IBM-supplied defaults for message management

- If you have *msgid* (message processing) statements in the MPFLSTxx member, the defaults vary and are described in the syntax and parameter descriptions for the *msgid*, .DEFAULT, and .NO_ENTRY statements.
- If you do not specify an MPFLSTxx member, the defaults are AUTO(YES), RETAIN(YES), and SUP(NO). The system considers all messages eligible for automation processing, it retains action messages, and displays all messages.

Listing the message processing attributes for the current MPFLSTxx

You can use the DISPLAY MPF command to list the results of the current MPFLSTxx member. The DISPLAY MPF command displays:

- The messages being suppressed by MPF.
- The action messages that are **not** being retained by the action message retention facility.
- The installation exits that receive control for selected messages.
- The status of the general-use WTO installation exit, IEAVMXIT.

For descriptions of the DISPLAY command, see *z/OS MVS System Commands*.

Note: The DISPLAY MPF command will not display if a message is eligible for automation processing. The hard-copy log, however, will contain the automation tracking indicator.

To diagnose MPF processing, you can also use the hard-copy log to determine the message processing options used for a message.

Using other methods to suppress messages

WTO/WTOR messages can be suppressed through an MPFLSTxx member and by other methods. When an installation uses more than one method, message suppression is performed in the following order:

1. The message processing facility (MPF) itself suppresses messages.
2. An installation-supplied WTO/WTOR installation exit (either IEAVMXIT or an exit routine you name on the USEREXIT parameter in MPFLSTxx) can suppress messages. (Note that the exit routine can also override MPF suppression. That is, the exit can prevent MPF from suppressing a message.)
3. An active subsystem (such as JES2, JES3, or NetView) can suppress messages.
4. CONTROL V command can suppress messages by specifying only the message levels that are to be displayed at a console. The LEVEL keyword on the CONSOLE statement in the CONSOLxx member of SYS1.PARMLIB controls the message levels for the console.

Controlling foreign message management

A foreign message is one that is received from another system. MPFLSTxx allows you to specify whether you want subsystems on your system to receive these messages. Many subsystems do not need to see foreign messages. You can reduce the amount of CPU utilized in the CONSOLE address space for foreign message processing by limiting the number of subsystems that receive foreign messages.

Syntax for controlling message management

To control the management of foreign messages, MPFLSTxx recognizes one statement type **.FORNSSI**. This statement type allows you to specify whether foreign messages (both WTOs and DOMs) are to be presented to any or all subsystem interface (SSI) exits listening to the WTO and DOM broadcast function codes. The **.FORNSSI** statement has the following syntax.

```
.FORNSSI   *ALL
                *NONE
                NOCHANGE
                (list of subsystems)
```

Syntax rules for the .FORNSSI statement

These rules apply specifically to the .FORNSSI statement in addition to the general rules listed in "General syntax rules for the creation of members" on page 9.

- Only one .FORNSSI statement is allowed. The system rejects any additional .FORNSSI statement and issues message IEF760I.
- A **.FORNSSI** statement without any operand is treated as a syntax error and the system issues message IEF760I.
- Subsystem names must be separated by a comma and must be enclosed in parentheses, even if the list is composed of only one subsystem name.
- The **.FORNSSI** statement cannot span to the next record.

IBM-supplied defaults for foreign message management

If no MPFLSTxx member has ever been set or SET MPF=NO is issued, the system defaults to the *ALL state.

Displaying the foreign message processing attributes for the current MPFLSTxx

You can use the DISPLAY MPF command to list the current installation options for handling foreign messages. For a description of the DISPLAY MPF command, see z/OS MVS System Commands.

MPFLSTxx parameters for managing foreign messages

.FORNSSI

.FORNSSI indicates the beginning of a statement that defines the attributes for handling foreign messages.

***ALL**

Indicates that all subsystems are to receive foreign messages and DOMs.

***NONE**

Indicates that no subsystems are to receive foreign messages and DOMs.

NOCHANGE

Indicates that the system are to retain the previous .FORNSSI statement.

(list of subsystems)

Indicates one or more subsystems that are to receive foreign messages and DOMs. Subsystems not in this list do not receive foreign messages and DOMs.

Controlling verbose message production

A *verbose message* is a multi-line message that contains additional lines of explanation. MPFLSTxx allows you to specify whether you want components of your system that support verbose messages to produce verbose messages or not. If you request that verbose messages be produced, they will be included in their

entirety in the JOBLOG but the extra lines of explanation will not be included in the SYSLOG or OPERLOG and will not be queued to any consoles.

Syntax for controlling the production of verbose messages

To control the production of verbose messages, MPFLSTxx recognizes the statement type **.MSGOPTION**, which allows you to specify whether verbose messages are to be produced by the components of your system. The syntax of the **.MSGOPTION** statement is as follows.

```
.MSGOPTION VERBOSE[(Y)]
                  [(N)]
```

Syntax rules for the .MSGOPTION statement

These rules apply specifically to the .MSGOPTION statement in addition to the general rules listed in "General syntax rules for the creation of members" on page 9.

- Only one .MSGOPTION statement is allowed. The system rejects any additional .MSGOPTION statement and issues message IEF760I.
- A .MSGOPTION statement without any operand is treated as a syntax error and the system issues message IEF760I.

IBM-supplied default for verbose message production

If no MPFLSTxx member has ever been set or SET MPF=NO is issued, the system defaults to the VERBOSE(N) state.

Displaying the verbose message production setting for the current MPFLSTxx

You can use the DISPLAY MPF command to display the current verbose message production setting. For a description of the DISPLAY MPF command, see *z/OS MVS System Commands*.

MPFLSTxx parameters for managing verbose message production

.MSGOPTION
Indicates the beginning of a statement that defines message processing options.

VERBOSE(Y)
Indicates that verbose messages are to be produced.

VERBOSE(N)
Indicates that verbose messages are not to be produced.

Statements/parameters for MPFLSTxx

msgid
Identifies a message or group of messages to be processed. The *msgid* consists of:

- A complete message identifier of one to ten characters. This is known as a **specific** message identifier.
- A message prefix, which is a portion of the message identifier, followed by an asterisk(*). This is known as a **generic** message identifier.

- A single quotation mark (') is **not** a valid character in a message identifier.

A message identifier begins with the first non-blank character of the message text and continues until the next blank.

Note: The system might insert a character, such as a + or *, preceding the message identifier. This character is **not** a part of the message identifier. Do not add it to the *msgid* specification.

You can specify only one msgid per record. If a msgid is repeated within an MPFLSTxx member, the system uses the options specified on the first record and ignores the duplicates.

If you want MPF to process a specific message, you must specify the complete message id, for example IEF124I.

If you want MPF to process all messages that begin with a specific prefix, you can specify the prefix and an asterisk, for example, IEF24*. Use the message prefix with an asterisk carefully. Too wide a suppression, such as IEF*, could suppress many messages that you need for effective system operation.

With one exception (see note below), specific message definitions take precedence over generic message definitions. For example, if you specify both a specific entry for IEF638I, and a generic entry that includes IEF638I (such as IEF63*), the system uses the specific entry to process IEF638I.

In situations where a message is included by more than one generic message definition, such as IEF6* and IEF63*, the **most specific** definition takes precedence. In this example, the system uses the entry for IEF63* for message IEF638I, if no specific entry exists.

Note: A *msgid* that consists of only a single asterisk ('*') does not mean "all messages," but rather messages that have a single asterisk as their message identifier.

,AUTO(YES/NO/token)

Specifies whether the message (*msgid*) is eligible for processing by an automation subsystem, such as NetView. AUTO(YES) makes the message eligible for automation processing. AUTO(NO), which is the system default, makes the message ineligible for automation processing.

If you specify *msgid* and do not specify the AUTO option, the system defaults to NO. If, however, you do not specify any message processing statements for a message, the system considers the message eligible for automation processing AUTO(YES). To change the system default, use the .DEFAULT or the .NO_ENTRY statement.

The *token* value is available for MPF installation exit processing and for processing by an automation subsystem. Specifying a token indicates that the message is eligible for processing by the automation subsystem. The token value must be 1 to 8 alphanumeric characters. You may not use a left parenthesis "(" as part of the token value. Imbedded blanks are allowed in the token value. For example, if you code an 'N,O, and a blank', AUTO(NO), the system takes the "NO" as a token value.

Note: A message might loop repetitively to an extended MCS console if all of the following are true:

- AUTO(YES) is specified, and
- The RACF OPERPARM segment specifies AUTO YES for the extended MCS console, and

- The console profile, as defined through TSO/E CONSPROF, specifies YES for the UNSOLDISP parameter or for the SOLDISP parameter or for both.

,RETAIN(YES/I,E,CE/NO)

If the message identified by the *msgid* is an action message (immediate, eventual, or critical eventual), RETAIN specifies whether the message is to be retained by the action message retention facility (AMRF). AMRF retains action messages only (not WTORs).

YES all action messages are to be retained; this is the default

I immediate action messages are to be retained

E eventual action messages are to be retained

CE critical eventual action messages are to be retained

NO no action messages are to be retained

You can specify any combination of *I*, *E*, or *CE*.

The system default is RETAIN(YES). You can use the .DEFAULT statement followed by a msgids to change the system default for the list of messages.

To view a retained message, use the DISPLAY R command.

,SUP(YES/ALL/NO)

Specifies whether MPF is to suppress the message identified by *msgid*.

SUP(YES) is the default, and indicates that the system is to suppress messages identified by *msgid*, with the following exceptions:

- Command responses where MCSFLAG=RESP was specified on the WTO or WTOR macro.
- Command responses with descriptor code 5 (immediate command response).

  Note: This exception does not apply to command responses with descriptor code 5 that are generated in response to the MONITOR command. SUP(YES) suppresses such responses.

SUP(ALL) specifies that the system is to suppress all messages identified by *msgid*, without exception.

SUP(NO) specifies that the system should not suppress messages identified by *msgid*.

If you specify a *msgid* without SUP, the system uses the default, SUP(YES), and does not display the message. If, however, you do not specify any message processing statements for a message, the system displays the message. To change the system defaults, use the .DEFAULT or the .NO_ENTRY statement.

,USEREXIT(*exitname*)

Specifies the name of an installation-supplied WTO/WTOR installation exit routine that is to get control each time the system issues the message(s). This routine can process the message(s); it can suppress, retain, or respond to a message. It can make the message eligible for automation processing, and take other actions on the message. The exit should be link-edited into an APF-authorized library that is part of the LNKLST concatenation.

The *exitname* can be from one to eight alphanumeric (A-Z, 0-9) and national characters (&, *, $). The first character must be alphabetic or

numeric. If you do not specify an exitname, the system uses IEAVMXIT, if it exists. To change the system default, use the .DEFAULT statement.

For more information about the WTO/WTOR installation exits, see Chapter 23, "CONSOLxx (console configuration definition)," on page 231 and *z/OS MVS Installation Exits*.

.DEFAULT

.DEFAULT allows you to specify the defaults that you want for the message processing records (msgids) that follow .DEFAULT. The options you specify on the .DEFAULT statement override the system defaults for messages that you list. On the .DEFAULT statement, you can specify that the message is eligible for automation processing, retention, and/or suppression, and the installation exit that is to process the message. Through AUTO(token), you can also specify information to be passed to the automation subsystem. .DEFAULT with no options, results in the message being ineligible for automation processing, and indicates that the system is to suppress and retain any listed action messages. IEAVMXIT, if it exists, receives that message.

You can use the .DEFAULT statement multiple times within an MPFLSTxx member. Each group of messages following a .DEFAULT statement should have common option values. This allows you to control attributes assigned by default to each message id without having to change every message processing record.

On a particular message record (msgid statement) that follows a .DEFAULT statement, you can specify specific operand values that override, for that message, the .DEFAULT values.

If there are multiple occurrences of a message id listed under one .DEFAULT statement, the system uses the options for the first occurrence and ignores the others.

If an MPFLSTxx member contains multiple .DEFAULT statements, the system uses the values on the .DEFAULT statement that precedes the first message record (msgid statement) in that group.

,AUTO(YES/NO/token)

Specifies whether a message or a list of messages following the .DEFAULT statement is eligible for processing by an automation subsystem, such as NetView. AUTO(YES) makes the message(s) eligible for automation processing. AUTO(NO) (the system default for the .DEFAULT statement) makes the subsequent message(s) ineligible for automation processing.

The *token* value is available for MPF installation exit processing and for processing by an automation subsystem. Specifying a token indicates that the message is eligible for processing by the automation subsystem. The token value must be 1 to 8 alphanumeric characters. You may not use a left parenthesis "(" as part of the token value. Imbedded blanks are allowed in the token value. For example, if you code an 'N,O, and a blank', AUTO(NO), the system uses the "NO" as a token value.

Note:

1. If you specify a .DEFAULT and omit the AUTO option, the system will not consider the message(s) eligible for automation processing. If, however, you do not specify any message processing statements for a message, the system considers the message eligible for automation processing.

2. A message might loop repetitively to an extended MCS console if all of the following are true:
   - AUTO(YES) is specified, and
   - The RACF OPERPARM segment specifies AUTO YES for the extended MCS console, and
   - The console profile, as defined through TSO/E CONSPROF, specifies YES for the UNSOLDISP parameter or for the SOLDISP parameter or for both.

,RETAIN(YES/I,E,CE/NO)

RETAIN specifies which subsequent action messages are to be retained by the action message retention facility (AMRF).

YES all action messages are to be retained; this is the default

I immediate action messages are to be retained

E eventual action messages are to be retained

CE critical eventual action messages are to be retained

NO no action messages are to be retained

You can specify any combination of I, E, or CE. To view a retained message, use the DISPLAY R command.

,SUP(YES/ALL/NO)

Specifies whether MPF is to suppress subsequent messages.

SUP(YES) is the default, and indicates that the system is to suppress subsequent messages, with the following exceptions:

- Command responses where MCSFLAG=RESP was specified on the WTO or WTOR macro.
- Command responses with descriptor code 5 (immediate command response).

  Note: This exception does not apply to command responses with descriptor code 5 that are generated in response to the MONITOR command. SUP(YES) suppresses such responses.

SUP(ALL) specifies that the system is to suppress all subsequent messages, without exception.

SUP(NO) specifies that the system should not suppress subsequent messages.

If you specify a .DEFAULT statement without SUP, the system uses the default, SUP(YES), and suppresses subsequent messages.

If you specify a .DEFAULT statement with one of the SUP options, and then specify SUP on a subsequent message record, the system uses the SUP value specified on that particular message record.

If you do not specify any message processing statements for a message, the system displays the message, regardless of any SUP value that might have been specified on a preceding .DEFAULT statement.

,USEREXIT(*exitname*)

Specifies the name of an installation-supplied WTO/WTOR installation exit routine that is to get control each time the system issues one of the

v indicated messages. This routine then processes the message(s). The exit
v should be link-edited into an APF-authorized library that is part of the
v LNKLST concatenation.

The *exitname* can be from one to eight alphanumeric (A-Z, 0-9) or national characters (@, $, *). The first character must be alphabetic or numeric. If you do not specify an *exitname* the system defaults to IEAVMXIT, if it exists.

For more information about the WTO/WTOR installation exits, see the CONSOLxx member of SYS1.PARMLIB in this book, and *z/OS MVS Installation Exits*.

MPFHCF=[x/&]

If JES3 writes the messages to the hard-copy log, this statement specifies the indicator (*x*) used to identify suppressed messages in the hard-copy log. (*JES2 ignores this statement.*) The indicator can be any character, including a blank. If *x* specifies an indicator that is not a printable character, the indicator is translated to a blank. When MPFHCF is not specified, the default character is an ampersand (&).

Note: This statement does NOT begin with a period (.).

.MSGIDS

NOCHANGE

.MSGIDS NOCHANGE specifies that the message identifiers are to be the same as those specified in the previously used concatenation of MPFLSTxx members (the one the system was previously using). Use this statement with the .MSGCOLR statement.

The .MSGIDS NOCHANGE statement should not be specified in an MPFLSTxx member or a concatenation of MPFLSTxx members that includes message identifiers. If syntactically correct message identifiers (msgid statements) precede the .MSGIDS NOCHANGE statement, the system ignores the statement, issues an IEF760I error message, and continues processing the remaining statements. If a valid .MSGIDS NOCHANGE statement precedes message identifiers, the system ignores the message identifiers, issues an IEF760I error message, and continues processing the remaining statements.

If multiple .MSGIDS NOCHANGE statements occur in one MPFLSTxx member, or a concatenation of MPFLSTxx members, the system processes each statement as though it were the first statement.

.NO_ENTRY

.NO_ENTRY specifies the message processing options for all messages that are NOT specified in MPFLSTxx members. The options you specify on the .NO_ENTRY statement override the system defaults for messages that are not specified in this member. On the .NO_ENTRY statement, you can specify whether all messages not specified in the MPFLSTxx member are to be considered eligible for automation processing, retained, and suppressed.

.NO_ENTRY is a very powerful statement and should be used with care.

Note: .NO_ENTRY requires the underscore (_) in the syntax. You might not, however, be able to print the underscore character on your printer.

If you specify a .NO_ENTRY statement with no options, the system considers the messages that are not specified in the MPFLSTxx member to be eligible for automation processing, it retains action and WTOR messages, and it displays

all messages (AUTO(YES) RETAIN(YES) SUP(NO)). These options are the same defaults that the system uses when no message processing record is specified in an MPFLSTxx member for a particular message.

.NO_ENTRY checks the syntax and then ignores the USEREXIT(exitname) statement. IEAVMXIT receives control, if it exists.

Specify only one .NO_ENTRY statement in an MPFLSTxx member. If there is more than one occurrence of a .NO_ENTRY statement in an MPFLSTxx member, the system checks the syntax of the duplicate and uses the options on the first .NO_ENTRY statement. If there is more than one occurrence of a .NO_ENTRY statement in a concatenation of MPFLSTxx members, the system uses the options on the first valid .NO_ENTRY statement in the concatenation.

,AUTO(YES/NO/token)

For messages that are not identified in MPFLSTxx, AUTO indicates whether the message is eligible for processing by an automation subsystem, such as NetView. AUTO(YES), the system default, makes the messages eligible for automation processing. AUTO(NO) makes the messages ineligible for automation processing. (Specifying AUTO without either YES or NO results in a syntax error.)

The *token* value is available for MPF installation exit processing and for processing by an automation subsystem. Specifying a token indicates that the message is eligible for processing by the automation subsystem. The token value must be 1 to 8 alphanumeric characters. You may not use a left parenthesis "(" as part of the token value. Imbedded blanks are allowed in the token value. For example, if you code an 'N,O, and a blank', AUTO(NO), the system uses the "NO" as a token value.

Note:

1. IEAVMXIT, if it exists, gets control for all messages whose message ids are not specified in the MPFLSTxx member.
2. A message might loop repetitively to an extended MCS console if all of the following are true:
   - AUTO(YES) is specified, and
   - The RACF OPERPARM segment specifies AUTO YES for the extended MCS console, and
   - The console profile, as defined through TSO/E CONSPROF, specifies YES for the UNSOLDISP parameter or for the SOLDISP parameter or for both.

,RETAIN(YES/I,E,CE/NO)

For all action messages (immediate, eventual, or critical eventual) not identified in MPFLSTxx, RETAIN specifies which messages are to be retained by the action message retention facility (AMRF).

- YES indicates that all action messages are to be retained. The default is RETAIN(YES).
- I indicates that immediate action messages are to be retained.
- E indicates that eventual action messages are to be retained.
- CE indicates that critical eventual action messages are to be retained.
- NO indicates that no action messages are to be retained.

You can specify any combination of I, E, or CE. To view a retained message, use the DISPLAY R command.

,SUP(YES/ALL/NO)
Specifies whether MPF is to suppress messages that are not specified in MPFLSTxx members.

SUP(YES) indicates that the system is to suppress messages, with the following exceptions:

- Command responses where MCSFLAG=RESP was specified on the WTO or WTOR macro.
- Command responses with descriptor code 5 (immediate command response).

  Note: This exception does not apply to command responses with descriptor code 5 that are generated in response to the MONITOR command. SUP(YES) suppresses such responses.

SUP(ALL) specifies that the system is to suppress all messages, without exception. SUP(NO) specifies that the system should not suppress messages.

: SUP(ALL) causes the system to suppress **all** messages that are not identified in the active MPFLSTxx member. This setting might result in most or all messages being suppressed from the MCS console. Suppression of all messages is useful in certain situations, such as a remote system, but it can be detrimental if an operator is expecting these messages.

Controlling command processing using MPFLSTxx

Besides message management, MPFLSTxx allows you to specify up to six names of command installation exits that let you modify commands to be processed. You can take the following steps in the command installation exits:

- Change the text of commands.
- In a sysplex, change the destination of commands by routing them to a different system for execution.
- Modify a console's authority to use a particular command. That is, you can use the exit to:
  - Authorize the command from a console that normally would not have the authority to issue the command.
  - Reject the command from a console that normally would not have the authority to issue the command.
- Execute commands.
- Suppress commands.

The command exits receive control every time a command is issued. For more information about the command installation exit, see *z/OS MVS Installation Exits*.

MPFLSTxx parameters for controlling command processing

.CMD
.CMD allows you to specify the installation exit(s) that modifies a command issued in a system or sysplex. These installation exits allow you to modify how commands are issued, the command text, or the MCS command authority of the console that is to issue the command.

You can have only one .CMD statement in an MPFLSTxx member. If more than one .CMD statement appears in an MPFLSTxx member (or a concatenation of MPFLSTxx members), the system uses the first occurrence of the .CMD statement.

For more information about the command installation exit, see *z/OS MVS Installation Exits*.

USEREXIT(*exitnam1***[,***exitnam2***],...)**
USEREXIT specifies the names of one or more command installation exits. Each name can be from one to eight alphanumeric characters. The command installation exit specified by *exitname* receives control when an MVS command is issued. Each installation exit is invoked in the order specified on this statement. You can code up to six unique 8-character command installation exit names on a statement, separating the names with one or more commas or blanks.

NOCHANGE
NOCHANGE specifies that the command installation exits are to be the same as those specified in the previously used concatenation of MPFLSTxx members. NOCHANGE is the default.

The system processes either the .CMD NOCHANGE statement or the installation command exit names (.CMD USEREXIT) depending on which is coded first in MPFLSTxx. The system ignores the second one.

If comments are used on the .CMD statement, at least one blank must separate them from the rest of the data on the statement.

Deactivating a command exit

There are times when you might want to deactivate a command exit routine, perhaps because its function is not required at particular times or because you want to modify the routine. You can deactivate a command exit routine in either of two ways:

- Specify, on the .CMD statement of the appropriate MPFLSTxx member, the name of a command exit that does not exist in SYS1.LINKLIB, such as 'USEREXIT(NONE)'. Enter the SET MPF=xx command to refresh the MPFLSTxx member.

  This action effectively deactivates any command exits that were enabled during the prior MPFLSTxx activation. The system issues an informational message that can be ignored in this case.
- Enter the SET MPF=NO command to disable all active MPFLSTxx members. Remove the exit name from the .CMD statement of the appropriate MPFLSTxx member and enter the SET MPF=xx command to resume MPF processing.

  : Entering the SET MPF=NO command causes all installation-specified MPF options to be deactivated. IBM-supplied defaults are used until the installation re-activates its MPFLSTxx members.

Approaches to message suppression using MPFLSTxx

You can use MPFLSTxx for message suppression conservatively or aggressively. In a *conservative* environment, you would suppress messages by identifying the complete ids. You would suppress, however, only messages regarded from an operator's perspective as useless.

In an *aggressive* environment, you would suppress messages through the conservative approach and you would possibly choose to suppress all messages from certain components. If you decide to suppress all of the messages from a specific component, make sure that none of the messages have any vital importance to the operator. (Suppressing a message does not affect processing by the NetView subsystem.) In an aggressive approach, you could also choose to suppress messages that rarely are important to the operator.

Five lists of suppressible messages are shown below. They are examples that you should review according to the environment in which you might use them. Make sure that you review each message and determine whether your installation should suppress it or not. Notice that the list of message(s) that are considered suppressible in a conservative environment are identified by the full message identifier. In the list of the 'more aggressive' approach to message suppression, the messages are identified by the message prefix followed by an asterisk. The five lists are:

- Conservative set of non-JES or system messages.
- Aggressive set of non-JES or system messages.
- Conservative set of JES2 messages.
- Aggressive set of JES2 messages.
- Conservative set of JES3 messages.

The decision to designate a message eligible for automation processing is independent of the decision to suppress the message. You may want the message suppressed but not processed by automation. The two processes occur independently according to their individual specifications.

The following lists are concerned with suppression, not automation processing.

Conservative list of suppressible non-JES messages

Figure 34 on page 668 shows example messages. Review each message according to the environment in which it might be used. Make sure that you review each message and determine whether your installation should suppress it or not.

```
ARC0100I  SETSYS COMMAND COMPLETED                  HSM
ARC0200I  TRAP IN MODULE XXX
ARC0208I  TRAP FOR ERROR CODE XX
ARC0503I  ALLOCATION ERROR, RETURN CODE=XX
ARC0728I  VTOC FOR VOLUME XX COPIED TO DATA SET
ARC0734I  ACTION=MIGRATE FRVOL=XX TOVOL=XX TRACKS
CSV003I   MODULE NOT FOUND
CSV011I   FETCH FAILED
CSV300I   PROBABLE INVALID RECORD COUNT
DFS035I   BATCH INITIALIZATION COMPLETE
DFS092I   IMS/VS LOG TERMINATED
DFS627I   IMS/VS RESOURCE CLEANUP COMPLETED OR FAILED FOR JOB
DFS629I   IMS TCB ABEND IMS|SYS
DFS2207I  IMS/VS LOG(S) BLOCKSIZE=XXX,BUFNO=YYY
DFS2208I  XXXX LOGGING IN EFFECT ON IMS/VS ZZZZ
DFS2500I  *MDA00 DATABASE/DATASET ALLOCATED/UNALLOCATED
DSI090I   NCCF LOAD FAILED MSG DURING STARTUP
IAT4801   JOB JJJ EXPRESS CANCELLED BY INTERPRETER DSP
ICB402I   VOLUME XXX NOT FOUND IN MSVC INVENTORY
ICB411I   UNABLE TO RESTORE BASE VOLUME XX RECORD
ICH70001I LAST ACCESS AT HH.MM.SS ON YY.DDD
IEA848I   NO DUMP PRODUCED FOR THIS ABEND ...
IEA989I   SLIP TRAP MATCHED
IEA995I   SYMPTOM DUMP OUTPUT
```

Figure 34. Example: Conservative list of suppressible non-JES messages (Part 1 of 2)

```
IEC070I   (VSAM EOB ERROR)
IEC130I   DD STATEMENT MISSING
IEC141I   013-RC (open error)
IEC331I   ABEND MESSAGES
IEC161I   VSAM OPEN ERROR MESSAGES.
IEC705I   TAPE ON device IS label_type
IEC801I   LNA THRESHOLD TRANS= ...
IEC999I   IFG0TC0A...
IEE043I   SYSTEM LOG DATA SET HAS BEEN QUEUED
IEE400I   THESE MESSAGES CANCELLED - XX
IEF097I   USERID ASSIGNED
IEF125I   LOGGED ON
IEF126I   LOGGED OFF
IEF170I   (53 BYTES OF WTP TEXT)
IEF176I   WTR DDD WAITING FOR WORK
IEF188I   PROBLEM PROGRAM ATTRIBUTES ASSIGNED
IEF196I   ALLOCATION
IEF202I   STEP sss WAS NOT RUN BECAUSE OF cde
IEF236I   ALLOCATION FOR job
IEF237I   ALLOCATED TO
IEF287I   DSN DISP VOL SER NOS =
IEF288I   dsn SYSOUTIEF403I   job STARTED
IEF404I   job ENDED
IEF450I   JOB ABEND
IEF452I   JOB NOT RUN JCL ERROR
IEF453I   JOB FAILED JCL ERROR
IEF677I   WARNING MESSAGE(S) FOR JOB JOBNAME ISSUED
IEF722I   JOB FAILED - SECURITY REASON
IEF861I   FOLLOWING RESERVED DATA SET NAMES UNAVAILABLE TO JOB
IEF863I   DSN=DSNAME
IKJ144I   UNDEFINED USER(S) XXX
IKJ572I   USER(S) XXX NOT LOGGED ON
IKJ605I   USER(S) XXX NOT LOGGED ON
IKJ606I   USERID ALREADY LOGGED ON
IKT100I   USERID CANCELLED DUE TO UNCONDITIONAL LOGOFF
IKT108I   USERID RECEIVE ERROR,RPLRTNCD=XX ETC.
IOS050I   CHANNEL DETECTED ERROR
IOS071I   (MIH message)
IST234I   I/O ERROR ON TERMINAL XXX
IST259I   INOP RECEIVED FOR NODENAME
IST521I   GBIND QUEUED FOR COS ETC
IST522I   ERNN ACTIVATION FAILED ...
IST523I   REASON =
IST530I   GBIND PENDING MESSAGE
IST532I   EVENT ID MESSAGE
IST619I   ID FAILED - RECOVERY IN PROGRESS
IST621I   RECOVERY SUCCESSFUL FOR NETWORK NODE
```

Figure 35. Example: Conservative list of suppressible non-JES messages (Part 2 of 2)

Aggressive list of suppressible non-JES messages

For a more aggressive approach to message suppression of non-JES messages, you might choose to add the message prefixes in Figure 36 on page 670 to the preceding conservative list. The following are examples. Review each message according to the environment in which it might be used. Make sure that you review each message and determine whether your installation should suppress it or not.

```
ADM*      ALL ADMPRINT MSGS
AHL*      ALL GTF MESSAGES
DFH1500   CICS INITIALIZATION MESSAGES
DFS0433   NUMBER OF BUFFERS FOR SUBPOOL SIZE NNN INCREASED
DFS0435   NUMBER OF BUFFERS INVALID ON CARD N
DFS0540   DFSZDC00 PROGRAM=, CKPTID=,
DFS0730   UNABLE TO OPEN/CLOSE DATASET WITH DDNAME ...
DFS391I   (Variable message from utility)
DFS826I   BLDL FAILED FOR FOLLOWING DBDs
DFS830I   BLDL FAILED FOR FOLLOWING PSBS;
DFS840I   INDEX ERROR ...
IAT7903   JOB JJJ HELD OUTPUT PURGED
IKF*      ALL COBOL MESSAGESERB*      RMF MESSAGES
IBM*      ALL PLI MESSAGES TO PROGRAMMER
IEC03*    B37-D37-E37 ETC. ABENDS
IEF251I   JOB CANCELLED
IFO*      ALL ASSEMBLER F MESSAGES
IFY*      ALL VS FORTRAN MESSAGES
VPW004*   VPW CLASS DEST STATUS         VIRTUAL PAPER WRITER
VPW012*   VPW IO ERROR(READ) INPUTDCB
VPW016*   VPW NO CATALOG POINTER
```

Figure 36. Example: Aggressive list of suppressible non-JES messages

Conservative list of suppressible JES2 messages

To suppress JES2 messages through MPF processing, you must specify the "$" prefix and the appropriate message identifier. If you have more than one JES2 subsystem, you must specify the message to be suppressed by MPF for each subsystem. Figure 37 on page 671 shows examples. Review each message according to the environment in which it might be used. Make sure that you review each message and determine whether your installation should suppress it or not.

```
$HASP001   R,TEXT VIA $DM
$HASP002   AUTOMATIC COMMANDS HALTED
$HASP100   ON TSORDR/INTRDR/ETC..... RDR
$HASP101   jobname HELD
$HASP110   jobname -- ILLEGAL JOB STATEMENT
$HASP111   jobname -- INVALID /*ROUTE CARD
$HASP112   jobname -- INVALID /*JOBPARM CARD
$HASP113   jobname -- INVALID /*OUTPUT CARD
$HASP114   jobname -- INVALID EXECUTION MODE
$HASP115   jobname -- INVALID /*NETACCT CARD
$HASP116   jobname -- INVALID /*NOTIFY CARD
$HASP117   jobname -- INVALID /*XMIT CARD
$HASP118   jobname -- INVALID /*CONTROL STATEMENT
$HASP120   DEVNAME COMMAND (e.g. INTRDR $VS,'CMD')
$HASP125   SKIPPING FOR JOBCARD
$HASP160   devname - INACTIVE
$HASP165   ENDED ...
$HASP200   XX STARTED ON LINEX
$HASP203   XX DISCONNECTED FROM LINEX
$HASP208   xxx SCHEDULED FOR yyy SNA,VTAM,...
$HASP210   SESSION zzzz LOGGED OFF LINElna
$HASP240   MESSAGES TO INACTIVE SPOOL MEMBER DISCARDED
$HASP244   XX INVALID COMMAND
$HASP249   NX command-text
$HASP250   JOB XXXXXXXX IS PURGED
$HASP301   JOBNAME - DUPLICATE JOB NAME - JOB DELAYED
$HASP309   INIT XX INACTIVE
$HASP310   XXX TERMINATED AT END OF MEMORY
$HASP395   JOBNAME ENDED
$HASP396   XXX TERMINATED
$HASP503   ERROR PROCESSING NCC RECORD RECEIVED
$HASP520   XXX ON devname
$HASP524   devname INACTIVE
$HASP530   XXX jobname on devname
$HASP532   jobname devname RESTARTED
$HASP534   devname INACTIVE
$HASP540   jobname ON devname
$HASP543   jobname devname DELETED
```

Figure 37. Example: Conservative list of suppressible JES2 messages

Aggressive list of suppressible JES2 messages

For a more aggressive approach to message suppression of JES2 messages, you might chose to add the messages in Figure 38 to the preceding conservative list. These messages are examples. Review each message according to the environment in which it might be used. Make sure that you review each message and determine whether your installation should suppress it or not.

```
$HASP150   OUTGRP ON device
$HASP308   jobname - ESTIMATED TIME EXCEEDED
$HASP373   jobname STARTED - INIT xx
$HASP375   jobname ESTIMATE EXCEEDED BY
$HASP406   jobname WAS EXECUTING
```

Figure 38. Example: Aggressive list of suppressible JES2 messages

Conservative list of suppressible JES3 messages

With JES3, there is no particular category of messages that are generally suppressible. Therefore, the list contains only specific messages identified by complete message identifiers. Figure 39 on page 672 shows examples. Review each message according to the environment in which it might be used. Make sure that

you review each message and determine whether your installation should suppress it or not.

```
IAT1600    LINES EXCEEDED
IAT2000    JOB SELECTED
IAT2002    LSTOR=  ALLOC= ...
IAT2003    MPAINIT=  DI= ...
IAT2006    PREMATURE JOB TERM
IAT2007    GMS CONNECT - JOB JOBNO ...
IAT5200    JOB IN SETUP ON MAIN
IAT5918    MAIN JES3 V  (VERIFY DESCRIPTION)
IAT6101    JOB IS PRTY (job read in)
IAT6108    JOB (NOTIFY TEXT)
IAT6118    CARDS FLUSHED
IAT6160    JOB NET DJNET NOW ENTERING SYSTEM
IAT6201    JOB JOBNO=JJJ JOBS ...
IAT6306    JOB IS DSPNAME
IAT7001    JOB IS ON WRITER
IAT7007    JOB IS ON WRITER, PURGED
IAT7100    (INTERCOM cmd from dsp)
IAT7120    IO ERROR ON CONSOLE
IAT7310    NEW DJNET HAS COMPLETED
IAT7450    JOB PURGED
IAT7530    IO ERROR ON LINE
IAT9123    DATA RECEPTION ACTIVE ON LINE
IAT9124    JOB RECEPTION ACTIVE ON LINE
IAT9127    JOB IS FROM NODENAME
IAT9190    JOB IS BEING SENT ON LINE
IAT9191    JOB SENT TO NODE
```

Figure 39. Example: Conservative list of suppressible JES3 messages

Examples of MPFLSTxx members

The contents of MPFLSTxx depend on the goals of your specific installation. The following five examples show possible contents of MPFLSTxx.

Example 1: Figure 40 on page 673 assumes that you want to create a parmlib member named MPFLST7C to:

- Suppress some frequently issued JES2, MONITOR, and general messages that are of no interest to the operator or to automation processing.
- Set the color and highlighting attributes of messages to display eventual action messages in pink and non-action messages issued by problem programs in yellow.

The SET MPF=7C command establishes this list of messages as those that are to be suppressed and ignored by an automation subsystem, and also sets the color and highlighting attributes for eventual action and non-action messages. The system uses the IBM-supplied defaults for all other message types or console fields.

```
$HASP100
$HASP101
$HASP150
$HASP165
$HASP200
$HASP250
$HASP309
$HASP373
$HASP375
$HASP395
IEC130I
IEC705I
IEF125I
IEF126I
IEF165I
IEF170I
IEF236I
IEF237I
IEF403I
IEF404I
.MSGCOLR  EVETACTN(P,N),PPMSG(Y,N)
```

Figure 40. Example: creating parmlib member MPFLST7C

Example 2: Figure 41 assumes that you want to create a parmlib member named MPFLSTDF to:

- Reset the color, highlighting, and intensity attributes to the IBM-supplied default values.
- Suppress, retain, and mark eligible for automation processing the same messages as specified in the previously active concatenation of MPFLSTxx parmlib members.

```
.MSGCOLR DEFAULT
.MSGIDS NOCHANGE
```

Figure 41. Example: creating parmlib member MPFLSTDF

Example 3: Figure 42 assumes that you want to create a parmlib member named MPFLSTRV to:

- Display all messages in reverse video (that is, the characters are to appear in black on a colored background).
- Display eventual action messages in pink and non-action messages issued by problem programs in yellow.
- Suppress, retain, and mark eligible for automation processing the same messages as specified in the previously active concatenation of MPFLSTxx parmlib members.

```
.MSGCOLR URGATTN(R,R),IMEDACTN(W,R),EVETACTN(P,R)
.MSGCOLR GENMSG(G,R),PPMSG(Y,R),SELPEN(B,R)
.MSGCOLR INSTRERR(W,R),ENTRYARA(G,R),WARNLGEN(B,R)
.MSGCOLR WARNRGEN(B,R),WARNRURG(R,R),OOLCNTL(T,R)
.MSGCOLR OOLLABEL(T,R),OOLDATA(G,R)
.MSGIDS NOCHANGE
```

Figure 42. Example: creating parmlib member MPFLSTRV

Example 4: Figure 43 shows how to create a parmlib member named MPFLST18 to:

- Display all messages of the form IECxxxx and have control pass to an installation-supplied WTO/WTOR installation exit routine each time the system prepares to issue such a message.
- Suppress certain IEFxxxx messages and have control pass to an installation-supplied WTO/WTOR installation exit routine each time the system prepares to issue one of the messages.
- Allow action message IEE601E to be displayed at the operator console but not be retained by the action message retention facility.

```
IEC*,SUP(NO),USEREXIT(usrexit1) /*usrexit1 handles IEC messages*/
IEF170I,SUP(YES),USEREXIT(usrexit2)
IEF236I,SUP(YES),USEREXIT(usrexit2)
IEF237I,SUP(YES),USEREXIT(usrexit2)
IEF403I,SUP(YES),USEREXIT(usrexit2)
IEE601E,SUP(NO),RETAIN(NO)         /*display only*/
```

Figure 43. Example: creating parmlib member MPFLST18

Example 5: Figure 44 on page 675 shows how to create a parmlib member named MPFLST02 to:

- Specify that messages not identified in this member are not eligible for automation processing.
- Establish the default for specific messages so that they will be eligible for automation processing, suppressed, retained, and passed to an installation exit. Override this default for one particular message and have it displayed.
- Establish the default for specific messages so that they will be eligible for automation processing, suppressed and retained. Display one specific message.
- Establish the default for specific messages so they will not be retained. They will be displayed and passed to an installation exit for processing. The installation exit must turn off automation processing and use the token to select a console to reroute the message to.
- The last .DEFAULT statement with no options specified sets the defaults for the subsequent list of messages to AUTO(NO) RETAIN(YES) and SUP(YES) and the messages will be passed to IEAVMXIT, if it exists.

```
/*
/* DO NOT AUTOMATE THOSE MESSAGES NOT SPECIFIED IN MPFLST
/*
.NO_ENTRY,AUTO(NO)
/*
/* AUTOMATE CONSOLE BUFFERS AND USE MPF EXIT
/*
.DEFAULT,AUTO(YES),USEREXIT(COMMTASK)
IEA405E                        /*WTO BUFFER SHORTAGE - 80% FULL*/
IEA404A,SUP(NO)                /*SEVERE WTO BUFFER SHORTAGE - 100% FULL
                                  ALSO TELL OPER WHY JOBS IN WAIT*/
IEA406I                         /*TO BUFFER SHORTAGE RELIEVED*/
/*
/* AUTOMATE PRODUCTION JOB STATUS MESSAGES
.DEFAULT,AUTO(YES)
IEF402I,SUP(NO)                 /*JJJ FAILED IN ADDRESS SPACE X
                                   SYSTEM ABEND SXXX  REASON - RC
                                   NOTIFY OPER OF FAILURE /*
IEF403I                        /*JJJ-STARTED */
IEF404I                        /*JJJ-ENDED   */
/*
/* REROUTE THE TAPE MESSAGES
/*
/*   - AUTO TOKEN USED BY EXIT TO SELECT CONSOLE FOR MESSAGE
/*   - EXIT MUST TURN OFF AUTOMATION (IMPLIED BY TOKEN USE)
/*   - REROUTE EXIT WILL SEND MESSAGE TO PROPER CONSOLE
/*
.DEFAULT,RETAIN(NO),USEREXIT(REROUTE),AUTO(TAPEPOOL),SUP(NO)
IEF233*                     /*  M DDD,SER,LABEL,JJJ,SSS,DSN */
IEF234E                     /*  K/D/R DDD,SER PVT/PUB/DTR, JJJ,SSS,SPACE*/
IEC400A                     /*  M DDD,SER/DSN*/
IEC401A                     /*  F DDD,SER/DSN*/
IEC402D                     /*  F DDD,SER/DSN*/
IEC403A                     /*  M DDD,SER*/
IEC501*                     /*  M DDD,SER,LABEL,DENSITY,JJJ,SSS,DSN*/
IEC507D                     /*  E DDD,SER,JJJ,SSS,DSN*/
IEC509A                     /*  F DDD,SER,JJJ,SSS,DSN*/
/*
/*       RESET DEFAULT FOR OLD LIST
/*       (FROM A DIFFERENT MPFLSTxx MEMBER)
.DEFAULT
 IEC*
 IEF170I
 IEF236I
 IEF237I
 IEF403I
 IEE601E
```

Figure 44. Example: creating parmlib member MPFLST02

Chapter 72. MSGFLDxx (message flood automation parameters)

Use the MSGFLDxx member to define your installation policy for controlling message flooding situations, where xx should be alphabetic, numeric, or national characters. Special characters are not supported.

The following statements are provided:

- The comment statement indicates commentary information.
- The msgtype statement indicates the type of message that the parameters apply to.
- The DEFAULT statement specifies the default action to be taken for a specific address space that exceeds the job threshold message rate (JOBTHRESH / INTVLTIME) or a specific message that exceeds the message threshold message rate (MSGLIMIT / INTVLTIME). The built-in defaults apply otherwise.
- The DEFAULTCMD statement specifies the default command that will be issued if a CMD action has been specified for the address space.
- The JOB statement identifies up to 64 specific jobs for which specific actions are to be taken if REGULAR or ACTION messages from the job are involved in a message flooding situation. During intensive mode processing, if the specified job requires action to be taken against it, the actions will be taken from those specified on the JOB statement. You can use the JOB statement to override the DEFAULT (or built-in) actions for this specific job.
- The MSG statement defines up to 1024 specific messages for which specific actions are to be taken if the message is involved in a message flooding situation.

Parameter in IEASYSxx (or issued by the operator)

There is no IEASYSxx parameter to set MSGFLD=xx, but the MSGFLDxx suffix can be specified in CONSOLxx. It can also be set at any time using the SET MSGFLD= command.

When MSTRJCL=(xx[,L]) is specified on IEASYSxx, JCL messages issued for master scheduler processing are routed to consoles that receive operator information (routing code 2) or system/error maintenance (routing code 10) messages. Use MSTRJCL=(xx[,L]) only for debugging purposes. See Chapter 54, "IEASYSxx (system parameter list)," on page 431.

SET MSGFLD=xx

The two characters xx indicate the MSGFLDxx member that contains the parameters and actions for message flood automation processing. You can use alphabetic, numeric, or national characters to specify the two characters *xx*. Do not use special characters.

Syntax rules for MSGFLDxx

The following syntax rules apply to MSGFLDxx:

- All statements must begin in column 1 or they are ignored. (Statements with a blank in column 1 are ignored.)

- Statements are 80 columns in length, but only the first 71 columns are processed. Columns 72-80 are ignored. Specifications extending beyond column 71 are truncated without detection or warning.
- Any statement may be repeated if all of its parameters will not fit on a single statement.
  - If a parameter is specified on more than one statement, the specification on the last statement is the one that is used.
  - If a parameter is specified more than once on a statement, the rightmost specification is the one that is used.
  - If mutually-exclusive parameters (such as LOG/NOLOG) are specified on more than one statement, the parameter specified on the last statement is the one that is used.
  - If mutually-exclusive parameters are specified on a single statement, the rightmost specification is the one that is used.
- An asterisk * in column 1 indicates that the rest of the record is a comment.
- The /* characters in column 1 and 2 also indicate that the rest of the record is a comment. You do not need to supply the */ characters but, if you supply them, they will be accepted. The comment is considered ended at the end of the record; it does not extend to the next record.

IBM-supplied default for MSGFLDxx

If message flood automation is enabled by either:

- Specifying MSGFLD=(NONE,(ON)) on a CONSOLxx INIT statement, or
- Issuing a SETMF ON command before a MSGFLDxx parmlib member has been read using a SET MSGFLD=xx command,

the following built-in defaults are used :

- REGULAR parameters:

```
REGULAR MSGTHRESH=50
REGULAR JOBTHRESH=20
REGULAR INTVLTIME=1
REGULAR SYSIMTIME=2
REGULAR JOBIMTIME=2
```

  REGULAR actions: `LOG, AUTO, NODISPLAY, NOCMD`

- ACTION parameters:

```
ACTION MSGTHRESH=50
ACTION JOBTHRESH=20
ACTION INTVLTIME=1
ACTION SYSIMTIME=2
ACTION JOBIMTIME=2
```

  ACTION actions: `LOG, AUTO, NODISPLAY, NOCMD, NORETAIN`

- SPECIFIC parameters:

```
SPECIFIC MSGTHRESH=50
SPECIFIC MSGLIMIT=20
SPECIFIC INTVLTIME=1
SPECIFIC SYSIMTIME=2
SPECIFIC MSGIMTIME=2
```

  SPECIFIC actions:`LOG, AUTO, NODISPLAY, NORETAIN,NOIGNORE`

- SPECIFIC message set:

```
IOS000I NOLOG,NOAUTO,NODISPLAY
IOS002A NOLOG,NOAUTO,NODISPLAY
IOS017I NOLOG,NOAUTO,NODISPLAY
IOS107I NOLOG,NOAUTO,NODISPLAY
```

```
IOS109E NOLOG,NOAUTO,NODISPLAY
IOS251I NOLOG,NOAUTO,NODISPLAY
IOS291I NOLOG,NOAUTO,NODISPLAY
IOS444I NOLOG,NOAUTO,NODISPLAY
IOS450E NOLOG,NOAUTO,NODISPLAY
IEA476E NOLOG,NOAUTO,NODISPLAY
IEA491E NOLOG,NOAUTO,NODISPLAY
IEA494I NOLOG,NOAUTO,NODISPLAY
IEA497I NOLOG,NOAUTO,NODISPLAY
```

This message set was chosen to handle message floods that can occur during the GDPS HyperSwap IPL processing. Note that the IEA497I message has continuation messages that are issued as separate single line messages, without message IDs. These separate messages are not handled.

When a MSGFLDxx parmlib member is read, message flood automation first replaces any parameters that it was using with the built-in default values. It then replaces those values with any values specified in the parmlib member. This ensures that message flood automation always has valid policy values that it can use. The exception to this are the built-in SPECIFIC messages which are automatically deleted by the first MSGFLDxx PARMLIB member read.

Statements/parameters for MSGFLDxx

MSGFLDxx provides the following statement types:

- comment statement
- msgtype statement
- DEFAULT statement
- DEFAULTCMD statement
- JOB statement
- MSG statement

Syntax of the comment statement

A comment statement is indicated by placing an asterisk (*) in column 1 or by placing an open comment specification (/*) in columns 1 and 2. There is no need to place a closing comment specification on the statement although it is good practice to do so.

Syntax and parameters of the msgtype statement

A msgtype statement identifies the type of message that any following DEFAULT, DEFAULTCMD, JOB, and MSG statements apply to. The msgtype statement has the following syntax form.

```
msgtype keyword=value[,keyword=value]
```

msgtype

The message type, which is one of the following specifications:

- **REGULAR**
- **ACTION**
- **SPECIFIC**

keyword

The parameter name, which is one of the following specifications:

- **INTVLTIME**

- **JOBIMTIME** (not supported for msgtype SPECIFIC)
- **JOBTHRESH** (not supported for msgtype SPECIFIC)
- **MSGIMTIME** (not supported for msgtype REGULAR and ACTION)
- **MSGLIMIT** (not supported for msgtype REGULAR and ACTION)
- **MSGTHRESH**
- **SYSIMTIME**

value
The parameter value.

- For JOBTHRESH, MSGLIMIT and MSGTHRESH: a positive, non-zero integer count of messages in the range 1 to 999999999
- for INTVLTIME: a positive, non-zero integer time in seconds in the range 1 to 999999999
- For SYSIMTIME, JOBIMTIME and MSGIMTIME: a positive, non-zero floating point time in seconds in the range 0.000001 to 16777215.0

Note:

1. Floating point fractions such as 0.1 might not be stored exactly because of limitations of the floating point format.
2. For msgtype REGULAR and ACTION, the JOBTHRESH value must be less than the MSGTHRESH value.
3. For msgtype SPECIFIC, the MSGLIMIT value must be less than the MSGTHRESH value.

Syntax rules:

- The *msgtype* specification must begin in column 1 and must be separated from the first keyword by a single blank character.
- The *keyword* and the value are separated by an equal (=) character. There must be no blank characters on either side of the equal character.
- *keyword=value* pairs are separated by a comma. There must be no blank characters on either size of the comma.
- If *value* is a floating point number, it does not need to contain a decimal point unless a fractional value is being specified. The specifications 1.0 and 1. and 1 are equivalent as are 0.25 and .25.
- Comment text may follow the last *keyword=value* pair. There must be at least one blank character between the value and the beginning of the comment text. There is no need to preface the comment text with either an asterisk (*) or open comment specification (/*).

Example:

```
REGULAR MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1 comment
REGULAR SYSIMTIME=0.125,JOBIMTIME=0.25
```

Syntax and parameters of the DEFAULT statement

The DEFAULT statement specifies the default action to be taken for a specific address space that exceeds the job threshold message rate (JOBTHRESH / INTVLTIME) or a specific message that exceeds the message threshold message rate (MSGLIMIT / INTVLTIME). The built-in defaults apply otherwise. The DEFAULT statement has the following syntax form.

```
DEFAULT action[,action]
```

The DEFAULT statement applies to the previously specified *msgtype*. Zero or more DEFAULT statements should follow each *msgtype* statement.

action
The default action to be taken. It can be one of the following specifications:

LOG | NOLOG
Determines whether or not the message is written to the SYSLOG or OPERLOG.

AUTO | NOAUTO
Determines whether or not the message is queued to an EMCS console for automation.

DISPLAY | NODISPLAY
Determines whether or not the message is queued to a console for display.

CMD | NOCMD
Determines whether or not the command defined by the DEFAULTCMD statement is issued. Not valid for *msgtype* SPECIFIC.

RETAIN | NORETAIN
Determines whether or not the message is retained by the action message retention facility (AMRF). Not valid for *msgtype* REGULAR.

IGNORE | NOIGNORE
Determines whether or not message flood automation is to process the message. Not valid for *msgtype* ACTION or REGULAR.

Note: Specifying NOLOG, NODISPLAY, and NOAUTO together results in the message being deleted. Deleted messages are not sent to the console address space or to other systems in the sysplex.

Syntax rules:

- The DEFAULT specification must begin in column 1 and must be separated from the first action by a single blank character.
- The *action* parameter specifications are separated by a comma. There must be no blank characters on either size of the comma.
- Comment text may follow the last action. There must be at least one blank character between the action and the beginning of the comment text. There is no need to preface the comment text with either an asterisk (*) or open comment specification (/*).

Example:

```
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD
```

Syntax and parameters of the DEFAULTCMD statement

The DEFAULTCMD statement specifies the default command that will be issued if a CMD action has been specified for the address space. The DEFAULTCMD statement has the following syntax form.

```
DEFAULTCMD 'cmdchr[ASIDchar],cmdtext'
```

The DEFAULTCMD statement applies to the previously specified *msgtype*. Zero or more DEFAULTCMD statements should follow each *msgtype* statement.

cmdchr
A single character that can be used later in the *cmdtext* to indicate where the jobname should be substituted. Only a single substitution is performed.

ASIDchar
An optional single character that can be used later in the *cmdtext* to indicate where an address space identifier (ASID) should be substituted. *ASIDchar* must be different than the *cmdchr*. Only a single substitution is performed. The ASID is provided in printable hexadecimal, suitable for use in the CANCEL command.

cmdtext
Any valid z/OS or subsystem command. The maximum length is limited to 56 characters.

Note:

1. The DEFAULTCMD specification does not apply to *msgtype* SPECIFIC.
2. Your security product might prevent you from issuing the command that you have specified on the DEFAULTCMD statement. Because the command is issued from the address space that is causing the flood, the address space causing the flood must be authorized to issue the command. In terms of RACF, this means that the address space must have sufficient access authority. For example, to issue the MVS CANCEL command, UPDATE access authority is required.

Syntax rules:

- The DEFAULTCMD specification must begin in column 1 and must be separated from the opening quotation mark character of the command specification by a single blank character.
- The *cmdchr* character should be chosen so that it does not conflict with any special characters or punctuation characters used in the *cmdtext*.
- The *cmdchr* character must not have blank characters either before it or after it.
- The *cmdchr* is separated from the *cmdtext* by a comma. There must be no blank characters on either side of the comma. Message flood automation treats whatever follows the comma as the *cmdtext*.
- The *cmdtext* is any valid z/OS or subsystem command. There is no practical way for message flood automation to check the correctness of this command, so it is important that the command be specified correctly. An improperly specified command will result in an error when message flood automation attempts to issue it.

  The *cmdtext* may include special characters and punctuation characters. If single-quoted characters are used as part of the *cmdtext* specification, two single-quoted characters must be specified for each desired single-quoted character in the command text that will be used.
- Comment text may follow the closing quotation mark character. There must be at least one blank character between the quotation mark and the beginning of the comment text. There is no need to preface the comment text with either an asterisk (*) or open comment specification (/*).

Example:

```
DEFAULTCMD '#,CANCEL #' z/OS cancel job
DEFAULTCMD '#,*CANCEL,SY1,#' JES3 cancel job
DEFAULTCMD '#,SEND ''# is issuing a lot of messages'',BRDCST'
DEFAULTCMD '&%,CANCEL &,A=%'
```

Syntax and parameters of the JOB statement

JOB statements identify up to 64 specific jobs for which specific actions are to be taken if REGULAR or ACTION messages from the job are involved in a message flooding situation. During intensive mode processing, if the specified job requires action to be taken against it, the actions will be taken from those specified on the JOB statement. You can use the JOB statement to override the DEFAULT (or built-in) actions for this specific job. The JOB statement has the following syntax form.

```
JOB jobname [action][,action]
```

The JOB statement applies to the previously specified REGULAR or ACTION *msgtype* statements. Zero to 64 JOB statements should follow each REGULAR or ACTION *msgtype* statement.

jobname
The job name, which is 1 to 8 characters in length. You can use wildcard characters * and % when you specify a job name:

- An asterisk (*) matches a substring of any characters for any length (0 to n).
- A percent sign (%) matches a single character.

Note: Message flood automation assigns the job name NONAME to operating system services that issue messages that are not attributable to a particular job. The system assigns the name IEESYSAS to system services that occupy their own address spaces. Message flood automation cannot distinguish the difference between started tasks running in different address spaces with the same name.

action
If specified, is one or more of the actions as defined for the DEFAULT statement. If an action is not specified, the DEFAULT or built-in default actions are used.

Syntax rules:

- The JOB specification must begin in column 1 and must be separated from the *jobname* by a single blank character.
- The *jobname* is separated from the first *action*, if specified, by a single blank character. More than one blank character causes any following string to be treated as comment text.
- The *action* parameter specifications are separated by a comma. There must be no blank characters on either size of the comma.
- Comment text may follow the last action. There must be at least one blank character between the action and the beginning of the comment text. If *action* is not specified, there must be at least two blank characters. There is no need to preface the comment text with either an asterisk (*) or open comment specification (/*).

Example:

```
JOB PROD1001
JOB PROD1002   comment: note >1 blank required before comment
JOB PROD1003 LOG,NOAUTO
JOB PROD1004 LOG,NOAUTO comment: note only 1 blank required
JOB AOC%NV* LOG,AUTO,RETAIN
JOB LLA* LOG,AUTO
```

Syntax and parameters of the MSG statement

MSG statements define up to 1024 specific messages for which specific actions are to be taken if the message is involved in a message flooding situation. The MSG statement has the following syntax form.

```
MSG msgid [action][,action]
```

The MSG statement applies to the previously specified SPECIFIC *msgtype* statement. One to 1024 MSG statements should follow the SPECIFIC *msgtype* statement.

msgid
: A 1-to-10 character message identifier. Up to the first 10 characters of the first blank-delimited token in the message is treated as the message identifier.

  Message flood automation treats everything in the message up to the first blank as the message ID. If the message ID in the message is followed immediately by a special character before the first blank, the special character is treated as part of the message ID and must be specified for a match to occur. Also, message flood automation can handle message IDs with embedded special characters without problem, for example, message IDs that include embedded "-" characters.

  Note: The *msgid* identifier does not support wildcard specifications.

action
: If specified, is one or more of the actions as defined for the DEFAULT statement. CMD is not supported. If an action is not specified, the DEFAULT or built-in actions are taken for the message. You can use a MSG statement to override the DEFAULT or built-in actions to be taken for a specific message. If IGNORE is specified, the message is ignored by message flood automation and no action is taken for the message even if the message threshold is exceeded and other actions are specified.

Syntax rules:

- The MSG specification must begin in column 1 and must be separated from the *msgid* by a single blank character.
- The *msgid* is separated from the first action, if specified, by a single blank character. More than one blank character causes any following string to be treated as comment text.
- *action* specifications are separated by a comma. There must be no blank characters on either size of the comma.
- Comment text may follow the last action. There must be at least one blank character between the action and the beginning of the comment text. If *action* is not specified, there must be at least two blank characters. There is no need to preface the comment text with either an asterisk (*) or open comment specification (/*).

Example:

```
MSG IAT1001
MSG IAT1002   comment: note >1 blank required before comment
MSG IAT1003 LOG,NOAUTO
MSG IAT1004 LOG,NOAUTO comment: note only 1 blank required
MSG IOS000I NOLOG,NODISPLAY
MSG DSI064A NOLOG,NODISPLAY,NORETAIN
```

Suggested MSG specifications to handle specific conditions:

Provide MSG specifications for the following messages that can be produced if an ESCON Director or FICON switch fails:

```
IOS001E
IOS050I
IOS051I
IOS071I
IOS251I
IOS444I
IOS450E
```

Provide MSG specifications for the following messages which can be produced if a device fails:

```
IOS003A
```

Provide MSG specifications for the following messages if you perform HyperSwaps or are in a GDPS environment:

```
IOS000I NOLOG,NOAUTO,NODISPLAY
IOS002A NOLOG,NOAUTO,NODISPLAY
IOS017I NOLOG,NOAUTO,NODISPLAY
IOS107I NOLOG,NOAUTO,NODISPLAY
IOS109E NOLOG,NOAUTO,NODISPLAY
IOS251I NOLOG,NOAUTO,NODISPLAY
IOS291I NOLOG,NOAUTO,NODISPLAY
IOS444I NOLOG,NOAUTO,NODISPLAY
IOS450E NOLOG,NOAUTO,NODISPLAY
IEA476E NOLOG,NOAUTO,NODISPLAY
IEA491E NOLOG,NOAUTO,NODISPLAY
IEA494I NOLOG,NOAUTO,NODISPLAY
IEA497I NOLOG,NOAUTO,NODISPLAY
```

Note:

1. Specifying NOLOG, NOAUTO and NODISPLAY together causes the message to be completely deleted.
2. While NOAUTO was chosen as the default for the above list of messages to prevent message floods from overwhelming automation, if your automation has dependencies on any of these messages during a HyperSwap (including HyperSwaps in a GDPS environment), you might consider specifying AUTO for the message so that it can be detected by your automation. Refer to the GDPS documentation for further suggestions on use of Message Flood Automation in that environment.
3. If your automation has dependencies on any of these messages during a HyperSwap, you should specify AUTO for the message so that it can seen by your automation.

Syntax example

The values in the following MSGFLDxx parmlib member are for illustration purposes only. You should determine the values that are appropriate for your system. A sample MSGFLDxx parmlib member (named CNZZMFXX) is provided as an example in SYS1.SAMPLIB.

```
/*--------------------------------------------------------------------*/
/* Sample MSGFLDxx PARMLIB member. */
/*--------------------------------------------------------------------*/
/*--------------------------------------------------------------------*/
/* REGULAR message specifications. */
/*--------------------------------------------------------------------*/
REGULAR MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1
REGULAR SYSIMTIME=2,JOBIMTIME=2
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD
DEFAULTCMD '&,CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO
JOB LLA* AUTO
JOB ZAP1 CMD
/*--------------------------------------------------------------------*/
/* ACTION message specifications. */
/*--------------------------------------------------------------------*/
ACTION MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1
ACTION SYSIMTIME=2,JOBIMTIME=2
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD,NORETAIN
DEFAULTCMD '&,CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO,RETAIN
JOB LLA* AUTO
JOB ZAP2 CMD
/*--------------------------------------------------------------------*/
/* SPECIFIC message specifications. */
/*--------------------------------------------------------------------*/
SPECIFIC MSGTHRESH=50,INTVLTIME=1
SPECIFIC SYSIMTIME=2
SPECIFIC MSGIMTIME=2
SPECIFIC MSGLIMIT=20
DEFAULT LOG,NOAUTO,NODISPLAY,NORETAIN
MSG IOS001E
MSG IOS003A
MSG IOS050I
MSG IOS051I
MSG IOS071I
MSG IOS251I
MSG IOS444I
MSG IOS450E
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
*++++++++++++++++++++ end of member ++++++++++++++++++++++++++++++++*
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
```

Exempting messages from processing by message flood automation

If you have automation that assumes that suppression (NODISPLAY) implies no automation (NOAUTO), and the automation is critically dependent on specific messages (for example to signal that a program is UP or DOWN), you may want to code a SPECIFIC message entry for those messages. Specify an action of IGNORE which will cause Message Flood Automation to ignore the message even if the specific message threshold is exceeded and other actions have been specified (or defaulted) for the message. Providing a SPECIFIC message entry keeps the message out of both ACTION message and REGULAR message processing.

Chapter 73. MSTJCLxx (master scheduler JCL)

The MSTJCLxx parmlib member contains the master scheduler job control language (JCL) that controls system initialization and processing. You can place the master scheduler JCL in MSTJCLxx as an alternative to keeping it in the MSTJCLxx module in SYS1.LINKLIB.

The advantage to placing the master JCL in the MSTJCLxx parmlib member is that it is easier to make changes to the master JCL. If you specify the master JCL in the MSTJCLxx module in linklib, you must reassemble the module and link-edit it into the system each time a change is made. Specifying the master JCL in parmlib eliminates the need to reassemble and link-edit the module.

If the system finds a MSTJCLxx parmlib member, it uses the JCL in that member to start the master scheduler address space. If the system does not find a MSTJCLxx parmlib member, it uses the JCL in the MSTJCLxx module in linklib.

For more information about how to change the master JCL or specify alternate versions of the master JCL data set, see "Understanding the master scheduler job control language" on page 10.

Parameter in IEASYSxx (or supplied by the operator)

```
MSTRJCL=(xx[,L])
```

The two characters (A-Z, 0-9, @, #, or $), represented by xx, are appended to MSTJCL to identify one of the following:

- The MSTJCLxx member of SYS1.PARMLIB.
- The MSTJCLxx module in SYS1.LINKLIB.

If the MSTRJCL parameter is not specified, the system uses the MSTJCL00 parmlib member. If the system does not find the MSTJCL00 parmlib member, it uses the default master JCL in the MSTJCL00 module in SYS1.LINKLIB.

Performance implications

None.

Support for system symbols

You can specify symbols in MSTJCLxx. Keep in mind that the system does not process symbols in MSTJCLxx in the same way that it processes symbols in parmlib members. Because MSTJCLxx contains JCL, the system processes symbols in MSTJCLxx during JCL processing. The results of symbolic substitution reflect the substitution rules that are in effect during JCL processing.

For details about using system symbols and JCL symbols in JCL, see *z/OS MVS JCL Reference*.

Syntax rules for MSTJCLxx

The syntax rules for the MSTJCLxx parmlib member are the same as for the master JCL in the MSTJCLxx module in linklib. For details about the syntax of the master JCL, see “Writing your own master scheduler JCL” on page 14.

The following example shows how the IBM-supplied IEESMJCL member of SYS1.SAMPLIB might appear in the MSTJCLxx module in linklib.

```
MSTJCL05 CSECT
         DC    CL80'//MSTJCL05 JOB MSGLEVEL=(1,1),TIME=1440'
         DC    CL80'//         EXEC PGM=IEEMB860'
         DC    CL80'//STCINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//TSOINRDR DD SYSOUT=(A,INTRDR)'
         DC    CL80'//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR'
         DC    CL80'//IEFPARM  DD DSN=SYS1.PARMLIB,DISP=SHR'
         DC    CL80'//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR'
         DC    CL80'//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR'
         DC    CL80'/*'
         END
```

The following example shows how IEESMJCL might appear in the MSTJCLxx parmlib member.

```
   //MSTJCL05 JOB MSGLEVEL=(1,1),TIME=1440
   //         EXEC PGM=IEEMB860
   //STCINRDR DD SYSOUT=(A,INTRDR)
   //TSOINRDR DD SYSOUT=(A,INTRDR)
   //IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR
   //IEFPARM  DD DSN=SYS1.PARMLIB,DISP=SHR
   //SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
   //SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
```

Note: The only Brodcast data set name that TSO/E recognizes on the SYSLBC DD statement is SYS1.BRODCAST.

Comments are indicated by characters "//*" in columns 1, 2, and 3. Code the comments in columns 4 through 80. Do not continue a comment statement using continuation conventions. Instead, code additional comment statements. The following example shows how a comment is coded:.

```
    //* THE COMMENT STATEMENT CANNOT BE CONTINUED.
    //* BUT IF YOU HAVE A LOT TO SAY, YOU CAN FOLLOW
    //* A COMMENT STATEMENT WITH MORE COMMENT STATEMENTS.
```

IBM-supplied default for MSTJCLxx

There is no default MSTJCLxx parmlib member. The default master JCL resides in the IBM-supplied MSTJCL00 module in SYS1.LINKLIB.

Statements/parameters for MSTJCLxx

The JCL statements in MSTJCLxx are described in “Writing your own master scheduler JCL” on page 14.

Chapter 74. NUCLSTxx (customizing the nucleus region)

The NUCLSTxx member allows you to load installation-supplied modules into the system's DAT-ON nucleus region at IPL-time. You can use NUCLSTxx to:

- Add your installation's modules to the nucleus region.
- Delete nucleus-resident modules and replace them with alternate versions of the modules.

NUCLSTxx saves you from having to link-edit your installation's nucleus-resident routines (such as installation-written SVCs) into the IEANUC0x member of SYS1.NUCLEUS.

Adding and deleting modules

The modules to be added to the nucleus region, or deleted from it, must reside in members of SYS1.NUCLEUS. To add or delete modules, simply specify the members on INCLUDE or EXCLUDE statements in NUCLSTxx.

On each INCLUDE statement, specify one member of SYS1.NUCLEUS to be included in the nucleus region at IPL-time. Use INCLUDE for code that needs to reside in the nucleus region, such as device support routines, and installation-defined SVCs.

On each EXCLUDE statement, specify one member of SYS1.NUCLEUS to be excluded from the nucleus region at IPL-time. Use EXCLUDE statements to exclude members that you are replacing with INCLUDE statements. You can also use the EXCLUDE statement to exclude modules specified in nucleus module lists (NMLs) or module list tables (MLTs).

You cannot specify more than one member name for each INCLUDE or EXCLUDE statement. However, you can specify multiple INCLUDEs and EXCLUDEs in NUCLSTxx, in any order.

If you use INCLUDE and EXCLUDE statements to replace a member of SYS1.NUCLEUS in the nucleus region with another member, the new member must have a unique name.

Contradictory specifications

If you specify for the same member name both an INCLUDE and an EXCLUDE statement, the system uses the EXCLUDE statement.

Restrictions

Note the following restrictions when using NUCLSTxx:

- Multiple CSECT load modules must be link-edited with the scatter (SCTR) attribute of the linkage editor before they can be added to the nucleus region. If this is not done, the system enters a non-restartable wait state at system initialization.
- Do not use NUCLSTxx to load or delete an alternate nucleus (IEANUC0x). You can select an alternate nucleus member by specifying the suffix identifier of the alternate IEANUC0x member on either the NUCLEUS statement in the LOADxx

member of SYS1.PARMLIB, or on the LOAD parameter during system initialization. For more information, see "Specifying an alternate nucleus" on page 9.

NUCLSTxx compared with NMLDEF

NUCLSTxx is provided as an alternative to the NMLDEF macro. With NMLDEF, you identify modules to be added to the nucleus by creating a set of tables, called nucleus module lists (NMLs). You can use either NUCLSTxx or NMLDEF to customize the nucleus. However, you might prefer to use NUCLSTxx for the following reasons:

- NUCLSTxx is easier to code than NMLDEF.
- NUCLSTxx resides in SYS1.PARMLIB (or SYSn.IPLPARM — wherever LOADxx resides). NMLs must reside in SYS1.NUCLEUS.
- You can use different NUCLSTxx members to load different SYS1.NUCLEUS members into the nucleus region. Therefore, NUCLSTxx provides you with greater flexibility than NMLDEF in customizing the nucleus.

For information about the NMLDEF macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Relationship to the LOADxx member

Placement of NUCLSTxx

The NUCLSTxx member must reside in the same data set as the LOADxx member. This member can reside in either SYS1.PARMLIB or SYSn.IPLPARM, depending on how the installation defined its I/O configuration.

NUCLSTxx specification in LOADxx member

In LOADxx, code a NUCLST statement to specify the NUCLSTxx member to be used. On the NUCLST statement, you can optionally specify whether the system loads a wait state if any of the INCLUDE statements in the NUCLSTxx member specify a member that cannot be found in SYS1.NUCLEUS. The system does **not** load a wait state by default and continues to IPL.

For information about installing LOADxx, and on specifying the NUCLST statement in LOADxx, see Chapter 68, "LOADxx (system configuration data sets)," on page 619.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for NUCLSTxx

The following syntax rules apply to NUCLSTxx:

- Each record consists of 80 columns, although columns 73 through 80 are ignored.
- The fields are **column-dependent**, as shown in the "Statements/Parameters" section. Columns not shown to contain data must contain blanks.
- Comments must be preceded by an asterisk in column 1.
- Blank lines are ignored.

Syntax format of NUCLSTxx

```
INCLUDE
EXCLUDE
```

IBM-supplied default for NUCLSTxx

None.

Statements/parameters for NUCLSTxx

INCLUDE

The INCLUDE statement specifies the name of a member of SYS1.NUCLEUS that is to be loaded into the nucleus region at IPL-time.

INCLUDE statements for members that are already included in the nucleus region through other means, such as via NMLs, are ignored.

Column	Contents
1-7	INCLUDE
10-17	The name of a member of SYS1.NUCLEUS to be loaded into the nucleus region at IPL-time. **Default**: None

EXCLUDE

The EXCLUDE statement specifies the name of a member of SYS1.NUCLEUS that is to be excluded from the nucleus region at IPL-time.

Informational messages are issued for all members that are excluded. You might want to review these messages to ensure that you have not excluded any members that are needed. Although the messages are not displayed on any console, you can read them by checking the IPL WTO buffer.

You can use the EXCLUDE statement to override INCLUDEs provided through NMLs and MLTs. The EXCLUDE statement must name a member of SYS1.NUCLEUS; otherwise, the EXCLUDE statement is ignored.

You cannot use the EXCLUDE statement to exclude a CSECT that was previously linked into IEANUC01 by the linkage editor. You must use the linkage editor to replace or remove the CSECT by re-linking IEANUC01, using the REPLACE keyword and the SCTR attribute.

Column	Contents
1-7	EXCLUDE
10-17	The name of a member of SYS1.NUCLEUS to be excluded from the nucleus region at IPL-time. **Default**: None

Example of replacing modules

You can replace nucleus members by using a combination of INCLUDE and EXCLUDE statements. To replace one member with another, take the following steps:

1. In NUCLSTxx, code an INCLUDE statement for the SYS1.NUCLEUS member to be used, and an EXCLUDE statement for the member to be replaced, as shown:

   ```
   EXCLUDE oldmod
   INCLUDE newmod
   ```

   The *oldmod* is a member of SYS1.NUCLEUS that was to be included through either an MLT or an NML, and *newmod* is a modified copy of oldmod and also is a member of SYS1.NUCLEUS. Note that a different member name is used for the modified copy specified on the INCLUDE statement.
2. IPL the system.

Chapter 75. PFKTABxx (program function key table definition)

The PFKTABxx is an installation-created member of Parmlib. In this member, you define one or more program function key (PFK) tables. In another installation-created member of parmlib, CONSOLxx, you define the console configuration for the installation and specify which PFK member is to be used with that configuration. By using the CONSOLxx and PFKTABxx members together, the system can automatically initialize the console configuration with the related PFK table(s) during IPL. This processing reduces the number of operator responses, and eliminates the need to define manually the PF key settings.

Parameter in IEASYSxx (or entered by the operator)

None.

Syntax rules for PFKTABxx

The following rules apply to the creation of PFKTABxx:

- You may define multiple PFK tables in a member.
- Data must be contained in columns 1-71; the system ignores columns 72-80.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- The first statement type in a member must be PFKTAB TABLE(tablname).
- Code the statement type as the first data item on a record.
- For each PFK table, define each program function key only once.
- One or more blanks must follow the statement types; you must code at least one blank between the statement type and the keyword.
- Keyword values must be set off by parentheses. If you code multiple values on a keyword, separate the values with a blank or a comma.
- Do not use blanks, commas or comments in the middle of a keyword, between the keyword and the left parenthesis before the value, or in the middle of a value.
- A statement type continues to the next statement type in the member or until the end of the member. Therefore, there is no continuation character (except for CMD text, see below).
- CMD text that spans more than one (1) parmlib record line and specifies a system symbolic requires all lines to be enclosed with single quotation marks (') or double quotation marks (") and to be connected with a comma (,).

A Syntax Example:

```
PFKTAB TABLE(nnnnnnnn)

[         [{CMD({"cccccc[;cccccc]..."})}]          ]
[         [{   ({"cccccc"[,"cccccc"]..."})}]       ]
[         [{   ({'cccccc'[,'cccccc']...'})}]       ]
[PFK(xx)  [{    {'cccccc[;cccccc]...'} }][CON({Y})]]
[         [{                          }][    {N} ]]
[         [{KEY(kk[,kk]...)           }]          ]
```

Using the display command

Once the PFKTABxx member has been invoked through the SET PFK command, you may use the DISPLAY command to view the contents of that specific PFKTABxx member. For the correct syntax, see *z/OS MVS System Commands*.

IBM-supplied default for PFKTABxx

The IBM-supplied default member of SYS1.PARMLIB is PFKTAB00.

Statements/parameters for PFKTABxx

PFKTAB

TABLE (tablname)

PFKTAB indicates that a new PFK table is being defined. PFKTAB must be the first definition in a PFKTABxx member. The *tablname* indicates the name associated with this PFK table and must be 1-8 alphanumeric characters. The value you specify for tablname here is the value you specify for PFKTAB(tablname) in the CONSOLxx parmlib member.

PFK(xx)

Indicates the program function key that is being defined. (*xx*) is a decimal value from 1 through 24. Each xx value must be unique within a table.

```
CMD     {('command[;command]...')}
        {('command[;command]...')}
```

CMD indicates that the PFK is to have a command or commands associated with it. *command* specifies the command. Multiple commands on a CMD line must be separated with a semi-colon. The maximum length of the commands including the single quotation marks is 126 characters. If a command contains a single quotation mark, surround the command with double quotation marks. If the command contain double quotation marks, surround the command with single quotation marks. If a command contains an underscore, define the key as conversational CON(Y). When the system displays the command, the cursor will be under the character immediately to the right of the underscore. The system will not display the underscore. IBM suggests that you specify the complete command rather than take any parameter defaults, since the defaults might change. See *z/OS MVS System Commands* for details. JES3 commands must have an asterisk (*) for a prefix or an alternate prefix as specified on the CONSTD statement of the JES3 initialization deck.

If you want an underscore to appear in the command, use two underscores. They are treated as one underscore, and are not used for cursor placement.

```
KEY     {(xx)      }
        {(xx,xx...)}
```

KEY indicates that the PFK being defined is to be associated with another key or a list of PF keys. *xx* indicates the PFK to be processed when the PFK being defined is pressed. *xx* is a decimal number between 1-24 and cannot be the same value as on the PFK(xx). For example, do not define PFK(10) KEY(10). You may code a maximum of 62 keys in a key list.

```
CON     {(Y)}
        {(N)}
```

Specifies whether the PFK being defined is to be conversational or

nonconversational. If the PFK is conversational, CON(Y), the system will display the command so you can modify it before executing it. If the PFK being defined is associated with a list of keys, and the PFK is conversational, the system displays the command associated with each key in the key list so you can make modifications. If a PFK is nonconversational, the system automatically executes the command when the PFK is pressed. If you do not specify CON, the default is nonconversational; the command will not be displayed before executing.

Note: If a command contains a single underscore used to place the cursor, you must specify CON(Y).

Chapter 76. PROGxx (authorized program list, exits, LNKLST sets and LPA)

The PROGxx parmlib member contains the following optional statement types:

- APF, which defines the format and contents of the APF-authorized program library list. (See "Using the APF statement.")
- EXIT, which controls the use of exits and exit routines. (See "Using the EXIT statement" on page 698.)
- SYSLIB, which allows for the definition of alternate data sets for the system defaults (SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, SYS1.SIEAMIGE, and SYS1.LPALIB) at the beginning of the LNKLST and the LPALST concatenations. (See "Using the SYSLIB statement" on page 699.)
- LNKLST, which controls the definition and activation of a LNKLST set of data sets for the LNKLST concatenation. (See "Using the LNKLST statement" on page 701.)
- LPA, which defines the modules to be added to, or deleted from, LPA after IPL. (See "Using the LPA statement" on page 706.)
- REFRPROT, which indicates that REFR programs are protected. (See "Using the REFRPROT statement" on page 707.)
- NOREFRPROT, which indicates that REFR programs are not protected. (See "Using the NOREFRPROT statement" on page 707.)
- TRACKDIRLOAD, which enables system-wide tracking of directed load modules. (See "Using the TRACKDIRLOAD statement" on page 708.)
- NOTRACKDIRLOAD, which disables system-wide tracking of directed load modules. (See "Using the NOTRACKDIRLOAD statement" on page 708.)

Using the APF statement

Use the APF statement to specify the following:

- The format (dynamic or static) of the APF-authorized library list.
- Program libraries to be added to the APF list.
- Program libraries to be deleted from the APF list.

The system automatically adds SYS1.LINKLIB and SYS1.SVCLIB to the APF list at IPL. In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

If you specify a *dynamic* APF list format in PROGxx, you can update the APF list at any time during normal processing or at initial program load (IPL). You can also enter an unlimited number of libraries in the APF list.

If you specify a *static* APF list format in PROGxx, you can define the list only at IPL, and are limited to defining a maximum of 255 library names (SYS1.LINKLIB and SYS1.SVCLIB, which are automatically placed in the list at IPL, and up to 253 libraries specified by your installation).

Note:

1. If you currently specify APF-authorized libraries in the IEAAPFxx parmlib member, you can convert the format of IEAAPFxx to PROGxx using the IEAAPFPR REXX exec provided by IBM. For information about how to perform this conversion, see "Using the IEAAPFPR exec."
2. Except for concatenations opened during system initialization, an unauthorized library concatenated to any authorized libraries will cause the system to consider all the concatenated libraries unauthorized.
3. If you allow storage management subsystem (SMS) to manage a library, the system may move the library to a different volume during normal SMS processing. To ensure that the library retains authorization, specify SMS on its APF statement.
4. When LNKAUTH=APFTAB is specified, the system considers SYS1.MIGLIB and SYS1.CSSLIB to be APF-authorized when they are accessed as part of the concatenation (even when they are not included in the APF list).

Defining aliases in the APF list

Do not define aliases in the APF list because IBM's data management services (for example, OPEN processing) map an alias to its actual library name and query the APF list by the actual library name. An alias in the APF list does not authorize anything.

Using the IEAAPFPR exec

If you haven't yet converted IEAAPFxx parmlib members to PROGxx parmlib members, you can do it using the following procedure:

Place the IEAAPFPR exec in a data set that is accessible to ISPF and PDF edit macros. If you specify the EXECUTIL SEARCHDD(YES) command, or if you have modified the TSO/E installation parameters to search SYSEXEC automatically, place IEAAPFPR in a data set allocated to SYSEXEC. Otherwise, place IEAAPFPR in a data set allocated to SYSPROC.

To invoke IEAAPFPR:

1. Make sure that the IEAAPFxx member to be converted is valid and syntactically correct.
2. Edit a PROGxx parmlib member.
3. Copy the IEAAPFxx member into the PROGxx member.
4. Enter the following on the edit command line and press ENTER. The system places the modified member in the edit buffer.

   ```
   IEAAPFPR
   ```
5. Save the new PROGxx member.

Note: After you add PROG=xx to IEASYSxx, remove APF=xx from IEASYSxx and IEASYS00 to avoid duplication of processing.

Using the EXIT statement

Use the EXIT statement type to specify statements that:

- Add exit routines to an exit.
- Replace exit routines for an exit.
- Modify exit routines for an exit.

- Delete exit routines from an exit.
- Undefine implicitly defined exits.
- Change the attributes of an exit.

You can use the PROGxx EXIT statement to define exits to the dynamic exits facility at IPL. You can use multiple ADD statements to add more than one exit routine to a particular exit.

Previously defined exit definitions can be modified with the PROGxx EXIT statement, the SET PROG=xx operator command, or the SETPROG EXIT operator command through the following methods:

- The EXIT statement of the PROGxx parmlib member. The PROGxx EXIT statement interacts with the PROG=xx parameter of IEASYSxx and the SET PROG=xx command. At IPL, operators can use PROG=xx to specify the particular PROGxx parmlib member the system is to use. During normal processing, operators can use the SET PROG=xx command to set a current PROGxx parmlib member.
- The SETPROG EXIT operator command. This command performs the same functions as the EXIT statement of the PROGxx parmlib member. See *z/OS MVS System Commands* for information about the SETPROG EXIT command.
- The CSVDYNEX macro. The CSVDYNEX macro can be used to define exits to the dynamic exits facility, control their use within a program, and associate one or more exit routines with those exits. It can also be used to associate exit routines with the existing SMF and allocation exits, which have been defined to the dynamic exits facility.

An installation can use any of these methods to control dynamic exits. For example, an exit routine can be associated with an exit using the CSVDYNEX ADD request, the SETPROG EXIT,ADD operator command, or the EXIT statement of PROGxx.

Note:

1. If you currently specify exits in the EXITxx parmlib member, IBM suggests that you convert the format of EXITxx to PROGxx using the IEFEXPR REXX exec provided by IBM.
2. Exits that are currently specified in the SMFPRMxx parmlib member can also be specified in the PROGxx parmlib member. See “Specifying SMF Exits to the Dynamic Exits Facility” on page 767 for an example of how to use the EXIT ADD statement to specify SMF exits in PROGxx parmlib members.
3. Exit definitions are not replaced by the SET PROG=xx or SETPROG EXIT commands. They are modified as specified in the command.
4. The SET PROG=xx command may need authorization to define a dynamic exit via the REQUEST=DEFINE option of the CSVDYNEX macro. See *z/OS MVS Planning: Operations* for instructions.

Using the SYSLIB statement

Use the SYSLIB statement at IPL when you want to change the default system data sets that are placed at the beginning of the LNKLST concatenation or LPALST concatenation. The system recognizes the following:

- LINKLIB data set (which defaults to SYS1.LINKLIB)
- MIGLIB data set (which defaults to SYS1.MIGLIB)
- CSSLIB data set (which defaults to SYS1.CSSLIB)

- LINKLIBE data set (which defaults to SYS1.SIEALNKE)
- MIGLIBE data set (which defaults to SYS1.SIEAMIGE)
- LPALIB data set (which defaults to SYS1.LPALIB)

The system always places the LINKLIB, MIGLIB, CSSLIB, LINKLIBE, and MIGLIBE data sets at the beginning of the LNKLST concatenation and the LPALIB data set at the beginning of the LPALST concatenation. SYSLIB can be used only at IPL.

Use the SYSLIB LINKLIB statement to change the LINKLIB data set. Use the SYSLIB MIGLIB statement to change the MIGLIB data set. Use the SYSLIB CSSLIB statement to change the CSSLIB data set. Use the SYSLIB LINKLIBE statement to change the LINKLIBE data set. Use the SYSLIB MIGLIBE statement to change the MIGLIBE data set. Use the SYSLIB LPALIB statement to change the LPALIB data set.

When you use SYSLIB statements to change the defaults, you must ensure that SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE are defined to the LNKLST concatenation and that SYS1.LPALIB is defined to the LPALST concatenation for the system.

You can use SYSLIB statements in PROGxx when you want to override system code, either for testing or as part of customization. Use the LINKLIB, MIGLIB, CSSLIB, LINKLIBE, and MIGLIBE options of the SYSLIB statement to place the data sets you define at the beginning of the LNKLST concatenation. Use the LPALIB option of the SYSLIB statement to place the alternate data set you define at the beginning of the LPALST concatenation. Using these alternate system data sets, you can override system code supplied in LINKLIB, MIGLIB, CSSLIB, LINKLIBE, MIGLIBE, and LPALIB to test fixes or make exits or other vendors code available to the system without having to modify the system code itself.

If you use SYSLIB statements for testing, you can re-IPL after testing with the regular system libraries appearing first in the link list and LPA list concatenations. You can specify SYSLIB statements in PROGxx and use either PROGxx with LNKLST statements or LNKLSTxx members to define and activate the remainder of the LNKLST concatenation.

Some system exits and tables reside in their own load modules and, for some of them, default exits are supplied in LPALIB and LINKLIB by IBM. Some products, from IBM and other vendors, supply exits meant to replace IBM's default exits. Some vendors ship replacement modules meant to replace IBM-supplied modules. Your installation might have exits and tables that replace LINKLIB and LPALIB at the beginning of the link and LPA lists. This allows you to override these modules rather than replacing them, while keeping the IBM-supplied default modules available for recovery and for use on different system images, and keeping other vendors code separated from IBM's code. This makes it easier to share software among images with different requirements or software licenses and can reduce the time that is required to install new levels of software.

For examples of specifying SYSLIB statements, see “Example of the SYSLIB statement” on page 716 and “Examples of the LNKLST statement” on page 723.

Using the LNKLST statement

A LNKLST set consists of an ordered list of data sets for processing as the LNKLST concatenation. Every LNKLST set contains the LINKLIB, MIGLIB, CSSLIB, LINKLIBE, and MIGLIBE data sets as the first data sets in the LNKLST concatenation. Unless overridden by SYSLIB statements, every LNKLST set begins with:

- SYS1.LINKLIB
- SYS1.MIGLIB
- SYS1.CSSLIB
- SYS1.SIEALNKE
- SYS1.SIEAMIGE

The system automatically adds these data sets to the beginning of the LNKLST set that you define. If these data sets are not available to the system at IPL, a wait state occurs.

Use the LNKLST statement:

- To define the LNKLST set.
- To add a data set to the LNKLST set.
- To delete a data set from the LNKLST set.
- To remove the definition of a LNKLST set from PROGxx (valid only after IPL).
- To test for the location of a routine associated with one of the data sets in the LNKLST set (valid only after IPL).
- To associate a job or address space with the current LNKLST set (valid only after IPL).
- To indicate that a LNKLST set is to be activated.

Note: You must enter parameters on the LNKLST statement in the order that is shown in the syntax diagrams. See “Syntax format of the LNKLST statements” on page 717

You can add a data set to any LNKLST set that you define and specify the position of the data set in the list, however an alias cannot be added for the LNKLST created at IPL. You cannot add the data set before any of the system default data sets in the concatenation; that is, you can only concatenate the data set after the CSSLIB data set in the LNKLST set. To change the system default data sets placed at the beginning of the LNKLST concatenation, see “Using the SYSLIB statement” on page 699.

Using PROGxx instead of LNKLSTxx

You can use LNKLST statements in PROGxx instead of using LNKLSTxx to define the LNKLST concatenation. At IPL, ensure that you have a LNKLST ACTIVATE statement for the LNKLST set that you have defined, and specify PROG=xx instead of LNK=xx.

Converting a LNKLSTxx member to PROGxx using the CSVLNKPR exec

To convert a LNKLSTxx member to PROGxx format, use the CSVLNKPR REXX exec in SYS1.SAMPLIB. To install CSVLNKPR, place the exec in a data set that is accessible to ISPF and PDF edit macros. If you specify EXECUTIL SEARCHDD(YES) command, or if you have modified the TSO/E installation

parameters to search SYSEXEC automatically, place CSVLNKPR in a data set allocated to SYSEXEC. Otherwise, place CSVLNKPR in a data set allocated to SYSPROC.

To invoke CSVLNKPR:

1. Make sure that the LNKLSTxx member to be converted is valid and syntactically correct.
2. Create a new, empty PROGxx parmlib member. It should be empty until you copy the LNKLSTxx member into it, because CSVLNKPR comments out any statement not in LNKLSTxx format.
3. Copy the LNKLST statements from the old LNKLSTxx member into the empty PROGxx member.

   Because the system automatically places the LINKLIB, MIGLIB, or CSSLIB data sets into the LNKLST set, you do not need to specify them in the new PROGxx member. The data set names for the LINKLIB, MIGLIB, and CSSLIB default to SYS1.xxxxLIB, but you can change the data set names, if desired, using the SYSLIB statement in PROGxx. If you specify the LINKLIB, MIGLIB, or CSSLIB data set in the LNKLSTxx member, and you convert the member to PROGxx format and IPL with that PROGxx member, the system will issue message CSV532I to indicate the presence of a superfluous LNKLST statement in that PROGxx member.
4. Enter CSVLNKPR on the edit command line. The system places the modified member in the edit buffer. If you enter CSVLNKPR without a name, the name of the LNKLST set in PROGxx defaults to LNKLSTXX. If you specify CSVLNKPR(*name*), the system uses *name* as the name of the LNKLST set in PROGxx.
5. Save the new PROGxx member.
6. After you have saved the new PROGxx member, remove the LNK=xx system parameter from IEASYSxx and from IEASYS00. Then activate PROGxx by specifying the PROG=xx system parameter at the next IPL. You can place PROG=xx in IEASYSxx and IEASYS00.

Using LNKLST processing

You can define multiple LNKLST sets, but only one LNKLST set is current in the system at any time. Any job or address space that is started after the current LNKLST set is activated is associated with the current LNKLST set. The job or address space continues to use the current LNKLST set until the LNKLST set for the job or address space is updated. (See the parameter description for LNKLST UPDATE.)

Changing the current LNKLST set

You can change the current LNKLST set dynamically through the SET PROG=xx and SETPROG LNKLST commands.

A LNKLST set remains allocated until there are no longer any jobs or address spaces associated with it. If the current LNKLST set is dynamically changed, any job or address space associated with the previous LNKLST set continues to use the data sets until the job or address space finishes processing. Thus, a previously current LNKLST set might be active or in use by a job or address space even though a new current LNKLST set has been activated. Jobs or address spaces that are started after the new current LNKLST set is activated use the new current LNKLST set.

An active LNKLST set **cannot** be modified. Once the last job or address space associated with a LNKLST set terminates, the LNKLST set is no longer active. The only other way to deactivate a LNKLST set is with LNKLST UPDATE. See "Removing or compressing a data set in an active LNKLST set" on page 705 for more information about LNKLST UPDATE.

Through SET PROG=xx and SETPROG LNKLST, you can also remove the definition of a LNKLST set from the system, associate a job or address space with the current LNKLST set, or locate a specific module associated with a data set in the LNKLST set. See *z/OS MVS System Commands*.

Concatenating data sets to the LNKLST concatenation

The number of data sets you can concatenate to form the LNKLST concatenation is limited by the total number of DASD extents the data sets will occupy. The total number of extents must not exceed 255. A partitioned data set extended (PDSE) counts as one extent.

The system concatenates as many of the data sets as possible until the limit of 255 extents is reached. The system ignores the remaining data sets. When the limit has been exceeded, the system writes error message IEA328E to the operator's console. This message is issued whether the concatenation is defined by LNKLSTxx or by PROGxx.

Allocating a PDS or PDSE for use with LNKLST

When using PDSs and PDSEs, you can allocate LNKLST data sets with secondary extents. However, IBM suggests that PDSs in the LNKLST be allocated with only primary extents, for two reasons. First, this makes it easier to stay within the 255-extent limit for an active LNKLST concatenation without having to reallocate data sets in fewer extents initially. Second, if a PDS is updated while in the link list, it can be extended if it has been allocated using secondary space. This can cause members to be placed in extents that did not exist when the LNKLST concatenation was opened. Any attempt to access a member in a new extent causes the requesting program to abend with a logical I/O error. This recommendation does not apply to PDSEs.

If an LNKLST PDS has expanded into a secondary extent since the most recent IPL, a program can use either of the following methods to access a member in the secondary extent:

- Accessing the member as part of a joblib, steplib, or tasklib. This method causes the data set to lose its authorized program facility (APF) authorization for the duration of the job, step, or task, unless the data set is specified in the APF list (through the PROGxx or IEAAPFxx member). Note that SETPROG APF or SET PROG can be used to dynamically update the APF table.

  OR

- Dynamic LNKLST processing can be used to define a new current LNKLST. The following is one way this could be accomplished:

  1. Define a new LNKLST set with the same data sets (including the one that has gone into a new extend) as are in the current LNKLST set.

     ```
     SETPROG LNKLST,DEFINE,NAME=new,COPYFROM=CURRENT
     ```

  2. Activate the new LNKLST set, making it the current LNKLST.

     ```
     SETPROG LNKLST,ACTIVATE,NAME=new
     ```

Now all address spaces that start while this new LNKLST set is current can able to access modules in the new extent. Address spaces that existed prior to this new LNKLST set becoming current will still not be able to access modules in the new extent.

3. The SETPROG LNKLST,UPDATE option may be used to force pre-existing address spaces to use the new current LNKLST.

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

   There is some risk associated with this command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL. Please see the SETPROG command in *z/OS MVS System Commands* for more important information about this option.

APF authorization for LNKLST data sets

SYS1.LINKLIB in the LNKLST concatenation is APF-authorized by default. And, if you accept the default for the LNKAUTH system parameter (LNKAUTH=LNKLST) or specify this value through an IEASYSxx member or as a response to the 'SPECIFY SYSTEM PARAMETERS' message at IPL time, data sets that are concatenated to SYS1.LINKLIB by default (like SYS1.MIGLIB and SYS1.CSSLIB) are automatically APF-authorized when accessed as part of the LNKLST concatenation.

For more information, see the description of the LNKAUTH parameter in Chapter 54, "IEASYSxx (system parameter list)," on page 431.

Cataloging LNKLST data sets

Data sets in the LNKLST concatenation must be cataloged in either the system master catalog or in a user catalog.

Cataloged data sets must be defined using the standard search order for requests as outlined in *z/OS DFSMS Managing Catalogs*. Variations in the standard search order can result in data sets not being found during LNKLST processing.

LNKLST data sets processed during IPL must be either cataloged in the master catalog or have their volume serials specified.

Note: The volume specification is not intended to get LNKLST to manage a data set that is not cataloged. It is designed to let the user put a user-cataloged data set in the LNKLST. Since user catalog support is not available when IPL is processing LNKLST, the volume must be specified to let LNKLST processing know where the data set resides. If a volume is specified, LNKLST processing will not try to locate the data set in the Master Catalog.

If you catalog a LNKLST data set in the system master catalog, the system will find the data set automatically during IPL.

If you plan to use a user catalog, you should be aware that the system will not find the data set unless you specify both the name of the data set and the volume serial number (VOLSER) of the DASD volume on which the data set resides. (This restriction also applies if you are defining the LNKLST concatenation in LNKLSTxx.)

Also, be aware that the system does not search OS CVOLs for LNKLST data sets.

Modifying the contents of LNKLST data sets

If you update members in LNKLST data sets:

- Be sure to refresh LLA's directory table after completing the updates if you want to have the changes take effect immediately. You can refresh LLA in the following ways:
  - To update specific entries in LLA's directory table, enter the MODIFY LLA UPDATE command.
  - To refresh all entries in LLA's directory table, enter the MODIFY LLA REFRESH command.
  - Recycle (stop and restart) LLA.

  For more information about these commands, see *z/OS MVS System Commands*.
- Do not remove the LNKLST data sets from LLA during the update, to avoid performance problems.

Removing an XCFAS ENQ

The XCFAS address space has an ENQ on each LNKLST data set. These ENQs provide serialization on the LNKLST data sets. An ENQ on a LNKLST data set prevents that data set from being altered as long as it is a member of an active LNKLST. There is no connection between sysplex processing and LNKLST processing.

There are times when you may want to remove the XCFAS ENQ; for example, when you want to update a data set of the same name on a different volume. Use the LNKLST UNALLOCATE statement, described in "Syntax format of the LNKLST statements" on page 717, or the SETPROG LNKLST,UNALLOCATE system command, described in *z/OS MVS System Commands*, to remove the ENQ.

LLA also holds an ENQ on each LNKLST data set. You can remove this ENQ by updating LLA to REMOVE the data set from LLA management, or by stopping LLA. See Chapter 25, "CSVLLAxx (library lookaside (LLA) list)," on page 277. For more information about an LLA-managed data set, see "Removing Libraries from LLA Management" in the *z/OS MVS Initialization and Tuning Guide*.

Removing or compressing a data set in an active LNKLST set

It is sometimes necessary to remove a data set from the active LNKLST set. Perhaps the data set is no longer needed, is causing a problem, or needs to be compressed. Use LNKLST DELETE to remove a data set. The data set cannot be removed while a LNKLST set is active (in use by at least one active address space). Use the following procedure to assure the data set being removed is not part of an active LNKLST set:

1. Define a new LNKLST set identical to the one from which the data set is to be removed.
2. Remove the appropriate data set from the new LNKLST set.
3. Activate the new LNKLST set. This makes the new LNKLST set CURRENT on the system.
4. Update all currently running jobs to use the current LNKLST set.

The following example shows how to remove a data set from an active LNKLST set:

```
SETPROG LNKLST DEFINE NAME(NEWLLSET) COPYFROM(OLDLLSET)
SETPROG LNKLST DELETE NAME(NEWLLSET) DSNAME(data set.to.be.removed)
SETPROG LNKLST ACTIVATE NAME(NEWLLSET)
SETPROG LNKLST UPDATE JOB(*)
```

Note: Using UPDATE to switch LNKLST sets for an active job could result in fetch failures. IBM suggests that you do this only when necessary.

When a new LNKLST set is made current, data sets that are no longer part of an active LNKLST continue to be managed by LLA. If you want LLA to no longer manage and to drop its allocation of a data set you have dynamically removed for all LNKLST sets, you must issue a MODIFY LLA,UPDATE=xx pointing to a CSVLLAxx member specifying the REMOVE keyword for the library removed from the LNKLST. Although stopping and starting LLA after the dynamic LNKLST removal of the data set causes LLA to drop its management and allocation of the data set, this also slows system performance. To avoid slowing system performance, use the MODIFY LLA command to change the library indexes.

When compressing non-LNKLST libraries, see the REMOVE statement in "Statements/parameters for CSVLLAxx" on page 278.

Placement of SYSLIB and LNKLST statements in PROGxx

You can place LNKLST statements for a LNKLST set in different PROGxx members. For example, you can specify PROG=(01,02,03) and place the LNKLST DEFINE statement in PROG01, LNKLST ADD statements in PROG02, and the LNKLST ACTIVATE statement in PROG03.

SYSLIB statements must always appear before any LNKLST statements in PROGxx. If you specify multiple PROG=xx members, define any SYSLIB statements ahead of LNKLST statements. For example, if you specify PROG=(01,02) during IPL, consider the following:

- If PROG01 has a LNKLST statement, ensure that no SYSLIB statement appears after a LNKLST statement, or in PROG02.
- If only PROG02 has a LNKLST statement, ensure that no SYSLIB statement appears after a LNKLST statement in PROG02.

Using the LPA statement

Use the LPA statement to specify:

- Modules that are to be added to the LPA at the end of the IPL. This is needed only for program objects within PDSEs but can be used for load modules within PDSs.
- Modules that are to be added to the LPA following IPL.
- Modules that are to be deleted from the LPA following IPL.
- Threshold values for minimum amounts of CSA storage that must be available after an ADD operation.

You can also initiate a change to LPA from a program via the CSVDYLPA macro, or by an operator using the SETPROG command. The PROG system parameter can be used to specify CSA threshold values, but not to request ADD or DELETE operations.

You can exercise certain controls over the modules to be loaded:

- You specify a data set from which the system is to load the modules. You must be authorized to make the request.
- You can request that the modules be placed into fixed common storage.
- You can request that only the full pages within a load module be page-protected. This does leave the likelihood of the beginning and/or end of a load module not being page protected. By default, each module is individually page-protected. This is, however, wasteful of common storage, as each module needs then to occupy a whole number of 4096-byte pages. In all cases the module will be in key 0 storage.
- You can indicate that the module being replaced is the routine for a specific SVC. In this case, the system updates the SVC table for the specified SVC with the new entry point address.

By default, LPA module alias names are not automatically handled. By default, if a module has aliases, the module name and all associated aliases must be specified within the same request. Otherwise, one of the following outcomes could occur, depending upon the initial state of the system:

- The module name or alias will not be found
- A duplicate copy of the same module will be loaded
- A previous copy of the module will be used.

You can use the ADDALIAS option to indicate that the system is to process aliases of the specified modules.

LPA modules are considered as coming from an authorized library. As part of its LPA search, the system finds modules that were added dynamically. A module added dynamically is found before one of the same name added during IPL.

The LPA statement is intended to replace modules only in cases where the owning product verifies the replacement. Otherwise, replacement could result in partial updates, or if the module address has already been saved by its owning product, an LPA search will not be done and the updated module will not be found.

It is sometimes necessary to re-IPL to replace LPA modules. For example, many service updates of LPA modules will require a re-IPL.

Using the REFRPROT statement

Use the REFRPROT statement type to specify that REFR programs are protected from modification by placing them in key 0, non-fetch protected storage, and page protecting the full pages. Therefore, any parts of the program that are on partial pages are not page-protected. For more information on protection of REFR programs, see *z/OS MVS Program Management: User's Guide and Reference*.

Note that the specification of REFRPROT will cause modules link-edited with the REFR attribute to be placed into key0, non-fetch protected, page-protected storage, regardless of their APF authorization.

Using the NOREFRPROT statement

Use the NOREFRPROT statement type to specify that REFR programs are not protected from modification.

Using the TRACKDIRLOAD statement

Use the SETPROG TRACKDIRLOAD statement to enable system-wide tracking of directed load modules. A directed load module is a module that is directly loaded to a specified storage address. When enabled, mapping information about directed load modules is included in the maps produced by hardware instrumentation services (HIS). Directed load tracking is enabled by default.

Using the NOTRACKDIRLOAD statement

Use the SETPROG NOTRACKDIRLOAD statement to disable system-wide tracking of directed load modules.

Using the DEFAULTS statement

Use the DEFAULTS statement type to set defaults that the system is to use when processing LPA and LNKLST statements in PROGxx and the SETPROG LPA, SETPROG LNKLST, and DISPLAY PROG,EXIT commands.

Parameter in IEASYSxx (or specified by the operator)

PROG=xx

The two-character identifier *xx* is appended to PROG to identify the PROGxx parmlib member. *xx* can be any two alphanumeric characters (A-Z, 0-9) or (@, #, or $).

You can also specify multiple PROGxx members on this parameter. For example, you can specify two active members using the form PROG=(01,02).

Regardless of whether you specify the PROG=xx parameter, the system always places the following libraries in the APF list:

- SYS1.LINKLIB and SYS1.SVCLIB
- If the APF=xx system parameter is specified, the libraries contained in IEAAPFxx.

Note: In addition, any module in the link pack area (pageable LPA, modified LPA, fixed LPA, or dynamic LPA) will be treated by the system as though it came from an APF-authorized library. Ensure that you have properly protected SYS1.LPALIB and any other library that contributes modules to the link pack area to avoid system security and integrity exposures, just as you would protect any APF-authorized library.

PROG=xx and APF=xx

If you specify both the PROG=xx and the APF=xx parameters, then the system places into the APF list the libraries listed in IEAAPFxx, followed by the libraries listed in the PROGxx member or members.

The system will process IEAAPFxx and PROGxx parameters if both are specified. If you decide to use PROGxx **only**, convert the format of IEAAPFxx to PROGxx and then remove APF=xx system parameters from IEASYSxx and IEASYS00.

PROG=xx and EXIT=xx

The system will process first PROGxx and then EXITxx parameters if both are specified. If you decide to use PROGxx only, convert the format of EXITxx to PROGxx and then remove EXIT=xx system parameters from IEASYSxx and IEASYS00.

PROG=xx and LNK=xx

You can specify PROG=xx instead of LNK=xx for the LNKLST concatenation. Whether you use PROGxx or LNKLSTxx to define the LNKLST concatenation, the system always places the LINKLIB, MIGLIB, and CSSLIB data sets (either the system defaults or the data sets specified on SYSLIB statements) first in the concatenation.

If you use PROGxx and do not use LNKLST statements, the system uses LNKLSTxx, if specified, on LNK=xx to define the LNKLST concatenation.

If you define a LNKLST set to be activated through PROGxx and specify both PROG=xx and LNK=xx, the system uses the definitions in PROGxx and issues message CSV487I:

LNK IPL PARAMETER HAS BEEN IGNORED. LNKLST SET *lnklstname* IS BEING USED

IBM-supplied default

None.

Syntax rules for PROGxx

These rules apply to the creation of PROGxx:

- Enter data only in columns 1 through 71. Do not enter data in columns 72 through 80; the system ignores these columns.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

Syntax format of the APF statement

```
APF FORMAT(DYNAMIC|STATIC)
APF ADD | DELETE
    DSNAME(dsname)
    SMS | VOLUME(volname)
```

APF
Statement type indicating that an action is to be performed on the APF list.

FORMAT(DYNAMIC|STATIC)
Specifies that the format of the APF list is dynamic or static. Before you change the format of the APF list to dynamic, validate programs and vendor products are converted to use dynamic APF services (see Chapter 41, "IEAAPFxx (authorized program facility list)," on page 371), and that the proper program products are installed (see *z/OS Planning for Installation*).

Example: FORMAT(DYNAMIC)

Default: None

ADD|DELETE
Indicates whether you want to add or delete a library from the APF list.

Default: None

DSNAME(*dsname*)
The 44-character name of a library that you want to add or delete from the APF list. DSN, LIB, and LIBRARY are accepted synonyms for this parameter.

Example: DSNAME(SYSTEM.ACCT.DATA)

Default: None

SMS|VOLUME(*volname*)
The identifier for the volume containing the library specified on the DSNAME parameter, which is one of the following:

- SMS, which indicates that the library is SMS-managed.
- A six character identifier for the volume
- ******, which indicates that the library is located on the current SYSRES volume
- *MCAT*, which indicates that the library is located on the volume containing the master catalog.

VOL is an accepted abbreviation for the VOLUME parameter.

Example: VOLUME(874932) or VOLUME(&SYSRx).

Default: None

Example of the APF statement

The following example shows a PROG*xx* parmlib member that sets the format of the APF list to dynamic, and adds the following libraries to the APF list:

- SYS1.SUPER.UTILS on volume 614703
- SYS1.ACCTG.RECRDS on the current SYSRES volume
- SYS1.COMPU.DATA, an SMS-managed library.

```
APF FORMAT(DYNAMIC)
APF ADD
    DSNAME(SYS1.SUPER.UTILS)
    VOLUME(614703)
APF ADD /* Accounting records */
    DSNAME(SYS1.ACCTG.RECRDS)
    VOLUME(******)
APF ADD
    DSNAME(SYS1.COMPU.DATA)
    SMS
```

Syntax format of the EXIT statements

Syntax format of EXIT ADD:

```
EXIT ADD
     EXITNAME(ex)
     MODNAME(mmmm)
     [STATE({ACTIVE|INACTIVE})]
     [DSNAME(dd)]
     [JOBNAME(jjj|*)]
     [ABENDNUM(n[,CONSEC])]
     [FIRST|LAST]
     [PARAM(param)]
```

Syntax format of EXIT REPLACE:

```
EXIT REPLACE
     EXITNAME(ex)
     MODNAME(mmmm)
     [STATE({ACTIVE|INACTIVE})]
     [DSNAME(dsn)]
```

Syntax format of EXIT MODIFY:

```
EXIT MODIFY
     EXITNAME(ex)
     MODNAME(mmmm)
     [STATE({ACTIVE|INACTIVE})]
     [JOBNAME(jjj|*)]
```

Syntax format of EXIT DELETE:

```
EXIT DELETE
     EXITNAME(ex)
     MODNAME(mmmm)
     [FORCE({YES|NO})]
```

Syntax format of EXIT UNDEFINE:

```
EXIT UNDEFINE
     EXITNAME(ex)
```

Syntax format of EXIT ATTRIB:

```
EXIT ATTRIB
     EXITNAME(ex)
     KEEPRC(compare,kk)
```

EXIT
Statement type indicating that an action is to be performed on an exit or an exit routine.

ADD
Specifies that an exit routine is to be added to an exit.

Default value: None

REPLACE
Replaces an exit routine for an exit.

MODIFY
Specifies that the state of an exit routine is to be changed to active or inactive, and sets the conditions under which the exit routine is to be given control.

Default value: None

DELETE
Specifies that an exit routine is to be deleted from an exit.

Default value: None

UNDEFINE
Specifies that an implicitly defined exit is to be undefined. You define an exit implicitly if you add exit routines to it before it has been defined, or if you set attributes for it using the ATTRIB parameter before it has been defined. You can determine which exits have been implicitly defined by using the DISPLAY PROG,EXIT,ALL,IMPLICIT command.

Default value: None

ATTRIB
Specifies that the attributes of an exit are to be changed.

Default value: None

EXITNAME(ex)
The 1-16 character name of the exit. Names must be unique within the system. You can use alphanumerics, underscores, periods, and #, $, or @. (but never with the character string "SYS").

Default value: None

MODNAME(*mmmm*)
The 1-8 character name of the exit routine. You can use alphanumerics and #, $, and @.

If the DSNAME parameter is not specified, the system tries to locate the exit routine using the LPA, the LNKLST concatenation, and the nucleus. The MODNAME parameter is required for the ADD, DELETE, and MODIFY requests.

Default value: None

[STATE({ACTIVE|INACTIVE})]
Indicates the state of the exit routine. ACTIVE indicates that the exit routine is to be given control when the exit is called. INACTIVE indicates that the exit routine is not to be given control when the exit is called.

Default value: The default for the ADD parameter is ACTIVE. The default for the MODIFY parameter is to leave the current state of the exit routine unchanged.

[DSNAME(dd)]
The 1-44 character data set name of a load library in which the exit routine resides. The data set must be cataloged.

If the data set has been migrated, the issuer of the command that references PROGxx waits until the data set has been retrieved before continuing.

If the PROGxx member is specified in the IEASYSxx member, an exit routine fetched from the data set specified in a DSNAME parameter cannot be given control until the master scheduler is initialized. In contrast, an exit routine not fetched from the DSNAME-specified data set can be given control once the LNKLST is opened.

Default value: If the DSNAME parameter is not specified, the system tries to locate the exit routine using the LPA, the LNKLST concatenation, and the nucleus.

[JOBNAME(jjj|*)]

The 1-8 character name of the job or jobs for which this exit routine is to be given control. Note that you can only specify the JOBNAME parameter for an active exit routine. If the exit is called from another job, this exit routine is not given control. You can use the JOBNAME parameter to restrict the processing of an exit routine to a particular job.To indicate the name of more than one job, use an asterisk for the last character. A matching jobname is one that matches all the characters preceding the asterisk.

If you specify JOBNAME=*, you are requesting that the system not check for the jobname.

Default value: The default for the ADD parameter is JOBNAME=*, which indicates that the exit routine is to be given control when any job calls the exit. The default for the MODIFY parameter is to use the jobname or jobnames as specified on the ADD request.

[ABENDNUM(n[,CONSEC])]

Indicates when the system should stop giving control to the exit routine in case of abends. ABENDNUM(n) indicates that the exit routine is not to be given control after the *n*th abend. ABENDNUM=(n,CONSEC) indicates that there must be *n* consecutive abends before the system stops giving control to the exit routine. CONSEC is not supported if this exit has FASTPATH processing in effect and either a PSW key 8 to 15 or ANYKEY processing in effect.

Default value: The default is to use the ABENDNUM characteristics that were specified (or defaulted) when the exit was defined, or, if the exit is implicitly defined, the ABENDNUM characteristics that are specified when the exit is subsequently defined. The ABENDNUM value must not exceed 8 decimal digits.

[FIRST]

Specifies that the system is to call the exit routine before all other exit routines associated with this exit, unless another routine, added after it, also specifies the FIRST parameter.

Default value: If neither the FIRST parameter nor the LAST parameter is specified, the system may call the exit routine in any order relative to other exit routines associated with this exit.

[LAST]

Specifies that the system is to call the exit routine after all other exit routines associated with this exit, unless other routines are added after it.

Default value: If neither the FIRST parameter nor the LAST parameter is specified, the system may call the exit routine in any order relative to other exit routines associated with this exit.

PARAM=param
The 1 - 8 character parameter that is passed to the exit routine. If this parameter is less than 8 bytes, it is padded with blanks until it is 8 bytes. The first 4 bytes are passed to the exit routine in access register 0. The second 4 bytes are passed to the exit routine in access register 1. If you specify characters other than the following characters within the PARAM string, you must enclose the parameter string within single quotation marks:

- Uppercase alphabetic characters A - Z
- Numerics 0 - 9
- Special characters @, $, #
- Period, asterisk, question mark, underscore, hyphen

[FORCE({YES|NO})]
Indicates that the system is to delete the exit routine. The exit routine will no longer be given control. Specify FORCE(YES) for an exit with FASTPATH processing in effect, and either a PSW key 8 to 15, or ANYKEY processing in effect. Assuming the exit has FASTPATH processing in effect, and the PSW key is 8 to 15, or ANYKEY processing is in effect:

- FORCE(NO) changes the state of the exit routine to inactive. The system does not free the storage.
- FORCE(YES) frees the storage of the exit routine immediately. Use FORCE(YES) only if you are sure that no exit is running that exit routine.

For exits that are non-FASTPATH or whose PSW key is 0 to 7, and not ANYKEY, the system frees the storage when it determines that no other exits are using the exit routine.

Default value: FORCE(NO)

KEEPRC(compare,kk)
Specifies a comparison and a return code which, if true, cause the return information produced by this exit routine to be returned to the exit caller. The valid choices for *compare* are EQ, NE, GT, LT, GE, and LE. For example, with *KEEPRC(GT,4)*, if the exit routine produces a return code of 8, the compare for greater than with 4 is true, and KEEPRC processing causes the information produced by this exit routine to be returned to the exit caller.

If return codes from more than one exit routine match the conditions specified, the system returns information from the exit routine that finishes first.

Default value: The default is not to perform KEEPRC processing. Do not enter more than 8 decimal digits when specifying a value for *kk*. Refer to *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information about using the KEEPRC parameter.

Examples of EXIT statements

The following example shows a PROGxx parmlib member that does the following:

- Adds exit routine R1 to exit SYS.IEFUJI.
- Modifies exit routine R2 for exit SYS.IEFUSI to be inactive.
- Deletes exit routine R3 from exit SYS.IEFACTRT.
- Defines exit SYS.IEFUTL so that the system will “keep” information for return code 4 should any exit routine return with that return code.

```
EXIT ADD EXITNAME(SYS.IEFUJI) MODNAME(R1)
EXIT MODIFY EXITNAME(SYSSTC.IEFUSI) MODNAME(R2) STATE(INACTIVE)
EXIT DELETE EXITNAME(SYSJES3.IEFACTRT) MODNAME(R3)
EXIT ATTRIB EXITNAME(SYSTSO.IEFUTL) KEEPRC(EQ,4)
```

Syntax format of the SYSLIB statement

```
SYSLIB LINKLIB(name)    [ {VOLUME|VOLSER|VOL} (volser) ]

SYSLIB MIGLIB(name)     [ {VOLUME|VOLSER|VOL} (volser) ]

SYSLIB CSSLIB(name)     [ {VOLUME|VOLSER|VOL} (volser) ]

SYSLIB LINKLIBE(name)   [ {VOLUME|VOLSER|VOL} (volser) ]

SYSLIB MIGLIBE(name)    [ {VOLUME|VOLSER|VOL} (volser) ]

SYSLIB LPALIB(name)     [ {VOLUME|VOLSER|VOL} (volser) ]
```

SYSLIB
Statement type indicating that an alternate data set is to be defined for SYS1.LINKLIB, SYS1.MIGLIB, SYS1.SIEALNKE, SYS1.SIEAMIGE and SYS1.CSSLIB in the LNKLST concatenation, and for SYS1.LPALIB in the LPALST concatenation.

IBM suggests that you use different data set names for LINKLIB, MIGLIB, CSSLIB, LINKLIBE on all SYSLIB statements. If the same data set name is used more than once, LLA will not manage the entire LNKLST set.

LINKLIB(*name*)
Specifies the name of the LINKLIB data set. If you specify a library other than SYS1.LINKLIB, you must ensure that SYS1.LINKLIB occurs within the LNKLST concatenation. The system places the LINKLIB data set first in the LNKLST concatenation.

Default: If you do not specify LINKLIB, the system uses SYS1.LINKLIB as the LINKLIB data set.

MIGLIB(*name*)
Specifies the name of the MIGLIB data set. If you specify a library other than SYS1.MIGLIB, you must ensure that SYS1.MIGLIB occurs within the LNKLST concatenation. The system places the MIGLIB data set after the LINKLIB data set in the LNKLST concatenation.

Default: If you do not specify MIGLIB, the system uses SYS1.MIGLIB as the MIGLIB data set.

CSSLIB(*name*)
Specifies the name of the CSSLIB data set. If you specify a library other than SYS1.CSSLIB, you must ensure that SYS1.CSSLIB occurs within the LNKLST concatenation. The system places the CSSLIB data set after the MIGLIB data set in the LNKLST concatenation.

Default: If you do not specify CSSLIB, the system uses SYS1.CSSLIB as the CSSLIB data set.

LINKLIBE(*name*)
Specifies the name of the LINKLIBE data set. If you specify a library other than SYS1.SIEALNKE, you must ensure that SYS1.SIEALNKE occurs within the LNKLST concatenation. The system places the LINKLIBE data set after the CSSLIB data set in the LNKLST concatenation.

Default: If you do not specify LINKLIBE, the system uses SYS1.SIEALNKE as the LINKLIBE data set.

MIGLIBE(*name*)
Specifies the name of the MIGLIBE data set. If you specify a library other than SYS1.SIEAMIGE, you must ensure that SYS1.SIEAMIGE occurs within the LNKLST concatenation. The system places the MIGLIBE data set after the LINKLIBE data set in the LNKLST concatenation.

Default: If you do not specify MIGLIBE, the system uses SYS1.SIEAMIGE as the MIGLIBE data set.

LPALIB(*name*)
Specifies the name of the LPALIB data set. If you specify a library other than SYS1.LPALIB, you must ensure that SYS1.LPALIB occurs within the LPALST concatenation. The system places the LPALIB data set first in the LPALST concatenation.

Default: If you do not specify LPALIB, the system uses SYS1.LPALIB as the LPALIB data set.

[{VOLUME|VOLSER|VOL} (volser)]
Specifies the name of the volume on which the data set resides. The name can be from 1 to 6 characters.

This parameter is optional when the data set is cataloged in the system master catalog; the parameter is required when the data set is cataloged in a user catalog or if the data set is the LPALIB data set and it is not cataloged. You can use a value of ****** to indicate that the data set is located on the current SYSRES volume, or use a value of "*MCAT*" to indicate that the data set is located on the volume containing the master catalog.

For information other than that about the LPALIB data set, see “Cataloging LNKLST data sets” on page 704.

Note: If you do not specify [{VOLUME|VOLSER|VOL} (volser)], the system uses the volume indicated in the system master catalog.

Example of the SYSLIB statement

The following example shows a PROG*xx* parmlib member to be IPLed in a test environment that is applying code fixes for the system. The libraries that are specified on these SYSLIB statements contain no data set members. Whether you use PROGxx or LNKLSTxx to activate the LNKLST concatenation, the system places SYS2.LINKLIB, SYS2.MIGLIB, SYS2.CSSLIB, SYS2.SIEALNKE, and SYS2.SIEAMIGE at the start of the LNKLST concatenation. The system places SYS2.LPALIB at the beginning of the LPALST concatenation. (You must ensure that SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE and SYS1.SIEAMIGE appear somewhere in the LNKLST concatenation and SYS1.LPALIB appears somewhere in the LPALST concatenation.)

```
SYSLIB LINKLIB(SYS2.LINKLIB)
SYSLIB MIGLIB(SYS2.MIGLIB)
SYSLIB CSSLIB(SYS2.CSSLIB)
SYSLIB LINKLIBE(SYS2.SIEALNKE)
SYSLIB MIGLIBE(SYS2.SIEAMIGE)
SYSLIB LPALIB(SYS2.LPALIB)
```

Syntax format of the LNKLST statements

Rule: You must enter parameters on the LNKLST statement in the order shown in the syntax diagrams.

Syntax Format of LNKLST DEFINE:

```
LNKLST DEFINE NAME(name)
       [COPYFROM(name)]
       [NOCHECK]
```

Syntax Format of LNKLST ADD:

```
LNKLST ADD NAME(name) DSNAME(dsname)
       [{VOLUME|VOLSER|VOL}(name)]
       [{ATBOTTOM|ATTOP|AFTER(dsname)}]
       [CONCAT(NOCHECK|CHECK)]
```

Syntax Format of LNKLST DELETE:

```
LNKLST DELETE NAME(name) DSNAME(dsname)
```

Syntax Format of LNKLST UNDEFINE:

```
LNKLST UNDEFINE NAME(name)
```

Syntax Format of LNKLST TEST:

```
LNKLST TEST NAME(name) MODNAME(modname)
```

Syntax Format of LNKLST ACTIVATE:

```
LNKLST ACTIVATE NAME(name)
```

Syntax Format of LNKLST UPDATE:

```
LNKLST UPDATE {JOB(jobname)}
              {ASID(asid)}
              [DELAY(delay)]
```

Syntax Format of UNALLOCATE:

```
LNKLST UNALLOCATE
```

Syntax Format of ALLOCATE:

```
LNKLST ALLOCATE
```

LNKLST
Statement type indicating that an action is to be performed for a LNKLST set. LINKLIST, LINKLST, LNK, or LNKLIST can be specified as an alternative to LNKLST.

DEFINE
Specifies the definition of a LNKLST set (a set of ordered data sets for the LNKLST concatenation).

Default: None

ADD
Indicates that you want to add a data set to the specified LNKLST set.

IBM suggests using the COPYFROM parameter,when feasible, to define a LNKLST set instead of defining the set and then using LNKLST ADD statements to add individual data sets. The use of COPYFROM reduces system processing time. Use the LNKLST ADD statement to add a new data set to a specific LNKLST set.

You cannot add a data set to the current or active LNKLST set. If a data set has been migrated, the request waits until the data set is available. For information about the maximum number of data sets you can define to a LNKLST set, see "Concatenating data sets to the LNKLST concatenation" on page 703.

Default: None.

DELETE
Indicates that you want to delete a data set from the specified LNKLST set. You cannot delete a data set from the current or active LNKLST set.

Default: None.

UNDEFINE
Indicates that you want to remove the definition of the LNKLST set from the system. You cannot remove the definition of the current LNKLST set, another LNKLST set that is being actively used by a job or address space, or the LNKLST defined at IPL through LNKLSTxx and the LNK parameter of IEASYSxx.

Note: You cannot specify this parameter at IPL. It is valid only when you specify SET PROG=xx.

See "Removing or compressing a data set in an active LNKLST set" on page 705 for information about LLA management of LNKLST sets.

Default: None.

TEST
Indicates that you want to locate a specific routine associated with a data set in the LNKLST set. If the system locates the data set, the system indicates the name of the data set. If a data set has been migrated, the request waits until the data set is available.

Note: You cannot specify this parameter at IPL. It is valid only when you specify SET PROG=xx.

ACTIVATE
Indicates that you want to activate the specified LNKLST set as the LNKLST concatenation. If you use SETPROG LNKLST or SET PROG=xx to activate another LNKLST set after IPL, jobs or address spaces that are still active continue to use the previous current LNKLST set. To associate a job or an address space with the current LNKLST set after IPL, you can use the UPDATE option on the SETPROG LNKLST command or the LNKLST UPDATE statement specified in the member for SET PROG=xx.

If a data set in the LNKLST set has been migrated before the LNKLST set is activated, the request waits until the data set is available.

When the ACTIVATE request completes, the system issues an event (ENF) signal (event code 52). Depending on the options specified in SMFPRMxx, whenever a LNKLST set is activated, the system records SMF record type 90 subtype 29. See Chapter 78, "SMFPRMxx (system management facilities (SMF) parameters)," on page 741.

You must have a **LNKLST ACTIVATE** statement defined in PROGxx when you use PROG=xx to activate the LNKLST set at IPL.

See "Removing or compressing a data set in an active LNKLST set" on page 705 for information about LLA management of LNKLST sets.

UPDATE
Indicates that the system is to update an address space or the specified job or jobs to use the current LNKLST set. Otherwise, if the job is using another LNKLST set when the current LNKLST set is activated, it will continue to use the previous LNKLST set until it completes operations. When the job completes and restarts, the job then uses the data sets defined in the new currently active LNKLST set.

See "Removing or compressing a data set in an active LNKLST set" on page 705 for information about LLA management of LNKLST sets.

Note:

1. You cannot specify this parameter at IPL. It is valid only when you specify SET PROG=xx.
2. Be careful when you use UPDATE. Updating jobs in an address space while it is fetching a module can cause the fetch to fail or result in fetching a copy that is not up-to-date. The system does not attempt to verify the validity of the data for the update function.

UNALLOCATE
Indicates that you want to undo all existing allocations obtained while processing active LNKLST sets. This also releases the SYSDSN ENQ.

Note:

1. Make sure that you do not delete or move the LNKLST data sets while the allocations are not in effect (from the time that you use the UNALLOCATE request until the time that you use the ALLOCATE request).
2. Once you have completed everything associated with the UNALLOCATE, you must specify LNKLST ALLOCATE to re-obtain the remaining ENQs.

ALLOCATE
Indicates that you want to re-obtain the allocation (and SYSDSN ENQ) for every data set in every active LNKLST.

NAME(*name*)
The name of the LNKLST set that you want to specify. Naming conventions are as follows:

- You can specify from 1 to 16 characters for name.
- You can use alphanumerics, underscores, periods, and #, $, or @.
- Do not use imbedded blanks.
- Do not use the names CURRENT or IPL. The system uses CURRENT to mean the current LNKLST set and IPL to mean LNKLST information specified in SYS1.PARMLIB member LNKLSTxx.
- Do not begin the name with SYS. SYS is reserved for IBM use.

Example: NAME(MY.LNKLST.SET)

[COPYFROM(*name*)]
The name of an existing LNKLST set from which to initialize the LNKLST set you are defining. If you specify CURRENT for the name, the system uses the current LNKLST set.

Example:

```
LNKLST DEFINE NAME(MY.LNKLST.SET)
COPYFROM(SYSTEM.ACCT.DATA)
```

Example: If you omit this parameter, the system initializes the LNKLST set with the LINKLIB, MIGLIB, CSSLIB, LINKLIBE, and MIGLIBE data sets in that order.

[NOCHECK]
Indicates that the system does not check to determine if the specified LNKLST set contains SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, SYS1.SIEAMIGE before allocating the LNKLST concatenation.

Note: Use NOCHECK with caution. NOCHECK is available ONLY to allow the creation of a LNKLST without SYS1.LINKLIB/MIGLIB/CSSLIB/ LINKLIBE/MIGLIBE. NOCHECK should be used ONLY when needed (such as if the pack has a problem).

For the following code to work, you must IPL using the SYSLIB statement to define an alternate LINKLIB data set. The LINKLIB, MIGLIB, CSSLIB, LINKLIBE, and MIGLIBE data sets are all determined during IPL and can never be changed or removed from the LNKLST. To create a LNKLST without those data sets, IPL specifying other data sets (empty PDSs can be used) with the PROGxx SYSLIB statement. The PROGxx statement would contain the following:

```
SYSLIB LINKLIB(SYS2.LINKLIB)
SYSLIB MIGLIB(SYS2.MIGLIB)
SYSLIB CSSLIB(SYS2.CSSLIB)
SYSLIB LINKLIBE(SYS2.SIEALNKE)
SYSLIB MIGLIBE(SYS2.SIEAMIGE)
```

Then IPL with prog=(xx) or prog=(xx,00).

You might use NOCHECK after you have modified SYS1.LINKLIB and want to compress SYS1.LINKLIB. The following procedure is an example:

1. Create a data set that contains a copy of SYS1.LINKLIB.
2. Define a LNKLST set that has the same name as the current LNKLST set but includes the replacement for SYS1.LINKLIB. Specify NOCHECK when defining this LNKLST.
3. Activate the LNKLST set you have defined with the replacement copy of SYS1.LINKLIB.
4. Stop the library lookaside (LLA) procedure. (See "Starting LLA" and "Modifying LLA" in *z/OS MVS Initialization and Tuning Guide*.)
5. Use SET PROG=xx with LNKLST UPDATE or SETPROG LNKLST UPDATE to specify that jobs use the LNKLST set. At this point, SYS1.LINKLIB is not active.
6. Compress SYS1.LINKLIB.
7. Use SET PROG=xx with LNKLST ACTIVATE or SETPROG LNKLST ACTIVATE to activate the original LNKLST set that includes SYS1.LINKLIB.
8. Use SET PROG=xx with LNKLST UPDATE or SETPROG LNKLST UPDATE to specify that jobs use the original LNKLST set and SYS1.LINKLIB.

DSNAME(*dsname*)

The 44-character name of a data set or library that you want to add to the specified LNKLST set or delete from the specified LNKLST set. DSN, LIB, LIBRARY are accepted synonyms for this parameter.

The data set can be a PDS or a PDSE. (IBM suggests that you use PDSEs because of the limitations on the number of extents for a LNKLST concatenation. See “Concatenating data sets to the LNKLST concatenation” on page 703.) For the LNKLST created at IPL, the *dsname* cannot be an alias.

Data sets to be added can be SMS-managed or non SMS-managed. After the system determines the volume and the SMS status of the data set, the following actions result in an error when the system tries to allocate the LNKLST set:

- If the data set in the LNKLST set changes status from SMS-managed to non SMS-managed, or from non-SMS managed to SMS-managed.
- If a non SMS-managed data set in the LNKLST set is deleted and moved to another volume.

In either case, to add the data set after the change has occurred, you must first delete the data set from the LNKLST set and add it again.

[{VOLUME|VOLSER|VOL}(*name*)]

The name of the volume on which the data set resides. The data set must be cataloged. If the volume does not match the name in the catalog, the ADD request fails. The name can be from 1 to 6 characters.

You can use a value of ****** to indicate that the data set is located on the current SYSRES volume. You can use a value of "*MCAT*" to indicate that the data set is located on the volume containing the master catalog.

When the data set is cataloged in a user catalog instead of the master catalog, you can use this parameter. If a data set is cataloged in a user catalog, but not in the system master catalog, you must specify the VOLSER of the volume on which the data set resides. See "Cataloging LNKLST data sets" on page 704.

Default: If you omit this parameter, the system uses the volume indicated in the catalog.

[{ATBOTTOM|ATTOP|AFTER(*dsname*)}]
Indicates where in the LNKLST set you want to place the data set.

ATBOTTOM indicates that you want to place the data set specified on the DSNAME parameter at the bottom of the list of data sets in the LNKLST set.

ATTOP indicates that you want the data set specified on the DSNAME parameter to be added to the beginning of the LNKLST set. The system always places the LINKLIB, MIGLIB, CSSLIB LINKLIBE, and MIGLIBE data sets in that order at the beginning of every LNKLST set in the LNKLST concatenation. If you use ATTOP, the system always places the data set after the CSSLIB data set.

AFTER(*dsname***)** indicates that the system places the data set specified on the DSNAME parameter after the data set specified by *dsname*. You cannot use this parameter to place a data set after the LINKLIB, MIGLIB, CSSLIB LINKLIBE, or MIGLIBE data set in the LNKLST set. Instead, use ATTOP if you want to place the data set immediately after the CSSLIB data set.

Default: If you omit ATBOTTOM, ATTOP, or AFTER, the system adds the data set to the bottom of the LNKLST set.

[CONCAT(NOCHECK|CHECK)]
Indicates whether to check to see if the concatenation defined by the LNKLST set is full.

NOCHECK indicates that you do not want to check to see if the concatenation is full. If the concatenation is full, it will be detected when the LNKLST set is activated.

CHECK indicates that you want to check to see if the concatenation is full. This implies that all the data sets in the LNKLST set must be concatenated. Specification of CHECK causes system processing to take longer.

Default: NOCHECK

MODNAME(*name*)
MODNAME specifies the name of a routine or module to be located in the LNKLST set. MODULE and MOD can be used as synonyms for MODNAME.

```
LNKLST TEST NAME(MY.LNKLST.SET) MODNAME(MYMODULE)
```

{JOB(*jobname*)|ASID(*asid*)}
Specifies the name of the job or address space.

JOB specifies the name of the job or jobs specified by *jobname*. You can use wildcard characters (? or *) for *jobname*. LNKLST UPDATE updates any job whose name matches the specified criteria. The system compares *jobname* to the name of any initiated job or jobs that match, or to the name of the address space.

ASID specifies the address space ID for the job.

```
LNKLST UPDATE JOB(MYJOB)
```

DELAY(*delay***)**
Indicates the number of seconds to delay the completion of the UPDATE operation. The number of seconds can be from 0 - 255.

The LNKLST statement is updated immediately, but the processing for closing and unallocating LNKLST data sets that are no longer in use is delayed by the specified amount of time.

Examples of the LNKLST statement

This example shows how to define LNKLST set MY.LINKLIST and indicate that MY.LINKLIST is to be activated at IPL. The resulting LNKLST concatenation consists of the LINKLIB, MIGLIB, CSSLIB LINKLIBE, and MIGLIBE data sets in that order:

```
LNKLST DEFINE NAME(MY.LINKLIST)
LNKLST ACTIVATE NAME(MY.LINKLIST)
```

This next example shows how to add data sets to the LNKLST set:

```
LNKLST DEFINE NAME(NEWLLSET) COPYFROM(OLDLLSET)
LNKLST ADD NAME(NEWLLSET) DSNAME(dataset.to.be.added)
LNKLST ACTIVATE NAME(NEWLLSET)
```

This example shows how the concatenation of data sets for LNKLST1 is defined.

```
SYSLIB LINKLIB(SYS2.LINKLIB)
SYSLIB MIGLIB(SYS2.MIGLIB)
SYSLIB CSSLIB(SYS2.CSSLIB)
SYSLIB LINKLIBE(SYS2.SIEALNKE)
SYSLIB MIGLIBE(SYS2.SIEAMIGE)
SYSLIB LPALIB(SYS2.LPALIB)
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.SIEALNKE)
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.SIEAMIGE)
LNKLST ADD NAME(LNKLST1) DSNAME(SYS1.AUXLIB) VOLUME(U32PAK)
LNKLST ACTIVATE NAME(LNKLST1)
```

As a result of these PROGxx specifications, the following data sets, in the order specified, are concatenated at IPL:

```
SYS2.LINKLIB,SYS2.MIGLIB,SYS2.CSSLIB,SYS2.SIEALNKE,SYS2.SIEAMIGE,SYS1.LINKLIB,
SYS1.MIGLIB,SYS1.CSSLIB, SYS1.AUXLIB
```

In the example of the LNKLST1 concatenation, note the following:

- The SYSLIB statements specify that SYS2.LINKLIB, SYS2.MIGLIB, SYS2.CSSLIB, SYS2.SIEALNKE, and SYS2.SIEAMIGE replace the system defaults at the beginning of the LNKLST concatenation.
- SYS2.LPALIB is to appear first in the LPALST concatenation.
- SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE must be defined in the LNKLST concatenation. SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE are specified, in that order, after SYS2.LINKLIB, SYS2.MIGLIB, SYS2.CSSLIB, SYS2.SIEALNKE and SYS2.SIEAMIGE.
- SYS1.AUXLIB is specified at the end of the LNKLST1. VOLUME indicates that SYS1.AUXLIB is cataloged on VOLSER U32PAK.

- The LNKLST ACTIVATE statement activates LNKLST1 at IPL.

Syntax format of the LPA statements

Syntax Format of LPA ADD:

```
LPA ADD
     MODNAME(modname,...) | MASK(mask)
     DSNAME(dsname | LNKLST)
     [FIXED|PAGEABLE]
     [PAGEPROTPAGE]
     [,SVCNUMDEC(svcnum) | ,SVCNUMDEC(svcnum,routcode)]
     [,ADDALIAS | ,NOADDALIAS]
```

Syntax Format of LPA DELETE:

```
LPA DELETE
     MODNAME(modname,...)
     FORCE(YES)
     [CURRENT|OLDEST]
```

Syntax Format of LPA CSAMIN:

```
LPA CSAMIN
     (below,above)
```

LPA
Statement type indicating that an action may be performed on LPA.

Note: LPA ADD and LPA DELETE cannot be used during IPL. They are for use in PROGxx members pointed to by SET PROG=xx after IPL.

ADD
Specifies that one or more modules is to be added to LPA.

Default Value: None

DELETE
Specifies that one or more modules is to be deleted from LPA. Only modules added to LPA after an IPL are eligible for dynamic deletion.

Default Value: None

CSAMIN
Specifies the minimum amount of CSA and ECSA that must remain after a module is added to LPA. If the requested ADD operation would reduce the CSA or ECSA below the defined minimum, the system rejects the operation.

Modules added to the system via dynamic LPA processing are placed into CSA or ECSA storage. Therefore, it is important to ensure that the system CSA and ECSA sizes are adequately defined to handle the additional consumption of CSA storage resulting from the issuance of the dynamic LPA request. Further protection can be gained through the use of the CSADMIN parameter.

Default Value: (0,0)

below
The minimum amount of below-16M CSA storage that must remain after the

ADD operation, expressed in the format n|nK|nM, where n is a decimal number, nK is n*1024, and nM is n*1024*1024,

above
The minimum amount of above-16M CSA storage that must remain after the ADD operation, expressed in the format n|nK|nM, where n is a decimal number, nK is n*1024, and nM is n*1024*1024,

MODNAME(modname,...,modname)
modname is the 1-8 character LPA module name or alias. If the last character of the modname is "*", it will be treated as X'C0'. This lets you directly specify the name of a load module that ends with that nonprintable character. Wildcard characters are not supported within *modname.* A maximum of 128 module names can be provided. MOD and MODULE can be used as synonyms of MODNAME.

Default Value: If MODNAME is not specified, MASK must be specified.

MASK(mask)
mask is the 1-8 character mask that is to be applied to all the members of the specified data set. It can contain wildcard characters "*" and "?" and all members that match will be processed.

Default Value: If MASK is not specified, MODNAME must be specified.

DSNAME(dsname)
dsname is the 1-44 character data set name that contains the module(s) or alias(es). When MODNAME is specified, you can specify DSNAME(LNKLST) if you want the system to search the lnklst instead of a particular data set. LNK, LNKLIST, LINKLST or LINKLIST can be specified as an alternative to LNKLST. The data set must be cataloged. It may be allocated as a PDS or PDSE program library.

The attribute of the CSA for each module is assigned as OWNER=SYSTEM. DSN, LIB, and LIBRARY can be used as synonyms of DSNAME.

Default Value: None

FIXED | PAGEABLE
Indicates whether the modules are to be placed in fixed or pageable storage. PAGE can be used as a synonym of PAGEABLE.

Default Value: PAGEABLE

PAGEPROTPAGE
Indicates whether or not to page protect the modules entirely. The default is to page protect the entire module. Be aware that when that default is taken, storage utilization for the modules increases, as each module gets allocated a number of whole pages (so that they can be page protected), rather than just the amount of storage that is truly necessary to load the module.

When PAGEPROTPAGE is requested, however, only the whole pages within each load module are page protected, which keeps the storage use to the minimum amount but which makes it possible that a storage overlay of the beginning or end of the load module can occur.

PPPAGE and PPP can be used as synonyms of PAGEPROTPAGE.

Default Value: Page protect entire modules.

SVCNUMDEC=*svcnum* | **SVCNUMDEC=***(svcnum,routcode)*
SVCNUMDEC, or SVCDEC, identifies the entry within the SVC Table to update. The SVC number, *svcnum*, can be from 0 - 255. For an extended SVC

(109, 116, 122, 137), you must specify the *routcode*, which is the extended SVC routing code. The *routcode* must be within the range that is supported by the SVC. For example, for SVC 109, the *routcode* must be from 0 - 255. The *routcode* for other extended SVC number depends on the release of z/OS. Use the SVCNUMDEC function for updating already-defined SVC Table entries. Do not use it to create new entries, as you might not get all of the attributes that you need.

Default Value: This is not an SVC routine. The SVC table is not updated.

ADDALIAS | NOADDALIAS
ADDALIAS, or ALIAS, indicates to process provided names and aliases of the provided names. NOADDALIAS, or NOALIAS, indicates to process only the names provided. You can use NOADDALIAS to override the default values set by DEFAULTS LPA ADDALIAS.

Default Value: NOADDALIAS, or the value set by DEFAULTS LPA ADDALIAS | NOADDALIAS.

FORCE(YES)
Confirms that the delete requestor understands the ramifications of deleting a module from LPA, when the system can have no knowledge of whether any code is currently executing within the specified module.

Default Value: None. Required parameter.

CURRENT | OLDEST
CURRENT specifies that the current copy is to be deleted. OLDEST specifies that the oldest dynamic copy other than the current one is to be deleted. CUR can be used as a synonym of CURRENT. OLD can be used as a synonym of OLDEST.

Default Value: CURRENT

Syntax format of the REFRPROT statement

```
REFRPROT
```

REFRPROT
Statement type indicating that REFR modules are protected from modification.

Syntax format of the NOREFRPROT statement

```
NOREFRPROT
```

NOREFRPROT
Statement type indicating that REFR modules are not protected from modification. NOREFRPROT is the default.

Syntax format of the TRACKDIRLOAD statement

```
TRACKDIRLOAD
```

TRACKDIRLOAD
Statement type that enables system-wide tracking of directed load modules. TRACKDIRLOAD is the default.

Syntax format of the NOTRACKDIRLOAD statement

```
NOTRACKDIRLOAD
```

NOTRACKDIRLOAD
Statement type that disables system-wide tracking of directed load modules.

Syntax format of the DEFAULTS statements

```
DEFAULTS LNKLST [REQCOPYFROM | NOREQCOPYFROM] [COPYFROMCUR | NOCOPYFROMCUR]
DEFAULTS LPA    [ADDALIAS | NOADDALIAS]
DEFAULTS EXIT   [ EXITTYPE({ALL | INSTALLATION | NOTPROGRAM}) ]
```

ADDALIAS | NOADDALIAS
ADDALIAS, or ALIAS, indicates to process provided names and aliases of the provided names. NOADDALIAS, or NOALIAS, indicates to process only the names provided.

Default: NOADDALIAS.

ALL | INSTALLATION | NOTPROGRAM
Indicates that a DISPLAY PROG,EXIT command uses either ALL, INSTALLATION, or NOTPROGRAM if none of the following exit types are specified:

- ALL, which applies to exits that are either classified or not classified.
- INSTALLATION, or INSTALL, which applies to exits that are classified as installation exits.
- NOTPROGRAM, which applies to exits that are classified as program exits. PROGRAM cannot be specified as a default.
- PROGRAM, or NOTPROG, which applies to exits that are either classified or not classified as an exit that is not an installation exit.

If one of these exit types is specified, the default is not used.

Note: Some exits are not classified as installation or program exits.

Default: ALL

COPYFROMCUR | NOCOPYFROMCUR
Indicates that a LNKLST DEFINE statement, or SETPROG LNKLST,DEFINE command must specify, or must not specify COPYFROM(CURRENT).

If DEFAULTS LNKLST COPYFROMCUR is specified and LNKLST DEFAULTS REQCOPYFROM is specified, the COPYFROM keyword does not have to be specified with the LNKLST DEFINE statement.

COPYFROMCUR, or CFCUR, indicates that the COPYFROM(CURRENT) keyword must be specified.

NOCOPYFROMCUR, or NOCFCUR, indicates that the COPYFROM(CURRENT) keyword must not be specified.

Default: NOCOPYFROMCUR

DEFAULTS
Statement type indicating to set defaults that are applied on LNKLST and LPA statements in PROGxx and also applied on the SETPROG LNKLST, SETPROG LPA, and DISPLAY PROG,EXIT commands.

EXIT
Indicates that this default applies to the DISPLAY PROG,EXIT command.

LNKLST
Indicates that this default applies to LNKLST statements in PROGxx and SETPROG LNKLST commands.

LPA
Indicates that this default applies to LPA statements in PROGxx and SETPROG LPA commands.

REQCOPYFROM | NOREQCOPYFROM
Indicates that a LNKLST DEFINE statement, or SETPROG LNKLST,DEFINE command must specify, or must not specify the COPYFROM or NOCOPYFROM keyword.

REQCOPYFROM, or REQCF, indicates that the COPYFROM keyword must be specified.

NOREQCOPYFROM, or NOREQCF, indicates that the NOCOPYFROM keyword must be specified.

Default: NOREQCOPYFROM

Chapter 77. SCHEDxx (PPT, master trace table, and abend codes for automatic restart)

Use the SCHEDxx member of parmlib to specify the following:

- Size of the master trace table.
- Abend codes that are eligible for automatic restart.
- Programs that are to be included in the program properties table (PPT) and thus receive special attributes.

Note: In previous releases of MVS, SCHEDxx allowed you to define attributes for the eligible device table (EDT) through the EDT statement. As of MVS/SP 5.1, the EDT statement is no longer valid. Instead, use HCD to define attributes for the EDT in the input/output definition file (IODF). For more information about defining the EDT, see *z/OS HCD Planning*.

You can use the following statement types in SCHEDxx:

MT Defines the size of the master trace table if one exists.

PPT
Allows the installation to specify a list of programs that require special attributes or to change the attributes of the IBM-supplied default entries in the PPT (see Table 34 on page 737). The system scans this PPT to determine which, if any, special attributes apply to the program it is initiating.

Note:

1. Usually, you should not add or change programs in the program properties table (PPT) unless the instructions for installing a program direct you to do so. If you do add a program to the PPT, or change an existing entry's properties, ensure that the program can function with the properties you have assigned to it. For example, a program designed to run in program protect key 8 might not be able to run in key 10 because of hardcoded key specifications in the program. If you were to specify KEY(10) in this case, the program will not function correctly.
2. If the processor that your system runs on does not support program protect key 9, do not assign key 9 to any programs. For specific processor models, see *z/OS Planning for Installation*.

RESTART
Allows you to add user abend codes to the list of abend codes that are eligible for automatic restart.

NORESTART
Allows you to delete system and user abend codes from the list of abend codes that are eligible for automatic restart.

Parameter in IEASYSxx (or specified by the operator)

```
SCH={aa             }
    {(aa,L)         }
    {(aa,bb,...[,L])}
    {(,L)           }
```

The two characters (A-Z, 0-9, @, #, or $), represented by aa (or bb, etc.), are appended to SCHED to identify one or more SCHEDxx members of parmlib.

If the L option is specified, either on the SCH= keyword in the IEASYSxx parmlib member or in response to the 'SPECIFY SYSTEM PARAMETERS' prompt, the system displays the contents of SCHEDxx on the operator's console during IPL.

Modifying the PPT between IPLs

You can use the SET SCH command to dynamically modify the contents of the PPT. The SET SCH command causes the system to replace the current PPT definitions with the IBM-supplied default PPT definitions, and the PPT definitions from one or more SCHEDxx members that you specify on the command. The new PPT definitions take effect immediately (that is, without requiring a reIPL of the system).

Note: Using the SET SCH command to change the properties of a program from NOCRITICALPAGING to CRITICALPAGING does not result in the system paging in any storage associated with this program that is currently paged out. It only guarantees that storage associated with the program that is currently paged in will not be paged out. To ensure that the associated storage of a program is all paged in after dynamically changing the properties of the program, end and restart the program.

For more information about the SET SCH command, see *z/OS MVS System Commands*.

Syntax rules for SCHEDxx

The following rules apply to the creation of SCHEDxx:

- Use columns 1 through 71. Do not use columns 72-80, because the system ignores these columns.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- A statement type consists of 1-10 characters.
- A statement must begin with a valid statement type followed by at least one blank.
- A statement ends with the beginning of the next valid statement type or EOF.
- A statement can be continued even though there is no explicit continuation character.
- Multiple statement types are accepted.
- Operands must be 60 characters or less and may not span multiple records.
- Operands must be separated by valid delimiters. Valid delimiters are a comma, a blank, or column 71. If the operand contains parenthesis, then the right parenthesis is accepted as a valid delimiter.

- Multiple occurrences of a delimiter are accepted but treated as one.

IBM-supplied default for SCHEDxx

None.

IBM-supplied sample member SCHEDxx

IBM provides a sample member, named SCHED00, in SYS1.SAMPLIB. SCHED00 contains sample statements for various programs; these statements are is not necessarily the IBM-supplied values.

To create a SCHEDxx member for your installation, you can copy SCHED00 to a SCHEDxx member and modify it according to your needs.

Statements/parameters for SCHEDxx

The SCHED statements are described as follows:

```
MT SIZE{(nnnK)}
       {(NONE)}
       {(24K) }
```

Specifies the size (in kilobytes) of the master trace table, which is used by the TRACE command. The system creates the master trace table during master scheduler initialization. By default, the master trace table is 24 kilobytes in size.

To specify a different size for the master trace table, specify *MT SIZE(nnnK)*, where *nnn* is any value from 16 to 999. You must add the letter 'K' for the increments (kilobytes), and enclose the entire value in parentheses.

For example, to have the system create a master trace table of 50 kilobytes, specify the following MT statement in SCHEDxx:

```
MT SIZE(50K)
```

If you do not want a master trace table to be created, specify **MT SIZE** as **(NONE)**, as follows:

```
MT SIZE(NONE)
```

No table is created.

If you specify more than one MT statement in a SCHEDxx member, the system will use the first occurrence and ignore any subsequent MT statements.

Default: MT SIZE(24K)

NORESTART CODES(code,code...)

Specifies which system and user abend codes are to be deleted from the table of abend codes that are eligible for automatic restart.

RESTART and NORESTART allows the installation to customize the list of abend codes that are eligible for automatic restart. This list is created by merging the user abend codes specified on the RESTART statement with the list of IBM-supplied system abend codes. A user abend code cannot have the same number as a system abend code. The system will ignore a user abend code that is a duplicate of a system abend code. The IBM-supplied system abend codes are listed in Table 33 on page 732.

Table 33. IBM-supplied system abend codes

Code	Code	Code	Code
001	20A	422	813
031	213	513	837
033	214	514	906
03A	217	613	913
0A3	2F3	614	926
0B0	313	626	937
0F3	314	637	A14
100	317	700	B14
106	32D	714	B37
113	413	717	C13
117	414	737	E1F
137	417	806	E37

Value Range: 0-FFF (Hex)

RESTART CODES(code,code...)
Specifies the user abend codes that are to be eligible for automatic restart. These are to be added to the system abend codes that are supplied by IBM.

Value Range: 0-FFF (Hex)

PPT
Allows the installation to specify a list of programs that require special attributes. Sometimes, your application programs will need to possess special properties to run as efficiently and securely as possible. For example, an application that requires access to fetch-protected system data will need a system key (0-7) instead of the usual problem program key of 8. Or, for example, an application that cannot run V=R, but must not be swapped out because of real-time considerations, will need to be identified to the system as nonswappable. Entries for these programs are created in the program properties table (PPT).

PGMNAME(name)
PGMNAME(name) identifies by name the program, specified in the PGM parameter on the EXEC statement for a job or step, that requires special attributes. It must consist of an alphabetic or national (#, @, or $) character followed by 0 to 7 alphanumeric or national characters.
PGMNAME(name)is required on the PPT statement. A program specified in the PPT will obtain special attributes only if all of the following are true:

- The program is fetched from an authorized library or from the link pack area (pageable, modified, fixed, or dynamic LPA).
- All STEPLIB data sets are authorized if a STEPLIB DD (or concatenation) exist for the step.
- All JOBLIB data sets are authorized if a JOBLIB DD (or concatenation) exists for this job and no STEPLIB DD exists for this step.

Otherwise, problem program attributes are assigned. All programs described by PPT entries must come from an APF library or concatenation, or from the link pack area.

Note:

1. To override an IBM-supplied entry in the PPT, use the same program name as the IBM-supplied entry.
2. Although you cannot remove entries from the PPT, you can create the effect of removing an entry from the PPT by specifying the program name without special attributes.

   For example, a JES2 installation might (for auditing purposes) want to "remove' the IATINTK entry from the PPT by including the following statement in the SCHEDxx member:

   ```
   PPT PGMNAME(IATINTK)
   ```

   This statement overrides the existing IATINTK entry in the PPT, and causes any APF-authorized program named IATINTK to run with the same attributes as a problem program.
3. If you specify more than one PPT statement with the same program name, the systems accepts only the first occurrence of the name and issues error message IEF732I for all subsequent occurrences of that name.
4. When referring to a DD statement, the system does not honor requests for special properties as defined in the PPT.

Default: NONE

{CANCEL}|{NOCANCEL}
The program specified on PGMNAME can be canceled (CANCEL) or cannot be canceled (NOCANCEL).

Default: CANCEL

KEY(n)
The program specified on PGMNAME is to have the protection key (n) assigned to it. The range of values for n is 0 through 15. A KEY value greater than 8 requires special consideration. Usually, keys greater than 8 are reserved for V=R programs, and the system assigns these keys dynamically. If the KEY field is specified for a V=R program, ensure that no other V=R program runs at the same time with the same key. A V=V program may run with a KEY value greater than 8 when V=R programs are running with channel program translation.

Value Range: 0-15

Default: Key 8

{SWAP}|{NOSWAP}
The program specified on PGMNAME is swappable (SWAP) or non-swappable (NOSWAP).

Default: SWAP

{PRIV | NOPRIV}
The program specified on PGMNAME is privileged (PRIV), or not privileged (NOPRIV).

A task marked PRIV is put in the SYSSTC service class if it is not explicitly classified in the WLM classification rules.

Default: NOPRIV

{DSI | NODSI}
The program specified on PGMNAME requires data set integrity (DSI) or does not require data set integrity (NODSI). Data set integrity means that the job holds an ENQ for the data sets it allocates. The DSI/NODSI option applies to batch allocation only. Dynamic allocation uses its input

parameters to determine whether to enqueue on data sets. JES2 is an exception in that it honors the PPT specifications and uses DSI/NODSI for dynamic allocations of the JES2 PROCLIB data sets.

If DSI is specified, the system acquires an ENQ for all data sets requested by the program. The ENQ is exclusive or shared, depending on the disposition on the DD request.

If NODSI is specified, the system still issues an ENQ for all data sets requested by the program. However, the ENQ is released before the problem program is started.

Jobs for which NODSI is specified are not started if the job contains either a JOBLIB or STEPLIB, and both of the following conditions are true:

- The job cannot get the ENQ on the following types of data set names:
  - GDG absolute generation data set name (unless the absolute generation data set name is specified on the JCL).
  - Real data set name (when its corresponding alias data set name is specified on the DD statement in the JCL).
- The installation has specified WAITALLOC(NO) on the SDSN_WAIT keyword in the ALLOCxx member of SYS1.PARMLIB, or has taken the default. For more information about the SDSN_WAIT keyword, see Chapter 4, "ALLOCxx (allocation system defaults)," on page 71.

Note:

1. For NODSI, the job must be a one-step job. If the job is not a one-step job, NODSI is nullified and the system issues message IEF188I. All other properties remain in effect.
2. NODSI is not honored for jobs that use unauthorized JOBLIBs or STEPLIBs. The system assumes DSI for these jobs.

Default: DSI

{PASS| NOPASS}

The program specified on PGMNAME can or cannot bypass security protection (password protection and RACF). PASS indicates that security protection is in effect; NOPASS indicates that security protection is not required. PASS is the default.

Jobs that request the bypass-security-protection property will receive the property. However, a protected data set cannot be deleted via JCL (that is, by coding a disposition of DELETE) without the password. This is because the bypass-security-protection property is turned off when the job enters deallocation processing.

Note that the NOPASS parameter option does not apply to the STEPLIB or JOBLIB DD from which the named program is to be fetched, and that the address space in which the program is executing must have the security access required to open the JOBLIB or STEPLIB dataset.

Default: PASS

{SYST | NOSYST}

The program specified on PGMNAME is a system task and is not timed (SYST) or is not a system task and is to be timed (NOSYST). For SYST, the program must be a one-step job started by a START or MOUNT command. If these conditions are not met, SYST is nullified and the system issues message IEF188I. All other properties remain in effect.

If procedures are multistep or if NOSYST is specified, TIME=1440 may be required to prevent timeout.

In goal mode, a task marked SYST is put in the SYSSTC service class unless it is explicitly classified in the WLM classification rules before any SPM rule entries.

Default: NOSYST

SPREF
The program specified on PGMNAME must have all private area short-term fixed pages assigned to preferred (nonreconfigurable and non-V=R) storage frames.

Default: None

LPREF
The program specified on PGMNAME must have all private area long-term fixed pages assigned to preferred (nonreconfigurable and non-V=R) storage frames.

Default: None

NOPREF
The program specified on PGMNAME does not need to have all private area short-term fixed pages assigned to preferred storage frames. That is, the program's short-term fixes are in fact short-term fixes and can be allowed in reconfigurable storage.

Default: None

Notes on using SPREF, LPREF, and NOPREF:

1. LPREF is meaningless for programs that issue SYSEVENT TRANSWAP to become nonswappable.
2. Use of SPREF, LPREF, and NOPREF forces the program's private area fixed pages and LSQA pages into preferred storage frames, thus ensuring that they will not prevent taking storage offline.
3. Use SPREF, LPREF, or NOPREF for programs whose fixed pages could prevent the successful execution of a VARY STOR,OFFLINE command (or could fragment the V=R area) if they were assigned frames in reconfigurable or V=R storage.
4. SPREF and LPREF are significant for swappable programs(SWAP) that have a special requirement for preferred frames.
5. SPREF and LPREF are intended for use with authorized swappable programs that issue SYSEVENT DONTSWAP to become nonswappable for short periods (rather than using NOSWAP).
6. SPREF and LPREF should be specified when the preferred storage requirements for a nonswappable program are unknown. This will ensure that all fix requests and LSQA requests will get preferred storage.
7. NOPREF is significant only for users of the SYSEVENT TRANSWAP. This includes V=R job steps, nonswappable programs, applications using the BTAM OPEN function, and any applications using a system function that issues SYSEVENT TRANSWAP.
8. For an application program that issues SYSEVENT DONTSWAP, or issues SYSEVENT REQSWAP followed by a SYSEVENT DONTSWAP, do **one** of the following:
   - List the program in the PPT and specify SPREF and LPREF.

This allows the program to be attached as swappable, but all LSQA and private area fixed pages will be assigned preferred frames during the entire job step.

- Remove SYSEVENTs REQSWAP and DONTSWAP from the program. List the program in the PPT as nonswappable (NOSWAP) and do not specify NOPREF.

  This allows the program to be attached as nonswappable, and all LSQA and private area fixed pages will be assigned preferred frames during the entire job step.

9. An I/O device requiring operator intervention can interfere with taking storage offline by fixing pages in reconfigurable storage. For example, a printer requiring action to be taken, or a tape unit with a mount pending. Until the required action is completed, the storage associated with the I/O operation cannot be taken offline. This problem *cannot* be bypassed by using SPREF, LPREF, or NOPREF.

{HONORIEFUSIREGION} | {NOHONORIEFUSIREGION}
When HONORIEFUSIREGION is specified, region and MEMLIMIT settings specified through, or otherwise affected by the IEFUSI exit take effect.

When NOHONORIEFUSIREGION is specified, region and MEMLIMIT values specified through, or otherwise affected by the IEFUSI exit are ignored. In other words, when NOHONORIEFUSIREGION is specified for a program, MEMLIMIT values or limits cannot be altered through the IEFUSI exit. This PPT attribute is used to bypass IEFUSI region controls for specific programs that require a larger than normal region in order to successfully execute.

For more information about controlling region size and region limits using the IEFUSI exit, see *z/OS MVS Initialization and Tuning Guide*.

Default: HONORIEFUSIREGION

{CRITICALPAGING} |{NOCRITICALPAGING}
The program specified on PGMNAME owns storage that is critical or is not critical for Hyperswap operation. CRITICALPAGING indicates that the storage associated with the program specified on PGMNAME is critical for Hyperswap operation. Therefore, storage management should never page out this storage if possible. NOCRITICALPAGING (the default) indicates that the storage associated with the program specified on PGMNAME is not critical for Hyperswap operation and can be paged out at any time.

Note:

1. IBM Supplied PPT entries marked CRITICALPAGING cannot be overridden.
2. Use this parameter in conjunction with the CRITICALPAGING switch to increase the likelihood of a successful HyperSwap operation. (See the FUNCTIONS statement of COUPLExx.)

Default: NOCRITICALPAGING

Program properties table (PPT)

Table 34 on page 737 shows the IBM-supplied program properties table values. The program name listed in the table is the PGM parameter value on the EXEC statement in a cataloged procedure in the SYS1.PROCLIB system library. The *membername* specified in the START operator command is the name of the

SYS1.PROCLIB member. See "Starting a system task from the console" in *z/OS MVS System Commands* for more information.

Table 34. IBM-supplied program properties table (PPT) values

Program Name	Program Description	NC	NS	PR	ST	ND	BP	Key	2P	1P	NP	NH	CP
AHLGTF	GTF	x	x		x			0			x		
AKPCSIEP	ISP		x		x	x		1			x		
ANFFIEP	IP Printway		x		x	x		1					
APSHPOSE	PSF AFP Download Plus		x		x	x		1			x		
APSKAFPD	PSF Download		x		x	x		1			x		
APSPPIEP	PSF		x		x	x		1			x		
ASBSCHIN	APPC/MVS Scheduler Address Space (ASCH)		x		x			1	x	x			
ASBSCHWL	APPC/MVS Message Log Writer			x				1					
ATBINITM	APPC/MVS Address Space		x		x			1	x	x			
ATBSDFMU	APPC/MVS SDFM Utility			x				1					
AVFMNBLD	AVM	x	x		x			3			x		
AXR	System REXX	x		x									
AXRRXTSS	System REXX			x									
BBGCTL	WAS		x	x				2					
BBGDAEMN	WAS		x	x	x			2					
BBOCTL	WAS		x	x				2					
BBODAEMN	WAS		x	x	x			2					
BHII1TPC	Basic HyperSwap BHIHSRV address space	x		x	x			0					
BNJLINTX	Netview for z/OS Start Up without SVC76		x					8					
BPEINI00	IMS base primitive environment		x		x			7					
BPXBATA2	WAS			x				2					
BPXINIT	OMVS	x	x		x			0					
BPXPINPR	OMVS	x			x			8	x	x			
BPXVCLNY	OMVS		x	x	x			8					
CBRIIAS	OTIS				x			5					
CBROAM	OAM		x		x			5					
CNLSSDT	MVS Message Service (MMS)		x		x			0	x	x			
COFMINIT	VLF		x		x	x	x	0					
COFMISDO	DLF	x	x		x	x	x	0					
CQSINIT0	IMS CQS		x		x			7			x		
CSFINIT	ICSF Initialization	x	x	x	x			0					
CSQYASCP	WebSphere MQSeries®				x			7					
CSVLLCRE	LLA		x		x		x	0					
CSVVFCRE	Virtual Fetch		x		x			0					

Table 34. IBM-supplied program properties table (PPT) values (continued)

Program Name	Program Description	NC	NS	PR	ST	ND	BP	Key	2P	1P	NP	NH	CP
DFHSIP	CICS driver		x					8			x		
DFSMVRC0	IMS Control Program		x		x			7					
DFSYSVI0	IMS OTMA callable service		x		x			7					
DSIMNT	NetView for z/OS Start Up without SVC76		x					8					
DSNUTILB	DB2 Batch							7					
DSNYASCP	DB2		x		x			7					
DUIFT000	NetView GMFHS program		x					8					
DWW1SJST	CICSVR	x	x	x	x			5					
DXRRLM00	IMS Manager		x		x			7					
EKGTC000	NetView RODM program	x	x					8					
EPWINIT	FFST™	x	x			x	x	0			x		
ERBMFMFC	RMF		x		x	x		8					
ERB3GMFC	RMF		x		x	x		8					
EZAPPAAA	NPF		x					8					
EZAPPFS	NPF		x					1					
EZBADNR	Automated Domain Name Registration		x	x	x			8					
EZBLBADV	Load Balancing Advisor		x	x	x			8					
EZBLBAGE	Load Balancing Agent		x	x	x			8					
EZBREINI	CommServer- Resolver	x	x	x	x			6					
EZBTCPIP	TCP/IP Address Space	x	x	x	x			6	x	x			
EZBTNINI	TN3270 Telnet procedure	x	x	x	x			6					
FPGINIT	Hardware accelerator manager	x	x		x		x	0					
FRDRVS00	IMS ORS		x		x			7					
GDEICASB	DFP/DFM		x		x			5			x		
GDEISASB	DFP/DFM				x			5			x		
GDEISBOT	DFP/DFM		x		x			5			x		
GEOXCFST	GDPS Address		x	x	x			8					x
GTZINIT	GTZ initialization			x	x			8				x	
HASJES20	JES2	x	x		x	x		1				x	
HCW4MDDM	DDM address		x		x			8		x			
HHLGTF	GTF	x	x		x			0			x		
HSAPINIT	System Automation Manager		x	x				8					
HWIAMIN1	Base control program internal interface (BCPii) address space		x		x			4					
HWIAMIN2	Base control program internal interface (BCPii) address space		x		x			4	x	x			x

Table 34. IBM-supplied program properties table (PPT) values (continued)

Program Name	Program Description	NC	NS	PR	ST	ND	BP	Key	2P	1P	NP	NH	CP
IASXWR00	External Writer	x			x			1					
IATCNDTK	JES3	x	x	x	x			1					
IATINTK	JES3	x	x		x	x		1				x	
IATINTKF	JES3 FSS		x		x	x		1					
IATINXM	JES3 Auxiliary Address Space	x	x	x	x			1	x	x			
IAZNJTCP	JES - IAZ			x	x			0					
IDAVSJST	SMSVSAM Address Space	x	x	x	x			5					
IEAVTDSV	Dumping Services			x	x		x	0	x	x			
IEDQTCAM	TCAM (see Note)		x					6			x		
IEEMB860	Master	x	x		x	x	x	0					
IEEVMNT2	Mount Command	x			x			0					
IEFIIC	Initiator	x		x	x			0					
IFASMF	SMF	x	x	x	x	x		0					
IFDOLT	OLTEP							8	x	x			
IGDSSI01	SMS	x	x		x		x	5					
IGG0CLX0	CAS	x	x	x	x	x		0	x				
IHLGTF	GTF	x	x		x			0			x		
IHVOINI	Automation I/O Operations		x					8					
IKTCAS00	TCAS	x		x	x			6					
IOSHMCTL	Basic HyperSwap management address space		x	x	x			0					x
IOSHSAPI	Basic HyperSwap API services address space		x	x	x			0					x
IOSVROUT	IOS	x	x		x		x	0				x	
IQCINI0$	IMS queue control program		x		x			7					
IQPINIT	PCIE services	x	x		x		x	0					
IRRSSM00	RACF	x	x	x	x			2					
ISFHCTL	SDSF		x					1					
ISTINM01	VTAM	x	x		x		x	6			x		
ITTTRCWR	CTRACE Writer Address Space	x	x	x	x		x	0	x	x			
IWMINJST	WLM	x	x		x			0	x	x			
IXCINJST	XCF	x	x		x			0	x	x			x
IXGBLF00	System Logger Address Space		x		x			0					
IXGBLF01	System Logger Address Space	x	x		x			0		x		x	
IXZIX00	JES Common Coupling Address Space	x	x	x	x			1					

Table 34. IBM-supplied program properties table (PPT) values (continued)

Program Name	Program Description	NC	NS	PR	ST	ND	BP	Key	2P	1P	NP	NH	CP
MVPTNF	TNF Address Space	x	x	x	x			0					
MVPXVMCF	VMCF Address Space	x	x	x	x			0					
SNALINK	SNALINK Address Space		x		x			6					
Note: Although telecommunications access method (TCAM) was withdrawn from service in 2002, IEDQTCAM remains in the PPT table for compatibility.													

Table 35 defines the column headings used in Table 34 on page 737.

Table 35. Explanation of column headings in PPT

Synonym	Meaning	SCHEDxx keyword
NC	Non-cancelable	NOCANCEL
NS	Non-swappable	NOSWAP
PR	Privileged	PRIV
ST	System task	SYST
ND	No data set integrity	NODSI
BP	Bypass password protection	NOPASS
Key	PSW key for this program	KEY(x)
2P	Second level preferred storage	SPREF
1P	First level preferred storage	LPREF
NP	No preferred storage	NOPREF
NH	No honor IEFUSI region settings	NOHONORIEFUSIREGION
CP	Critical paging	CRITICALPAGING

Chapter 78. SMFPRMxx (system management facilities (SMF) parameters)

The SMFPRMxx member allows you to control how system management facilities (SMF) works at your installation. You can use SMFPRMxx parameters to:

- Identify the system on which SMF is active.
- Specify global values for interval recording and synchronization that SMF, RMF, and other requestors can use to schedule the execution of their interval functions.
- Specify the data sets or log streams to be used for SMF recording.
- Allow compression of SMF data before recording to log streams.
- Specify the system identifier to be used in all SMF records.
- Select the SMF record types and subtypes SMF is to generate.
- Allow the operator to change the SMF parameters established at IPL.
- Specify the job wait time limit.
- Specify whether SMF is to invoke installation-supplied SMF exit routines.
- Specify whether the SMF dump program is to attempt to recover from abends.
- Specify the system response when SMF has used all of the buffered storage in its address space and is recording on data sets.
- Specify the system response when the last SMF data set is filled and no other data sets are available for use.
- Specify the amount of real time that SMF allows data to remain in an SMF buffer before it is written to a recording data set or a log stream.
- Specify the installation default MEMLIMIT.
- Specify whether only registration data and not usage data is to be recorded when using the IFAUSAGE macro.
- Specify valid user exists for the IFASMFDP and IFASMFDL programs.
- Specify whether or not to suppress empty execute channel program (EXCP) using the EMPTYEXCPSEC parameter.
- Specify whether SMF record flood support is active and the filter for the SMF record flood.

Note: If you specify that SMF is to record on data sets, the SMF data sets must be cataloged on DASD. If there are no data sets for SMF to use, SMF buffers data until you specify a data set for SMF to use. If SMF runs out of buffers, there might be a loss of data.

You can specify SMF parameters in several ways:

- Before the first IPL of a newly generated system by creating an SMFPRMxx parmlib member.
- At each initialization of SMF by entering the parameters at the console.
- During SMF execution, by using the SET SMF command to specify a different SMFPRMxx parmlib member or by using the SETSMF command to replace one or more previously defined SMF parameters.

See *z/OS MVS System Commands* for more information about the commands used with SMF. For information about setting up and using SMF, see *z/OS MVS System Management Facilities (SMF)*.

Using the SET command for SMFPRMxx

The SET operator command can be used to modify the SMF recording options dynamically by specifying which SMFPRMxx parmlib member is to be used. Also, if SMF terminates, the SET SMF command can be used to restart SMF.

Using the SET command, the installation can replace all the existing SMF options. For example, an installation can activate SMF recording after an IPL in which NOACTIVE is specified by using the SET command and choosing the parmlib member that contains the ACTIVE option. The SET command, however, cannot change the SID parameter.

For each IPL, a maximum of eight subsystems can be defined to SMF (through the SUBSYS parameter). This is a combined total of those specified at IPL and subsequent SET commands. If the maximum is reached, no new subsystems may be added. Those subsystems previously specified can be given different options.

Using the SETSMF command for SMFPRMxx

In contrast to the SET SMF command, which allows an installation to specify a different SMFPRMxx parmlib member or to restart SMF, the SETSMF operator command allows an installation to:

- Add a SUBPARM parameter value to those SMFPRMxx parameter values already set for this IPL.
- Replace SMFPRMxx parameter values (except ACTIVE, PROMPT, SID, and EXIT) with new ones for this IPL.

If the SMFPRMxx parameter values set for this IPL include NOPROMPT, the operator cannot use the SETSMF command.

Parameter in IEASYSxx (or supplied by the operator)

SMF=xx
The two alphanumeric characters, represented by xx, are appended to SMFPRM to identify the SMFPRMxx member of parmlib. If the parameter is not specified either in IEASYSxx or by the operator, the system uses parmlib member SMFPRM00. If the parmlib does not contain an SMFPRM00 member, the system uses the defaults provided by SMF.

Support for system symbols

You can specify system symbols in SMFPRMxx. In addition, you can specify the &SID symbol when naming SMF data sets (on the DSNAME parameter).

For information about how to use system symbols in shared parmlib members that require unique values, see Chapter 2, “Sharing parmlib definitions,” on page 33. For specific information about using symbols to name SMF data sets, see the description of the DSNAME and SID parameters in the description of this parmlib member.

Syntax rules for SMFPRMxx

The following rules apply to the creation of SMFPRMxx:

- Use columns 1 through 72; however, avoid using column 72 when system symbols are coded on the same line.
- Avoid embedded blanks.
- Comments can appear in columns 1-71 and must begin with "/*' and end with "*/".
- Enter each parameter in the format: keyword (value).
- Indicate continuation by placing a comma after the last entry on a record, followed by a blank before column 72.
- Limit SMFPRMxx to no more than 897 lines. If you exceed this limit, the system ignores the values specified in SMFPRMxx and uses default values instead. In addition, the system issues an informational message to the operator, indicating that SMFPRMxx is too large and that default values are to be used instead.

For a filter to have both MSG and DROP processing, the DROP filter will not be processed until the MSG filter has already been hit.

The following statements show an example of how to specify the syntax:

```
SID(3090),ACTIVE,
DSNAME(SYS1.MANA,SYS1.MANB,SYS1.MANC),
JWT(0030),SYS(TYPE(00:120),NOEXITS,
INTERVAL(004000),DETAIL)
```

In this next example, IFASMF.STREAM1 will use a maximum of 256M of buffer space and IFASMF.STREAM2 will use up to 1024M.

```
DSPSIZMAX(1024M)
LSNAME(IFASMF.STREAM1,TYPE(30,89),DSPSIZMAX(256M))
LSNAME(IFASMF.STREAM1,TYPE(127:255))
```

Syntax format of SMFPRMxx

```
INTVAL(mm)

SYNCVAL(mm)

ACTIVE|NOACTIVE

RECORDING(DATASET | LOGSTREAM)

DEFAULTLSNAME(logstreamname,NOBUFS({HALT}|{MSG}),BUFUSEWARN(nn),
              DSPSIZMAX(nnnnM | nG),COMPRESS(PERMFIX(nnnnM)) )

LSNAME(logstreamname,TYPE({aa,bb}|{aa,bb:zz}|{aa,bb:zz,...}), NOBUFS({HALT}|{MSG}),
        BUFUSEWARN(nn), DSPSIZMAX(nnnnM | nG),COMPRESS(PERMFIX(nnnnM)) )

DSNAME {(dataset)}

LISTDSN| NOLISTDSN

SID {(xxxx)                      }
    {(xxxx,SYSNAME(sysname))     }
    {(xxxx,ser#[,ser#...])       }
    {(xxxx,COMBIN(ser#[,ser#...]))}

REC({(ALL) })
    {(PERM)}
```

```
MAXDORM(mmss)| NOMAXDORM

MEMLIMIT(NOLIMIT)
         nnnnnM
         nnnnnG
         nnnnnT
         nnnnnP
EMPTYEXCPSEC{(NOSUPPRESS)}
            {(SUPPRESS)  }

STATUS([hhmmss]           )|NOSTATUS
       [SMF[,SYNC|NOSYNC]]

         JWT(hhmm)
         SWT(hhmm)
         TWT(hhmm)

         DDCONS {(YES)| (NO)}
         PROMPT{IPLR}|NOPROMPT
               {LIST}
               {ALL }

           AUTHSETSMF | NOAUTHSETSMF
         SYS([TYPE]     )
             [,INTERVAL]
             [,EXITS]
             [,DETAIL]

         TYPE    {aa,bb(cc)       }
         NOTYPE ({aa,bb:zz        })
                 {aa,dd(cc:yy),...}
                 {aa,bb(cc,...)   }

         NOINTERVAL|INTERVAL([hhmmss]          )
                             [SMF[SYNC|NOSYNC]]

         EXITS (exit name,exit name,...)|NOEXITS

         NODETAIL|DETAIL

         SUBPARM (name(parameter))

         SUBSYS(name,[TYPE]     )
                     [,INTERVAL]
                     [,EXITS]
                     [,DETAIL]

         DUMPABND {(RETRY)  }
                  {(NORETRY)}

         NOBUFFS {(MSG) }
                 {(HALT)}

         LASTDS {(MSG) }
                {(HALT)}

         MULCFUNC|NOMULCFUNC

         BUFSIZMAX (nnnnM | 1G)

         BUFUSEWARN (nn)

SMFDLEXIT({ USER1({exit name,exit name,...)} | NOUSER1 },
          { USER2({IRRADU00,exit name,...)} | NOUSER2 },
          { USER3({IRRADU86,exit name,...)} | NOUSER3 }

SMFDPEXIT({ USER1({exit name,exit name,...)} | NOUSER1 },
          { USER2({IRRADU00,exit name,...)} | NOUSER2 },
          { USER3({IRRADU86,exit name,...)} | NOUSER3 }
```

```
FLOOD({ON|OFF})
FLOODPOL(
    {TYPE({aa,bb})
              {aa,bb:zz}
              {aa,bb:zz,...}) },
     RECTHRESH(xxxx),
     INTVLTIME(xxxx),
     MAXHIGHINTS(xxxx),
     ENDINTVL(xxxx),
     ACTION({MSG|DROP})
    )
MAXEVENTINTRECS(nn)

DSPSIZMAX(nnnnM | nG))

PERMFIX(nnnnM)

SMF30COUNT | NOSMF30COUNT
```

IBM-supplied default for SMFPRMxx

None.

IBM-supplied sample for SMFPRMxx

IBM provides sample member, SMFPRM0L, in SYS1.SAMPLIB. You should modify this member according to your system requirements. You can place alternate values, plus additional values, in one or more alternate SMFPRMxx members.

Parameters for SMFPRMxx

This section describes the statements and parameters of SMFPRMxx.

INTVAL(*mm*)
Specifies the length of time (in minutes) from the end of an SMF global recording interval to the end of the next interval. For example, if you specify INTVAL(15), the SMF global recording interval ends every 15 minutes. INTVAL is a global interval value that other requestors, such as RMF, can use to schedule interval functions to execute in conjunction with the SMF interval function.

Choose a two-digit global interval value, *mm*, that divides evenly into 60 (01, 02, 03, 04, 05, 06, 10, 12, 15, 20, 30, and 60). Otherwise, the system can synchronize only the first interval.

Range:
01-60

Default:
30

Note: Only SMF records are controlled by the INTVAL and SYNCVAL parameters. The INTVAL and SYNCVAL parameters can influence other record types, such as when activated by the RMF Monitor I "SYNC(SMF)" option to write record types 70 through 79.

SYNCVAL(*mm*)
Specifies the two-digit global synchronization value (in minutes) for the SMF global recording interval, synchronizing the recording interval with the end of

the hour on the TOD clock. For example, if you specify SYNCVAL(15), the global recording interval is synchronized to 15 minutes past the hour. If you also specify INTVAL(30), SMF global recording intervals end at 15 minutes and 45 minutes past the hour.

When you specify the SYNCVAL parameter for interval synchronization, specify the global interval value with the INTVAL parameter (unless you accept the default for INTVAL).

Range:
00-59

Default:
00

Note: Only SMF records can be controlled by the INTVAL and SYNCVAL parameters. The INTVAL and SYNCVAL parameters can influence other record types, such as when activated by the RMF Monitor I "SYNC(SMF)" option to write record types 70 through 79.

ACTIVE|NOACTIVE
Specifies whether SMF recording is to be active.

Default: ACTIVE

RECORDING(DATASET | LOGSTREAM)
Specifies whether you want to write SMF records to MANx data sets or log streams. You can specify both SMF data sets and log streams in one SMFPRMxx member and then set either RECORDING(DATASET) or RECORDING(LOGSTREAM) on the SETSMF command to switch between data set and log stream recording.

Default: RECORDING(DATASET)

LSNAME(*logstreamname*,TYPE({aa,bb}|{aa,bb:zz} | {aa,bb:zz,...}),[NOBUFS({HALT|MSG})],[BUFUSEWARN(nn)],[DSPSIZMAX(nnnnM | nG)][,COMPRESS[(PERMFIX(nnnnM))]])
This optional parameter allows you to specify a log stream in which you want to record particular SMF record types on the TYPE subparameter. The log stream name must be composed as follows:

- The first seven characters must be 'IFASMF.'.
- You must have a minimum of 8 characters.
- You must have a maximum of 26 characters.
- It must conform to other log stream naming conventions as documented in IXGINVNT in *z/OS MVS Programming: Assembler Services Reference IAR-XCT*.

You can use system symbols and the &SID symbol in SMF log stream names. The resolved substitution text for the &SID system symbol is the system identifier specified on the SID parameter in SMFPRMxx. &SID can be used only to name resources in SMFPRMxx; you cannot specify &SID in other parmlib members.

Table 36 on page 747 shows examples of log stream names that use the &SYSNAME system symbol and the &SID symbol. The table shows the substitution texts for &SYSNAME and &SID, the data set names that specify the symbols, and the resolved texts for the log stream names.

Table 36. Examples of log stream names that use system symbols

Substitution Text		Log stream name	Resolved log stream name
&SID	&SYSNAME		
WRR1	—	IFASMF.SYS1.&SID..DATA	IFASMF.SYS1.WRR1DATA
WRR1	—	IFASMF.SYS1.SID&SID..DATA	IFASMF.SYS1.SIDWRR1.DATA
WRR1	SP52	IFASMF.SYS1.&SYSNAME; &SID..DATA	IFASMF.SYS1.SP52WRR1.DATA
—	SP52	IFASMF.SYS1.&SYSNAME	IFASMF.SYS1.SP52

If you specify an incorrect log stream name, the system issues message IFA700I.

If you specify the same record type on two or more different LSNAME parameters, the system writes the record to all specified log streams. For example, you might have an SMFPRMxx parmlib member with the following contents:

```
DEFAULTLSNAME(IFASMF.DEFAULT)
LSNAME(IFASMF.PERF,TYPE(30,89))
LSNAME(IFASMF.JOB,TYPE(30))
RECORDING(LOGSTREAM)
```

This allows you to collect job-related SMF data in the JOB log stream, and performance-related SMF data in the PERF log stream. Record type 30 fits into both categories, so you can specify that it is written to both log streams. Note that this arrangement can result in duplicate records being recorded.

TYPE({aa,bb}|{aa,bb:zz} | {aa,bb:zz,...})
TYPE specifies the SMF record types that SMF is to collect to the specified log stream on the LSNAME parameter. aa, bb, and zz are the decimal notations for each SMF record type. You cannot specify subtypes on the TYPE subparameter for LSNAME. A colon indicates the range of SMF record types (bb through zz) to be recorded.

Default: None.

Value Range: 0-255 (SMF record types)

Note: When any globally specified record types are rendered without an LSNAME home, the configuration is rejected with message IFA702I stating that no log streams were specified for record type *nnn nnn*. When all record types have a home, but an LSNAME was specified without the TYPE parameter, the LSNAME parameter is accepted but ignored. No record types are written to an LSNAME without the TYPE parameter.

NOBUFFS({HALT} |{MSG})
NOBUFFS specifies the system action when the logstream specified on this LSNAME parameter buffer area is full. This option can also be specified as a global option setting. This option overrides the globally specified NOBUFFS option for the logstream specified on this LSNAME parameter only. For a detailed description of this parameter, see the parameter option description for the global specification of NOBUFFS.

Default: The option specified on the global NOBUFFS parameter

BUFUSEWARN({nn})
BUFUSEWARN specifies the overall buffer warning level percentage (nn) when SMF starts to issue warning message IFA785E for the logstream specified on this LSNAME parameter. This option can also be specified as a global option setting that applies to all logstreams defined by the

LSNAME and DEFAULTLSNAME keywords. This option overrides the globally specified BUFUSEWARN option for the logstream specified on this LSNAME parameter only. For a detailed description of this parameter, see the parameter option description for the global specification of BUFUSEWARN.

Default: The option specified on the global BUFUSEWARN parameter

DSPSIZMAX(*nnnnM* | *nG***)**
The DSPSIZMAX suboption specifies the amount of storage this logstream is to use for buffers.

Default: 2G

COMPRESS(PERMFIX(*nnnn***M))**
COMPRESS is an optional parameter. When specified with a zEDC Express feature available, SMF compresses SMF records before writing to the log stream.

Default: NOCOMPRESS.

If all zEDC Express features fail or none are available for use, message IFA730I is issued and SMF continues writing non-compressed records to the log stream.

To restart a failed zEDC session, issue a SETSMF RECORDING=LOGSTREAM to retry compression or alter other SMF parameters via the SET or the SETSMF command. Message IFA731I is issued when compression is successfully enabled.

PERMFIX(*nnnn***M)**
PERMFIX is an optional parameter when COMPRESS is specified. PERMFIX specifies the default amount of storage that SMF can keep permanently fixed for purposes of communicating with the zEDC Express feature. Storage used by the zEDC Express feature has to be page fixed; however, fixed pages are a constrained resource. Increasing this number can improve performance of SMF, but decreases the fixed storage available to the other applications. Decreasing this number can increase the fixed storage available to other applications, but may degrade SMF performance. PERMFIX can range from a minimum of 1M to a maximum of 2GB. Due to processing needs, even if this value is NOPERMFIX, SMF may use up to 2MB of fixed storage for zEDC usage.

If specified, this value overrides the global PERMFIX value.

DEFAULTLSNAME(*logstreamname***,[**
NOBUFS({HALT|MSG})],[BUFUSEWARN(nn)],[DSPSIZMAX(nnnnM |
nG)][,COMPRESS[(PERMFIX(nnnnM))]])
This optional parameter specifies the default log stream name you want to use when you write SMF records to a log stream, except for the record types specified on LSNAME parameters. When you specify DEFAULTLSNAME (and do not override it with the LSNAME parameter), records are queued and written to the specified default log stream name. For example, the SMFPRMxx parmlib member contained the following contents:

```
DEFAULTLSNAME(IFASMF.DEFAULT)
LSNAME(IFASMF.PERF,TYPE(30,89))
RECORDING(LOGSTREAM)
```

This results in record types 30 and 89 going to log stream IFASMF.PERF, while all other record types go to the default log stream IFASMF.DEFAULT.

You must follow the log stream naming restrictions specified in the LSNAME parameter.

NOBUFFS({HALT} |{MSG})
NOBUFFS specifies the system action when the logstream specified on this DEFAULTLSNAME parameter buffer area is full. This option can also be specified as a global option setting. This option overrides the globally specified NOBUFFS option for the logstream specified on this DEFAULTLSNAME parameter only. See the parameter option description for the global specification of NOBUFFS for a detailed description of this parameter.

Default: The option specified on the global NOBUFFS parameter

BUFUSEWARN({nn})
BUFUSEWARN specifies the overall buffer warning level percentage (nn) when SMF starts to issue warning message IFA785E for the logstream specified on this DEFAULTLSNAME parameter. This option can also be specified as a global option setting that applies to all logstreams defined by the LSNAME and DEFAULTLSNAME keywords. This option overrides the globally specified BUFUSEWARN option for the logstream specified on this DEFAULTLSNAME parameter only. For a detailed description of this parameter, see the parameter option description for the global specification of BUFUSEWARN.

Default: The option specified on the global BUFUSEWARN parameter

DSPSIZMAX(*nnnn*M | *n*G)
The DSPSIZMAX suboption specifies the amount of storage this logstream is to use for buffers.

Default: 2G

COMPRESS(PERMFIX(*nnnn*M))
COMPRESS is an optional parameter. When specified with a zEDC Express feature available, SMF compresses SMF records before writing to the log stream. This option allows for greater throughput of SMF records.

If a zEDC Express feature fails or none are available for use, message IFA730I is issued and SMF continues writing non-compressed records to the log stream.

To restart a failed zEDC session, issue a SETSMF RECORDING=LOGSTREAM to retry compression or alter other SMF parameters via the SET or the SETSMF command. Message IFA731I is issued when compression is successfully enabled.

PERMFIX(*nnnn*M)
PERMFIX is an optional parameter when COMPRESS is specified. PERMFIX specifies the default amount of storage that SMF can keep permanently fixed for purposes of communicating with the zEDC Express feature. Storage used by the zEDC Express feature has to be page fixed; however, fixed pages are a constrained resource. Increasing this number can improve performance of SMF, but decreases the fixed storage available to the other applications. Decreasing this number can increase the fixed storage available to other applications, but may degrade SMF performance. PERMFIX can range from a minimum of 1M to a maximum of 2GB. Due to processing needs, even if this value is NOPERMFIX, SMF may use up to 2MB of fixed storage for zEDC usage.

If specified, this value overrides the global PERMFIX value.

Note:

1. You can only specify DEFAULTLSNAME once in the SMFPRMxx member. If you specify DEFAULTLSNAME more than once, the system issues error message IFA701I and rejects the duplicate default log stream name.
2. If you do not specify a DEFAULTLSNAME in SMFPRMxx and the SYS and SUBSYS parameters specify record types (on the TYPE and NOTYPE subparameters) that are not defined on the LSNAME parameters, the system issues error message IFA702I and rejects the log stream related parameters in the parmlib member.

DSNAME {(*dataset*)}

Specifies a list of data sets to be used for SMF data set recording. If DSNAME is specified with RECORDING(LOGSTREAM), the data sets specified will be opened and prepared for use, so that they will be ready for immediate use if a SETSMF RECORDING(DATASET) is entered.

The maximum length of the data set name is 44 characters, and must follow standard MVS data set naming conventions.

You can use system symbols and the &SID symbol in SMF data set names. The resolved substitution text for the &SID system symbol is the system identifier specified on the SID parameter in SMFPRMxx. &SID can be used only to name resources in SMFPRMxx; you cannot specify &SID in other parmlib members.

Table 37 shows examples of data set names that use the &SYSNAME system symbol and the &SID symbol. The table shows the substitution texts for &SYSNAME and &SID, the data set names that specify the symbols, and the resolved texts for the data set names.

Table 37. Examples of data set Names that Use System Symbols

Substitution text		Data set name	Resolved data set name
&SID	**&SYSNAME**		
WRR1	—	SYS1.&SID;DATA	SYS1.WRR1DATA
WRR1	—	SYS1.SID&SID;.DATA	SYS1.SIDWRR1.DATA
WRR1	SP52	SYS1.&SYSNAME;&SID;.DATA	SYS1.SP52WRR1.DATA
—	SP52	SYS1.&SYSNAME	SYS1.SP52

Defaults:

- When RECORDING(DATASET) is specified, the default is SYS1.MANX and SYS1.MANY.
- When RECORDING(LOGSTREAM) is specified, no defaut.

Note: The SID parameter can be modified before initialization completes, if the PROMPT(LIST) or PROMPT(ALL) parameter in SMFPRMxx is specified. If the value of SID changes, any data set names that specify the &SID symbol will also change (because the value on the SID parameter is also the substitution text for the &SID symbol).

DSPSIZMAX(*nnnn*M |*n*G)

Specifies the maximum amount of storage that a logstream data space will consume. This parameter applies to any logstreams specified with the LSNAME or DEFAULTLSNAME keyword that does not have this keyword specified as a suboption.

This is the global specification of the DSPSIZMAX value, and its value takes effort for logstreams where DSPSIZMAX was not specified.

Value range: 128M - 2048M, or 1G - 2G (128 megabytes to 2048 megabytes, or 1 gigabyte to 2 gigabytes)

Defaults: 2G

LISTDSN| NOLISTDSN

Specifies whether the system is to generate SMF data set status messages to the operator at IPL or SET SMF time. This parameter applies to SMF data set recording only; it does not apply to SMF log stream recording. The messages contain the following information for each data set used for SMF recording:

- data set name
- data set status
  - active
  - alternate
  - close pending
  - error
  - dump required
- data set size (in number of VSAM control-interval-sized blocks)
- percentage full

Default: LISTDSN

SID { (*xxxx*) } { (*xxxx*,SYSNAME(*sysname*)) } { (*xxxx*,*ser#*[,*ser#*...]) } { (*xxxx*,COMBIN(*ser#*[,*ser#*...]))}

Specifies the system identifier that is used in all SMF records. It is also the substitution text for the &SID symbol (also known as the SID value). You can specify the &SID symbol only on the DSNAME, LSNAME, and DEFAULTLSNAME parameters in this parmlib member. Do not specify &SID in other system definitions.

You can specify the SID value directly, or you can have the system select from several SID values, using the processor serial numbers or the SYSNAME value.

For example, suppose SMFPRMxx specifies:

```
SID(AAAA,012303)
SID(BBBB,012304)
```

When running on processor 012303, the system selects an SID value of AAAA. When running on processor 012304, the system selects an SID value of BBBB.

Suppose SMFPRMxx specifies:

```
SID(AAAA,SYSNAME(PRODSYS))
SID(BBBB,SYSNAME(TESTSYS))
```

When you IPL the system, the following is specified in IEASYSxx or IEASYMxx:

```
SYSNAME(TESTSYS)
```

In this case, the system selects BBBB as the SID value, because the name TESTSYS matches the name specified in IEASYSxx or IEASYMxx.

When defining the system identifier, you can use:

- System symbols and substrings of system symbols (the resolved substitution texts for the system symbols must contain 1-4 characters).

  The system substitutes text for system symbols before it validates the syntax of the SID parameter. If errors occur in system symbol notation, the system

prompts the operator to respecify the SID parameter. See "What are system symbols?" on page 33 for information about the syntax of system symbols.

- The system name specified in IEASYSxx or IEASYMxx.
- Processor serial numbers used at IPL.
- Combinations of processor serial numbers used at IPL.

The preferred way to define the system identifier is to assign the identifier to a system symbol.

The formats for the SID are:

(*xxxx***)**
Specifies a one-to-four-character string that the system is to use as the system identifier.

(*xxxx***,SYSNAME(***sysname***))**
Selects the specified system identifier when *sysname* matches the system name specified at IPL. See "Step 3. Determine where to specify the system name" on page 44 for more information.

(*xxxx***,***ser#***[,***ser#***...])**
Selects the system identifier by the processor identifier. The serial numbers must match exactly with the ID for the processors that are currently initialized. A serial number is six digits. If the system is running on a processor that supports more than 15 LPARs (for example, a z990 processor), the serial number format is *ppnnnn*, where *pp* is the LPAR identifier and *nnnn* is the CPU serial number. On a system that is running on a pre-z990 processor, or a processor that does not support more than 15 LPARs, the serial number format is *lpnnnn*, where*l* is the logical CPU address, *p* is the LPAR identifier, and *nnnn* is the CPU serial number.

(*xxxx***,COMBIN(***ser#***[,***ser#***...]))**
Selects the system identifier by the processor identifier. If the set of currently initialized processors matches any possible combination of serial numbers specified in COMBIN, *xxxx* is used as the system identifier. A serial number is six digits. If the system is running on a processor that supports more than 15 LPARs (for example, a z990 processor), the serial number format is *ppnnnn*, where *pp* is the LPAR identifier and *nnnn* is the CPU serial number. On a system that is running on a pre-z990 processor, or a processor does not support more than 15 LPARs, the serial number format is *lpnnnn*, where*l*is the logical CPU address, *p* is the LPAR identifier, and *nnnn* is the CPU serial number.

Syntax Examples:

```
SID(SYSA)
SID(SYSB,006204,106204)
SID(SYSC,SYSNAME(SYS0001))
SID(&SYSNAME(1:4))
SID(SYSE,COMBIN(006204,106204,206204))
```

Default: If the SID parameter is not specified, and no other SID specification is available, the system uses the four-digit processor model number.

Syntax Precedence: It is possible for more than one SID specification to apply to one system. For example, if a SMFPRMxx member includes two SID specifications, SID(AAAA) and SID(BBBB,SYSNAME(SYSBBBB)) and the system is IPLed with a system name of SYSBBBB, both SID specifications apply. In this case, the precedence rules listed in Table 38 on page 753 apply.

Table 38. SID Parameter Syntax Priority List

Syntax	Meaning	Priority
SID(*xxxx*,*ser#*[,*ser#*]...)	If the serial number *ser#* matches, the SID is *xxxx*.	1
SID(*xxxx*,SYSNAME(*sysname*))	If the system name (sysname) matches, the SID is *xxxx*.	1
SID(*xxxx*,COMBIN(*ser#*[,*ser#*]...))	If any of the listed serial numbers match, the SID is *xxxx*.	1
SID(*xxxx*)	The SID is *xxxx*.	2
{Default}	SID=four-digit processor model number.	3

The lower the number, the higher the priority. For syntax with similar priority numbers, the first occurrence in the SMFPRMxx parmlib member of a matching SID specification becomes the system identifier.

Note: These precedence rules do not apply if the SID parameter is modified using the PROMPT(LIST) or PROMPT(ALL) option. If the SID parameter is modified by using the PROMPT option, the SID value changes if the modified SID specification applies, regardless of whether the previous SID syntax had a higher priority than the modified SID syntax.

Assumption: IPL the system as a multiprocessor (006204,106204) with SYSNAME=SYSSYSD.

```
If SMFPRMAA specified at IPL  SID value   Option Display (D SMF,0)
   contains:
    SID(SYSA)                   SYSA      SID(SYSA)

If SMFPRMBB specified at IPL
   contains:
    SID(SYSB,006204,106204)     SYSB      SID(SYSB,006204,106204)

If SMFPRMCC specified at IPL
   contains:
    SID(SYSC,SYSNAME(SYSSYSD))  SYSC      SID(SYSC,SYSNAME(SYSSYSD))

If SMFPRMDD specified at IPL
   contains:
    SID(&SYSNAME(4:4))          SYSD      SID(SYSD)

If SMFPRMEE specified at IPL
   contains:
    SID(SYSE,COMBIN(006204,     SYSE      SID(SYSE,006204,106204)
               106204,206204))
```

REC({ (ALL) } { (PERM)}

Specifies whether information for type 17 SMF records (scratch data set status) is to be collected for temporary data sets. PERM specifies that type 17 SMF records are to be written only for non-temporary data sets. ALL specifies that type 17 SMF records are to be written for both temporary and non-temporary data sets.

Note: A temporary data set has a system–generated data set name either from DSN=&&datasetname or from the absence of any data set name. These system generated names are in the form SYSyyddd.Thhmmss....

Default: REC (PERM)

MAXDORM (*mmss*) | NOMAXDORM

This parameter applies to SMF data set recording and SMF log stream recording. It specifies the amount of real time that SMF allows data to remain in an SMF buffer before it is written to a recording data set or a log stream,

where *mm* is real time in minutes and *ss* is seconds. NOMAXDORM specifies that the data remains in the buffer until the buffer is full. The size of the non-full buffer written out when the MAXDORM reached is one Control Interval (CI).

Value Range: 0001-5959

Default: MAXDORM (3000). This indicates 30 minutes.

EMPTYEXCPSEC{(NOSUPPRESS) | (SUPPRESS)}
Specifies whether or not you want to suppress empty execute channel program (EXCP) entries in the SMF type 30 record. IBM recommends that the SUPPRESS option is specified.

- NOSUPPRESS, which is the default, specifies that the system generates an empty SMF type 30 record EXCP section for each SMS candidate volume in the storage group that is not allocated to the DD statement.

  By default, empty EXCP sections are also generated for non-dataset allocations like DD DUMMY or spool file allocations.
- SUPPRESS specifies that the system suppresses the creation of empty EXCP sections for non-allocated candidate volumes in the SMS storage group.

  With this option, empty EXCP sections that are generated for non-dataset allocations like DD DUMMY or spool file allocations are also suppressed.

STATUS(*option*) | NOSTATUS
Specifies the time interval between creations of the type 23 SMF record (SMF status).

Default: STATUS

The *option* values are as follows:

hhmmss
Specifies the length of the time interval in *hhmmss* format, where *hh* is the hours, *mm* is the minutes, and *ss* is the seconds.

Value Range: 000001-240000

Default: (010000) — Indicates a one-hour interval.

SMF[,SYNC | NOSYNC]
Specifies that SMF is to use the global interval value (specified with the INTVAL parameter) as the time interval. Specify SYNC or NOSYNC to indicate whether or not SMF should synchronize the creation of type 23 records with the hour (using the global synchronization value specified with the SYNCVAL parameter).

Default: NOSYNC

JWT (*hhmm*)
Specifies the maximum amount of time that a job or TSO/E user address space is allowed to wait continuously, where *hh* is the amount of real time in hours and *mm* is in minutes. "Continuous wait time" is defined as time spent waiting while the application program is in control. For example, the time required to recall a data set from HSM Migration Levels 1 or 2, or the time required to mount a tape is counted towards the job's continuous wait time if the allocation of the data set is dynamic (that is, issued while the program was running), while the time required for those activities will not be counted towards the job's continuous wait time if the allocation is static (that is, for a DD statement).

Note: When specified, the SWT and TWT values will override the JWT value for started tasks and TSO/E address spaces, respectively.

If the specified time limit expires, the system passes control to the SMF time limit exit, IEFUTL (if active). IEFUTL either extends the wait time or allows the system to end the job or TSO/E user address space abnormally.

Note: If TIME=1440 is coded on the JOB or EXEC JCL statement, or if it is defaulted by the JES class attribute of TIME=1440, IEFUTL is not invoked for that job.

Value Range: 0001-2400

Default: JWT (0010) This indicates 10 minutes

SWT (*hhmm***)**
Specifies the maximum amount of time that a started task is address space is allowed to wait continuously, where *hh* is the amount of real time in hours and *mm* is in minutes. When SWT is specified, its time value will override the time value specified or defaulted to by the JWT parameter, for started tasks only. "Continuous wait time" is defined as time spent waiting while the application program is in control. For example, the time required to recall a data set from HSM Migration Levels 1 or 2, or the time required to mount a tape is counted towards the job's continuous wait time if the allocation of the data set is dynamic (that is, issued while the program was running), while the time required for those activities will not be counted towards the job's continuous wait time if the allocation is static (that is, for a DD statement).

If the specified time limit expires, the system passes control to the SMF time limit exit, IEFUTL (if active). IEFUTL either extends the wait time or allows the system to end started task address space abnormally.

Note: If TIME=1440 is coded on the JOB or EXEC JCL statement, or if it is defaulted by the JES class attribute of TIME=1440, IEFUTL is not invoked for that job.

Value Range: 0001-2400

Default: When SWT is not specified, the value specified for JWT is used for determining when to time out started task address spaces.

TWT (*hhmm***)**
Specifies the maximum amount of time that a TSO/E user address space is allowed to wait continuously, where *hh* is the amount of real time in hours and *mm* is in minutes. When TWT is specified, its time value will override the time value specified or defaulted to by the JWT parameter, for TSO/E user address spaces only. "Continuous wait time" is defined as time spent waiting while the application program is in control. For example, the time required to recall a data set from HSM Migration Levels 1 or 2, or the time required to mount a tape is counted towards the job's continuous wait time if the allocation of the data set is dynamic (that is, issued while the program was running), while the time required for those activities will not be counted towards the job's continuous wait time if the allocation is static (that is, for a DD statement).

If the specified time limit expires, the system passes control to the SMF time limit exit, IEFUTL (if active). IEFUTL either extends the wait time or allows the system to end the TSO/E user address space abnormally.

Note: If TIME=1440 is coded on the JOB or EXEC JCL statement, or if it is defaulted by the JES class attribute of TIME=1440, IEFUTL is not invoked for that job.

Value Range: 0001-2400

Default: When TWT is not specified, the value for JWT is used for determining when to time out TSO/E user address spaces.

DDCONS {(YES)} | {(NO) }
Specifies whether duplicate EXCP entries for type 30 SMF records are to be consolidated. When DDCONS(YES) is specified, SMF merges the EXCP count for these duplicate entries into one entry if the following information is the same:

- DDNAME
- Device class
- Unit type
- Channel address
- Unit address

Long-running jobs might take a long time to end in this case, because of the building of the SMF type 30 records for a long-running job.

DDCONS(NO) requests that this consolidation function be bypassed, which results in a reduction in the amount of processing required to build the records, and thus a reduction in the amount of time required to complete the job.

Default: YES

PROMPT (*option*) | NOPROMPT
Specifies whether the selected SMF parameters are to be displayed on the system console at IPL time. The system can prompt the operator to supply a reason for the IPL or to modify the parmlib parameters. The *option* values are as follows:

- IPLR specifies that the operator is to supply a reason for the IPL.
- LIST specifies that the operator is prompted for possible modifications to the SMF parameters.
- ALL specifies that the operator is prompted for the IPL reason and can modify the SMF parameters.

NOPROMPT specifies that the parameters are not listed and the operator is not prompted unless there is a syntax error in the parmlib member.

Note: SMF parameter options may not be changed using the SETSMF command when the PROMPT(IPLR) or NOPROMPT parameter options are specified. To authorize SETSMF regardless of the PROMPT or NOPROMPT parameter options, specify the AUTHSETSMF parameter.

Default: PROMPT (ALL)

AUTHSETSMF | NOAUTHSETSMF
This parameter specifies whether changes are authorized to be made to the SMF parameter options via the SETSMF command. When AUTHSETSMF is specified, the SETSMF command is authorized, regardless of the specification of PROMPT or NOPROMPT. When NOAUTHSETSMF is specified, the SETSMF command is not authorized, regardless of the specification of PROMPT or NOPROMPT.

Default: None. When not specified, by default, SETSMF is not authorized if the PROMPT(IPLR) or NOPROMPT parameter options are specified.

SYS (*options***)**
Specifies the SMF recording options and exits for the entire system. The *options* are as follows; if the same *option* is specified more than once, the system uses the first valid operator reply.

TYPE SMF record types and subtypes to be collected.

INTERVAL
Time intervals between recording.

EXITS Exits that are to receive control at various points in SMF processing.

DETAIL
The level of SMF data collection for TSO users and started tasks.

The following information describes the *options* in greater detail.

TYPE{aa,bb(cc) } NOTYPE ({aa,bb:zz }) {aa,dd(cc:yy),...}
{aa,bb(cc,...)}
TYPE specifies the SMF record types and subtypes that SMF is to collect. aa, bb, dd, and zz are the decimal notations for each SMF record type. cc and yy are the decimal notations for the SMF record subtypes. A colon indicates the range of SMF record types (bb through zz) to be recorded or the range of subtypes (cc through yy for SMF record type dd) to be recorded.You can select SMF record subtypes on all SMF record types, as well as on user records.

NOTYPE specifies that SMF is to collect all SMF record types and subtypes ***except*** those specified. aa, bb, and zz are the decimal notations for each SMF record type. cc and yy are the decimal notations for each subtype. A colon indicates the range of SMF record types (bb through zz) or the range of subtypes (cc through yy for SMF record dd) that are ***not*** to be recorded.

Value Range:

- 0-255 for SMF record types
- 0-32767 for subtypes - subtype selection applies only to SMF data set recording.

Default: TYPE (0:255) (all types and subtypes)

NOINTERVAL |INTERVAL(*suboption***)**
NOINTERVAL specifies that no interval recording takes place. INTERVAL requests interval recording and specifies the length of the recording interval. At the end of each interval, SMF generates a type 30 record. For TSO/E users, SMF can also generate a type 32 record.

Interval recording allows the user to preserve accounting data for long-running jobs or TSO/E sessions. Because SMF records accounting data for each job or task each time the interval expires, the data is not completely lost if there is a system failure.

Default: NOINTERVAL

The *suboptions* are as follows:

hhmmss
Specifies the length of the time interval in *hhmmss* format, where *hh* is the hours, *mm* is the minutes, and *ss* is the seconds.

Value Range: 000001-240000

Default: N/A

SMF[,SYNC|NOSYNC]
Specifies that SMF is to use the global interval value (specified with the INTVAL keyword) as the time interval. Specify SYNC or NOSYNC to indicate whether SMF is to synchronize the creation of type 30 and 32 records with the hour (based on the global synchronization value specified with the SYNCVAL keyword).

Default: NOSYNC

EXITS (*exit name*, *exit name*,...) | NOEXITS
EXITS specifies which SMF exits are to be invoked. A maximum of 15 exits is allowed; if an exit is not specified, it is not invoked. If this parameter is not specified, SMF behaves as if this parameter is specified with all 15 exits listed here and all SMF system exits are invoked.

NOEXITS specifies that SMF exits are not invoked.

You can specify exits on the SYS and SUBSYS statements of SMFPRMxx. Your choice of SYS or SUBSYS depends on the scope of work you want to influence (system-wide or subsystem-wide), as follows:

- On the SYS parameter, specify the exits that are to affect work throughout the system, regardless of the subsystem that processes the work.
- On the SUBSYS parameter, specify the exits that are to affect work processed by a particular SMF-defined subsystem (JES2, JES3, STC, ASCH, or TSO) and no other subsystem specific exit points will be taken.

The SUBSYS specification overrides the SYS specification. Use SUBSYS to make exceptions to your SYS specification for particular subsystems.

Some SMF exits are not called for particular subsystems. Table 39 shows which exits can be called for subsystems that are specified on the SUBSYS statement.

Table 39. Which SMF exits are called for this subsystem?

Exit Point	SUBSYS Value				
	JES2	JES3	STC	ASCH	TSO
IEFACTRT	Yes	Yes	Yes	Yes	Yes
IEFUAV	No	No	No	Yes	No
IEFUJI	Yes	Yes	Yes	Yes	Yes
IEFUJP	Yes(2)	No	Yes	No	No
IEFUJV	Yes	Yes	Yes	Yes(1)	Yes
IEFUSI	Yes	Yes	Yes	Yes	Yes
IEFUSO	Yes(2)	No	Yes	No	No
IEFUTL	Yes	Yes	Yes	Yes	Yes
IEFU29	No	No	Yes	No	No
IEFU29L	No	No	No	No	No
IEFU83	Yes	Yes	Yes	Yes	Yes
IEFU84	Yes	Yes	Yes	Yes	Yes
IEFU85	Yes	Yes	Yes	Yes	Yes

Table 39. Which SMF exits are called for this subsystem? (continued)

Exit Point	SUBSYS Value				
	JES2	JES3	STC	ASCH	TSO

Note:

1. IBM suggests that you use IEFUAV instead of IEFUJV to validate accounting information for APPC/MVS transaction programs. For more information, see *z/OS MVS Installation Exits*
2. The installation can cause this exit to be bypassed on a job class basis, through the JOBCLASS(v) initialization statement. For more information about the JOBCLASS(v) statement, see *z/OS JES2 Initialization and Tuning Reference*.

For more information, refer to "Specifying SMF Exits to the Dynamic Exits Facility" on page 767.

NODETAIL | DETAIL
Specifies the level of SMF data collection for TSO and STC; specifying DETAIL or NODETAIL has no effect on any other types of work.

For TSO, when DETAIL is specified, type 32 SMF records contain the total count of each TSO/E command used, CPU time under TCBs and SRBs, and the total number of TGETs, TPUTs, EXCPs and transactions. When NODETAIL is specified for TSO, type 32 SMF records contain only the total count of each TSO/E command used.

NODETAIL is enforced for the master address space.

For STC, specifying DETAIL has no effect. Specifying NODETAIL has an effect only if INTERVAL is also specified. Specify NODETAIL *and* INTERVAL to exclude the EXCP sections from SMF type 30 subtype 4 and subtype 5 records collected for started tasks. Otherwise, the EXCP sections for SMF type 30 subtype 4 and subtype 5 records will be included. For long running tasks, excluding EXCP sections from SMF type 30 subtype 4 and subtype 5 records can greatly reduce:

- The amount of storage required to hold SMF records in memory
- The amount of DASD space required to write the records out to data sets
- The amount of CPU used at step end and job end

Default: NODETAIL

SUBPARM (*name***(***parameter***))**
Specifies the information to be passed to a specific subsystem where:

name specifies a one to four character subsystem name. The first character must be alphabetic or national (#, @, or $), and the remaining characters can be either alphanumeric or national characters.

parameters
specifies a 1 to 60 character information SMF does not check the validity of the information string. The inner set of parentheses marks the beginning and the end of the information string.

Default: None

MEMLIMIT(NOLIMIT)
nnnnnM
nnnnnG
nnnnnT

nnnnnP
Specifies the default memlimit that will be used by jobs that do not establish a MEMLIMIT in their JCL. See *z/OS MVS JCL Reference*. MEMLIMIT is the limit on the use of virtual storage above 2 gigabytes for a single address space. NOLIMIT means that there is no limit on the use of virtual storage above 2 gigabytes.

MEMLIMIT values are defined with *nnnnn*M for megabytes, *nnnnn*G for gigabytes, *nnnnn*T for terabytes, or *nnnnn*P for petabytes. For example, to request 1275 gigabytes, specify MEMLIMIT(1275G), or to request 15 petabytes, specify MEMLIMIT(15P). The command D SMF,O displays the current MEMLIMIT.

If you want SMF MEMLIMIT to be a certain quantity of exabytes, you must convert the exabytes into petabytes and use the *nnnnn*P nomenclature. There are 1000 petabytes in an exabyte, for example, 15 exabytes would be expressed as 15000P.

Note: If MEMLIMIT is not specified in SMFPRMxx, the default value for this system default is 2G.

Default value: 2G

For a complete description of MEMLIMIT, and the ways to define it, see *z/OS MVS Programming: Extended Addressability Guide*.

SUBSYS (name,options)
Specifies the SMF recording options and exits for particular subsystems.

name represents the one to four character name of a subsystem. The first character must be alphabetic or national (#, @, or $), and the remaining characters can be either alphanumeric or national characters.

options represents the valid options for SUBSYS. The options are as follows:

TYPE SMF record types and subtypes to be collected.

INTERVAL
Time intervals between recording.

EXITS Exits that are to receive control at various points in SMF processing.

DETAIL
The level of SMF data collection for TSO users and started tasks.

These options are the same as those you can specify for the SYS parameter. When you specify SUBSYS, any option you omit from the SUBSYS parameter defaults to the value specified for that option on the SYS parameter. If you omit SUBSYS, SMF uses the values for all of the options on the corresponding SYS parameter. For the EXIT option, if you omit SUBSYS an exit point name is created for any exit specified on the SYS parameter.

Data can be recorded for up to eight subsystems in any IPL, including those specified at IPL and through subsequent SET commands. When the limit is reached, no additional subsystems can be added. The SMF-defined subsystems are JES2, JES3, STC, ASCH, and TSO. Other valid subsystems include IBM-supplied (such as OMVS), vendor-supplied, and user-defined subsystems.The system assigns work to these subsystems as follows:

- Batch jobs are assigned to the job entry subsystem (JES2 or JES3) that submitted the work to the system.
- Work started from the operator console is assigned to the STC subsystem.

- APPC/MVS transaction programs initiated by the IBM-supplied APPC/MVS transaction scheduler are assigned to the ASCH subsystem.
- Logged-on TSO/E users are assigned to the TSO subsystem.

Default: See the description of **SYS (options).**

DUMPABND { (RETRY) }
{ (NORETRY)}

Specifies whether the SMF dump program attempts to recover in the event an abend occurs. This parameter applies to SMF data set recording only; it does not apply to SMF log stream recording.

RETRY specifies that the SMF dump program attempts to recover from abends and continue processing.

NORETRY specifies that the SMF dump program terminates when an abend occurs.

Note: The SMF dump program will override this parameter and the ABEND parameter (specified on SMF dumps) if the input data set is to be dumped and cleared, and an ABEND occurs AFTER the input data set has been cleared. For this case, the SMF dump program will attempt to recover from the ABEND to prevent the output data set from being deleted and SMF data from being lost, when the SMF dump program abnormally ends. For more information about the SMF dump program, see *z/OS MVS System Management Facilities (SMF)*.

Default: RETRY

NOBUFFS { (MSG) }
{ (HALT)}

For data set recording environments, specifies the system action when the SMF address space has run out of buffer space.

- MSG specifies that the system is to issue a message and continue processing; SMF data is lost until buffer storage is again available.
- HALT specifies that the system is to enter a restartable wait state. HALT means that no SMF data is lost.

Default: MSG

For logstream recording environments, specifies the system action when any one of the defined SMF logstream buffer areas run out of space. A separate buffer area is maintained for each SMF logstream and the option specified for this parameter applies to all logstreams defined for SMF data collection. The option specified can be overridden for a particular logstream buffer if NOBUFFS is also specified as a sub-option on the LSNAME or DEFAULTLSNAME SMFPRMxx parameter options.

MSG specifies that the system is to issue message IFA786W and continue processing. Note that when the logstream buffer is full, data that is designated to be written to that logstream only will be lost until the logstream buffer storage becomes available.

HALT specifies that the system is to enter a restartable wait state when the logstream buffer is full. Specifying HALT prevents data loss when the logstream buffer is full.

Default: MSG

LASTDS { (MSG) }

{ (HALT)}
For SMF data set recording, specifies the system action when the last available SMF data set is filled and there are no more available for SMF use.

This parameter applies to SMF data set recording only; it does not apply to SMF log stream recording. If you are using SMF log stream recording, LASTDS(HALT) is ignored. When you specify data sets in SMFPRMxx, SMF issues messages that pertain to LASTDS(MSG) depending on the status of the data sets during SMF initialization.

MSG specifies that the system is to issue a message and continue processing; SMF data is buffered until an SMF data set is available. If SMF runs out of buffers, there might be a loss of data.

HALT specifies that the system is to enter a restartable wait state.

Default: MSG

MULCFUNC|NOMULCFUNC
Specifies whether users of the IFAUSAGE service that registered specifying SCOPE=FUNCTION must use IFAUSAGE with the REQUEST=FUNCTIONxxx parameters.

MULCFUNC indicates that users of the IFAUSAGE service that registered specifying SCOPE=FUNCTION must use IFAUSAGE with the REQUEST=FUNCTIONxxx parameters. SMF is to set the CVTMULFN indicator OFF.

NOMULCFUNC indicates that users of the IFAUSAGE service that registered specifying SCOPE=FUNCTION do not need to use IFAUSAGE with the REQUEST=FUNCTIONxxx parameters. SMF is to set the CVTMULFN indicator ON. Any measured usage program using SCOPE=FUNCTION, such as DB2, can record only its registration data only and omit recording the usage data.

Default: MULCFUNC

BUFSIZMAX (nnnnM)
(1G)
Specifies the maximum amount of storage that SMF can allocate for SMF record data buffering purposes. This parameter applies only when recording to SMF data sets. It does not apply when recording to SMF log streams.

BUFSIZMAX values area are defined with nnnnM for megabytes or 1G for one gigabyte. For example, to request 1 gigabyte, specify BUFSIZMAX(1G) or BUFSIZMAX(1024M). To request 128 megabytes, specify BUFSIZMAX(128M).

The BUFSIZMAX value can be specified during an IPL in parmlib member SMFPRMxx and the value can be set higher or lower using the T SMF or SETSMF command.

D SMF,O displays the current BUFSIZMAX value.

Value range: 128M–1024M, or 1G (128 megabytes to 1024 megabytes, or 1 gigabyte)

Default: 0128M (128 megabytes)

BUFUSEWARN (nn)
Specifies the overall buffer warning level percentage (nn) when SMF starts to issue a warning message depending on the recording environment. For data set recording environments, the following occurs:

- SMF maintains a single buffer area for all data recording.

- The parameter option value specifies the overall buffer warning level percentage when SMF starts to issue warning message IEE986E. When the amount of in-use buffer percentage falls below the BUFUSEWARN value minus 5 (the default is 20%), message IEE986E is deleted. When SMF is using this percentage of buffer space, message IEE986E is issued. As each additional or incremental SMF buffer (8M) is used to buffer SMF record data, SMF issues an updated instance of message IEE986E that indicates the new buffer storage percentage in use. As in-use SMF buffers are no longer needed, the buffers are removed from the in-use chain. After eligible buffers are removed, an updated instance of message IEE986E is issued that indicates the changed (reduced) buffer storage percentage in-use. When the overall SMF buffer in-use percentage drops to 5 percent below the BUFUSEWARN value, SMF performs a delete-operator-message (DOM) for message IEE986E.

For logstream recording environments, the following occurs:

- A separate buffer area is maintained for each SMF logstream. The option value specified for this parameter applies to all SMF logstreams unless it is also specified as a sub-option on the LSNAME or DEFAULTLSNAME SMFPRMxx parameter options. BUFUSEWARN options specified on LSNAME or DEFAULTLSNAME override the global specification of this BUFUSEWARN option for that logstream.
- The parameter option value specifies the overall buffer warning level percentage when SMF starts to issue warning message IFA785E. When the amount of in-use buffer percentage falls below the BUFUSEWARN value minus 5 (the default is 20%), message IFA785E is deleted. When SMF is using this percentage of buffer space, message IFA785E is issued. As each additional or incremental SMF buffer (8M) is used to buffer SMF record data, SMF issues an updated instance of message IFA785E that indicates the new buffer storage percentage in use. As in-use SMF buffers are no longer needed, the buffers are removed from the in-use chain. After eligible buffers are removed, an updated instance of message IFA785E is issued that indicates the changed (reduced) buffer storage percentage in-use. When the overall SMF buffer in-use percentage drops to 5 percent below the BUFUSEWARN value, SMF performs a delete-operator-message (DOM) for message IFA785E.

The BUFUSEWARN value can be specified during an IPL in parmlib member SMFPRMxx, and you can set the value higher or lower using the T SMF or SETSMF commands. The D SMF,O command displays the current BUFUSEWARN value.

Value range: 10–90 (10% to 90%)

Default: 25 % (25% of BUFSIZMAX value)

Example:

```
NOBUFFS(MSG)
BUFUSEWARN(25)
DEFAULTLSNAME(ifasmf.default)
LSNAME(ifasmf.important,NOBUFFS(HALT),BUFUSEWARN(30))
```

In this example, the default for all log streams is NOBUFFS(MSG) and BUFUSEWARN(25). Because the DEFAULTLSNAME did not override these defaults, they will be applied to the IFASMF.DEFAULT log stream buffer. The IFASMF.IMPORTANT log stream did override these values, and therefore NOBUFFS(HALT) and BUFUSEWARN(30) will be applied. Note that in data set recording mode, the global values still apply.

SMFDLEXIT ({ USER1(exit name,exit name, . . .)} | NOUSER1)}
{ USER2(exit name,exit name, . . .)} | NOUSER2)},
{ USER3(exit name,exit name, . . .)} | NOUSER3)}
SMFDPEXIT ({ USER1(exit name,exit name, . . .)} | NOUSER1)},
({ USER2(exit name,exit name, . . .)} | NOUSER2)},
({ USER3(exit name,exit name, . . .)} | NOUSER3)}

Specifies whether you want to specify valid exits for either the IFASMFDL program through the SMFDLEXIT keyword or the IFASMFDP program through the SMFDPEXIT keyword. You can use the USER1, USER2, or USER3 parameters on either keyword to specify the exits. If a valid exit point does not exist, use the NOUSER1, NOUSER2, or NOUSER3 parameters. For USER1, USER2, or USER3, you can specify any combination of valid exit names for both SMFDLEXIT or SMFDPEXIT.

Default:

- SMFDLEXIT(USER2(IRRADU00),USER3(IRRADU86))
- SMFDPEXIT(USER2(IRRADU00),USER3(IRRADU86))

Example:

```
SMFDLEXIT ({ USER1(exit name,exit name, . . .)} | NOUSER1 )},
          ({ USER2(exit name,exit name, . . .)} | NOUSER2 )},
          ({ USER3(exit name,exit name, . . .)} | NOUSER3 )}

SMFDPEXIT ({ USER1(exit name,exit name, . . .)} | NOUSER1 )},
          ({ USER2(exit name,exit name, . . .)} | NOUSER2 )},
          ({ USER3(exit name,exit name, . . .)} | NOUSER3 )}
```

FLOOD(ON|OFF)

Specifies whether SMF record flood support is active.

For a description of SMF flood policies, see the topic on FLOOD and FLOODPOL - Specifying SMF Record Flood Options in *z/OS MVS System Management Facilities (SMF)*.

FLOODPOL(*ffff*)

Specifies a flood policy filter (*ffff*). The FLOOD SMFPRMxx option must be set to ON for this option to become activated. Each of the options below must be present in the filter (ffff).

TYPE({aa,bb}|{aa,bb:zz}|{aa,bb:zz,...})

Specifies the records for this filter.

Value range: 0-255 (SMF record types)

Default: None.

RECTHRESH(*xxxx*)

Specifies the number of records in an interval for this filter.

Value range: 1-9999

Default: None.

INTVLTIME(*ssss*)

Specifies a flood interval time value, given in tenths of seconds. A "flood rate" is defined by the **RECTHRESH** number of records being generated within **INTVLTIM** time. Note that flood policies are decided first on how many records are generated and then on how long it takes to generate them, rather than being based only on an interval basis.

Value range: 1-9999

Default: None.

MAXHIGHINTS(*xxxx*)
Specifies the number of intervals, defined by **INTVLTIME**, that must occur at or above the flood rate before action is taken. When the **THRESHRECS** number of records are generated within the **INTVLTIME** time period (interval) for the **MAXHIGHINTS** consecutive number of intervals, SMF enters into a flood state for the designated record type.

Value range: 1-9999

Default: None.

ENDINTVL(*ssss*)
Specifies the amount of time, in tenths of a second, that must elapse before SMF determines that a flood has ended. Once a flood state has begun, if less than the **RECTHRESH** records are generated in at least ENDINTVL amount of time, the flood will end. Note that SMF only checks for ending a flood as records are generated, as opposed to checking at the specified ENDINTVL amount of time.

Value range: 1-9999

Default: None.

ACTION({MSG|DROP})
Specifies the action to be taken when a flood state is entered.

MSG Issue warning message IFA780A at the start of the flood state. Message IFA781I is issued when the flooding has stopped.

DROP Issue message IFA782A at the start of the flood state and also begin dropping records. Any attempts to write a record through the SMFEWTM or SMFWTM macro results in a return code 52. At the end of the flood state, message IFA783I is issued that indicates the number of records that have been dropped.

For more information about SMFEWTM and SMFWTM, see SMFEWTM - Writing SMF Records and SMFWTM — Writing SMF Records in *z/OS MVS System Management Facilities (SMF)*.

Default: None.

Consider the following example SMFPRMxx parmlib option settings:

```
FLOOD(ON)

FLOODPOL(TYPE(4,5),RECTHRESH(1000),INTVLTIME(50),MAXHIGHINTS(15),ENDINTVL(120),ACTION(MSG))
FLOODPOL(TYPE(102),RECTHRESH(5000),INTVLTIME(10),MAXHIGHINTS(15),ENDINTVL(100),ACTION(MSG))
FLOODPOL(TYPE(102),RECTHRESH(5000),INTVLTIME(10),MAXHIGHINTS(15),ENDINTVL(50),ACTION(DROP))
```

In this example, two filters are set up. The first filter sets up a monitor for both record types 4 and 5. This filter detects when 1000 records are generated within 5 seconds and, if records continue to be generated at this rate for more than 15 consecutive five-second intervals, message IFA780A is issued. The flood state ends, for each record type, when fewer than 1000 records are generated in at least 12 seconds. Message IFA781I is issued when the flood state ends.

The second filter for record type 102 is a two-part filter. The first part that issues the warning message, IFA780A, is triggered when 5000 records are generated in less than 1 second, and records continue to be generated at that rate for more than 15 consecutive one-second intervals. The flood state ends

when fewer than 5000 records are generated in at least 10 seconds. Message IFA781I is issued when the flood ends. If the flood state persists, the DROP filter becomes active so that if 5000 type-102 records are generated within 1 second over more than 15 consecutive one-second intervals, message IFA782A is issued and records will be dropped. Records will stop being dropped once fewer than 5000 type 102 records are generated in at least 5 seconds. Message IEFA7831 is issued when the flood state for dropping records ends.

MAXEVENTINTRECS(*nn*)
This option is used to indicate the maximum number of event driven SMF type 30 and type 89 interval records that are allowed during a regular interval cycle. Extra SMF Type 30 and Type 89 interval records can be generated when a processor capacity change occurs. The additional interval records will be generated such that when a processor changes capacity, the current interval is expired and a new, event driven interval begins. The event-driven interval will expire at the regularly scheduled end of the current interval, or when the processor changes capacity again, whichever occurs first. This serves to capture data for the interval records relevant to the current processor capacity.

The system detects a change in processor speed when an ENF 41 is signaled, along with a change in processor capacity data.

nn Specifies the maximum number of these additional event driven interval records that will be allowed during a single interval. When the number of processor capacity changes during a single interval exceeds the value specified by *nn*, no additional type 30 or 89 event driven interval records will be generated for the interval, and data for the following, regularly scheduled interval will be relevant to the most recent processor capacity change.

The default value of zero for this option will result in only regularly scheduled interval records to be generated for each interval. A value greater than zero will allow up to that many extra sets of type 30 and 89 records to be collected within one interval as the processor capacity is changed. If precise capacity measurements are needed for billing, the value of *nn* should match the number of processor capacity changes expected in a single interval.

Value Range: 0-60

Default: 0

DSPSIZMAX(*nnnn*M | nG)
Specifies the maximum amount of storage that a logstream data space will consume. This parameter applies to any logstreams specified with the LSNAME or DEFAULTLSNAME keyword which does not have this keyword specified as a suboption.

Value Range: 128 megabytes to 2048 megabytes; or 1 gigabyte to 2 gigabytes

Default: 2G

PERMFIX(*nnnn*M)
PERMFIX is an optional parameter when COMPRESS is specified. PERMFIX specifies the default amount of storage that SMF can keep permanently fixed for purposes of communicating with the zEDC Express feature. Storage used by the zEDC Express feature has to be page fixed; however, fixed pages are a constrained resource. Increasing this number can improve performance of SMF, but decreases the fixed storage available to the other applications. Decreasing this number can increase the fixed storage available to other applications, but may degrade SMF performance. PERMFIX can range from a minimum of 1M

to a maximum of 2GB. Due to processing needs, even if this value is NOPERMFIX, SMF may use up to 2MB of fixed storage for zEDC usage per log stream with a COMPRESS specification.

This is the global PERMFIX option and can be overridden by the PERMFIX suboption of the LSNAME or DEFAULTLSNAME parameters.

Default: NOPERMFIX.

SMF30COUNT | NOSMF30COUNT

Specifies whether the counter data section (SMF30CDS) should be produced in any SMF type 30 records. The counter data section contains data derived from the z/Architecture CPU Counter Facility. In order for data to be produced in the section, the Hardware Instrumentation Services (HIS) component must have enabled the appropriate counter set.

See the topic on setting up hardware event data collection in *z/OS MVS System Commands* for more information about enabling counter sets.

Default: NOSMF30COUNT.

Specifying SMF Exits to the Dynamic Exits Facility

IBM has defined the SMF exits to the dynamic exit facility. Through the PROGxx parmlib member, you can associate multiple exit routines with SMF exits, at IPL or while the system is running.

To define SMF exits to the dynamic exits facility, you must specify the exits in both PROGxx and SMFPRMxx. The system does not call SMF exits that are defined to PROGxx only. (If you do not plan to take advantage of the dynamic exits facility, you need only define SMF exits in SMFPRMxx).

Default: All exits are invoked.

Note: The PROGxx parmlib member allows you to specify installation exits and control their use. Through PROGxx, you can associate multiple exit routines with exits, at IPL or while the system is running. IBM suggests that you use PROGxx in addition to SMFPRMxx to specify exits, whether or not you want to take advantage of these functions.

The following example shows how you can specify SMF exits in a PROGxx parmlib member. If you specify the following in SMFPRMxx,

```
SYS(...EXITS(IEFU83,IEFU84,IEFUJI)...)
SUBSYS(STC,...EXITS(IEFU83,IEFU85)...)
SUBSYS(TSO,...)
SUBSYS(JES3,...EXITS(IEFUJI)...)
```

you would add the following to get the equivalent processing in PROGxx:

```
EXIT ADD EXITNAME(SYS.IEFU83) MODNAME(IEFU83)
EXIT ADD EXITNAME(SYS.IEFU84) MODNAME(IEFU84)
EXIT ADD EXITNAME(SYS.IEFUJI) MODNAME(IEFUJI)
EXIT ADD EXITNAME(SYSSTC.IEFU83) MODNAME(IEFU83)
EXIT ADD EXITNAME(SYSSTC.IEFU85) MODNAME(IEFU85)
EXIT ADD EXITNAME(SYSJES3.IEFUJI) MODNAME(IEFUJI)
```

- Because the TSO SUBSYS statement did not specify the EXITS parameter, all of the exits defined within the SYS statement are eligible to be called for the TSO subsystem.

- Because IEFU83 was defined in both the SYS statement and the STC SUBSYS statement, it needs to be specified on matching EXIT ADD statements for SYS and SYSSTC.
- Because IEFUJI was defined in both the SYS statement and the JES3 SUBSYS statement, it needs to be specified on matching EXIT ADD statements for SYS and SYSJES3.

When you associate new exit routines with SMF exits through PROGxx or the SETPROG command, you must use the following naming conventions:

- For exits listed on the EXITS keyword of the SYS statement in SMFPRMxx, each exit will have the name SYS.*yyyy* (where *yyyy* is one of the exits listed).
- For exits listed on the EXITS keyword of the SUBSYS statement of SMFPRMxx, each exit will have the name SYS*xxxx*.*yyyy* (*xxxx* is the name of the subsystem and *yyyy* is one of the exits listed).
- Where a SYS statement has been coded that does not contain an EXITS keyword, each exit will have the name SYS.*yyyy* (*yyyy* is one of the exits listed). The list of exits will be all SMF system exits.
- Where a SUBSYS statement has been coded that does not contain an EXITS keyword, each exit will have the name SYS*xxxx*.*yyyy* (*xxxx* is the name of the subsystem and *yyyy* is one of the exits listed). The list of exits will be propagated from the EXITS keyword in the SYS statement. If the EXITS keyword is **not** coded in the SYS statement as well, the list of exits will be all SMF system exits.

For information about using PROGxx to control the use of exits and exit routines, see Chapter 76, “PROGxx (authorized program list, exits, LNKLST sets and LPA),” on page 697.

Using SMFPRMxx parameters

See Using SMFPRMxx parameters in *z/OS MVS System Management Facilities (SMF)*.

Chapter 79. TSOKEY00 (TSO/VTAM time sharing parameters)

TSOKEY00 contains TSO/VTAM time sharing parameters. Starting TSO/VTAM time sharing activates the terminal control address space (TCAS). The function of TCAS is to accept TSO/VTAM logon requests and to create an address space for each TSO/E user. TCAS builds a TCAS table (TCAST) and inserts the parameter values into it. The VTAM terminal I/O coordinator (VTIOC), which is the interface between TSO/E and VTAM, uses these values to control the time sharing buffers, the maximum number of users, and other operational variables.

TSOKEY00 or an alternate member name may be specified by using the MEMBER operand or the keyword operand (overriding keyword MBR) of the operator START command, or by coding it into the PARMLIB DD statement of the cataloged procedure evoked by the START command. If a member name is not specified but a parmlib name is specified (on the PARMLIB DD statement), the system defaults to TSOKEY00. If neither a member name nor a parmlib name is specified, default values in the TCAS program are used.

If, during TCAS initialization, an I/O error occurs after some of the values have been read from parmlib, the default values in the TCAS program supply the remaining parameter values for the TCAS table.

If the specified value for a parameter does not fall within the value range, the default value is used.

Parameter in IEASYSxx (or supplied by the operator)

None

Syntax rules for TSOKEY00

The following rules apply to the creation of the TSO/VTAM time sharing parmlib member:

- For each record, columns 1 through 71 are valid for data. Columns 72 through 80 are ignored.
- Data may be continued from one record to another by ending a record with a comma and blank in contiguous data columns, then inserting data in any column of the data area on the next record.
- A parameter must be complete in a record. It may not cross record boundaries. It may not be repeated.
- A comma must separate adjacent keywords.
- Invalid or misspelled parameters are ignored. Default values are substituted, and are listed on the device specified by the PRINTOUT DD statement of the procedure used to start TSO/VTAM time sharing, or on the device specified by the device name operand of the operator START command.

IBM-supplied default for TSOKEY00

The default values are internal constants in the TCAS program.

```
USERMAX=40,RECONLIM=3,BUFRSIZE=132,HIBFREXT=48000,
LOBFREXT=24000,CHNLEN=4,SCRSIZE=480,ENGTRANS=BASE,
ACBPW=password,    (optional - no default)
MODE=NOBREAK,MODESW=NO,
RCFBDUMP=xxyyz,    (optional - no default)
CONFTXT=YES,
GNAME=gentso,      (optional - no default)
BASENAME=TSO,,CODEPAGE=NO
```

Note: All parameters in the TSOKEY00 parmlib member must precede any comments; otherwise, the parameters will be ignored. For example:

```
USERMAX=100,
RECONLIM=3,
BUFRSIZE=132,
HIBFREXT=6600,
LOBFREXT=3300,
CHNLEN=4,
SCRSIZE=1920
ENGTRANS=BASE
/* CHANGE LOG
/*
/* DATE     WHO   WHAT
/* 12/04/09 MMB    INCREASED USERMAX FROM 70 TO 100
/* 11/02/09 MMB    INCREASED USERMAX FROM 50 TO 70
/* 09/26/11 KIB    ADD ENGTRANS VALUE
/*
```

Statements/parameters for TSOKEY00

The TSO/VTAM time sharing parameters are as follows:

USERMAX
Specifies the maximum number of users that may be logged on to the TSO/VTAM time sharing system at one time. (Note that because VTAM considers each user to be an application program, and because there must be an APPL definition statement that defines each application program to VTAM during VTAM network definition, this value may not exceed the number of APPL definition statements.)

Value Range: 0-maximum number of address spaces in the system

Default: 40

RECONLIM
Specifies the time limit in minutes within which a user may reconnect after his line has been disconnected (that is, the amount of time an address space remains active in the event of terminal disconnection). Note that a setting of RECONLIM=0 means that there is a zero wait for reconnection, which means that a reconnect is not possible.

Value Range: 0-32767

Default: 3

BUFRSIZE
Specifies the size in bytes of a VTIOC buffer. Input and output data smaller than the BUFRSIZE value uses cells equal to the BUFRSIZE value. Input and output data larger than the BUFRSIZE value is assigned to dynamically allocated buffers (allocated by the GETMAIN macro) equal to the data size.

Value Range: 4-3016

Default: 132

HIBFREXT
Specifies the maximum amount (in bytes) of virtual storage that can be dynamically allocated for output data. When this value is reached, no more output requests are honored until LOBFREXT is reached. (Input requests are rejected only when no virtual storage is available in the address space.)

Value Range: Maximum size TPUT used at the installation-maximum amount of available virtual storage. Minimum value of 48000 is forced to minimize the output wait due to an out of storage condition. For details about TPUT, see *z/OS TSO/E Programming Services*.

Default: 48000

LOBFREXT
Specifies the minimum number of virtual storage bytes that can be dynamically allocated for output data. When this value is reached, tasks that are suspended for lack of output buffers are marked as dispatchable.

Value Range: 0-(HIBFREXT-1)

Default: 24000

CHNLEN
Specifies for output the number of request units (RUs) in each RU chain for IBM 3767 and IBM 3770 terminals. This value determines the maximum size of a chain and, therefore, the maximum amount of data that can be sent to the terminal in one chain (CHNLEN x 256=data transmitted). This value also determines the maximum amount of data purged when the CANCEL key is pressed to stop printing of outbound (host-to-terminal) data.

The CHNLEN value is associated with the values specified on the PACING operand of the LU macro during network control program definition, and on the VPACING operand of the LU or PU macro during VTAM definition. Use the CHNLEN default value (4) if the PACING and VPACING default values are used. Otherwise, calculate CHNLEN by using the formula (2*pn*-2*pm*)+(2*vn*-2*vm*), where *pn* and *pm* are the PACING *n* and *m* values, and *vn* and *vm* are the VPACING *n* and *m* values. If more than one network control program is used with TSO/VTAM time sharing, the smallest value found when calculating the CHNLEN value should be used. Otherwise, the network control program could enter slowdown mode.

Value Range: 1-10

Default: 4

SCRSIZE
Specifies the default screen size of IBM 3270 systems network architecture (SNA) terminals. The valid values for this parameter are:

Screen Size	Value
12 x 40	480
24 x 80	1920
32 x 80	2560
43 x 80	3440
27 x 132	3564
62 x 160	9920

(Use the FEATUR2 operand of the LOCAL or TERMINAL macros to specify non-SNA IBM 3270 screen sizes.) This value affects only those terminals whose sizes were not specified by using the PSERVIC parameter of the MODEENT macro during VTAM definition.

If an installation has IBM 3270 SNA terminals of only one screen size, SCRSIZE should specify that size. If the installation has IBM 3270 SNA terminals of both screen sizes, SCRSIZE may be used to define the more common size, and the PSERVIC parameter may be used to define the less common size.

Value Range: 480-9920

Default: 480

ENGTRANS

Specifies the type of English language translation that is to be performed when the TPUT EDIT macro is used to write to terminals that support the coded graphic set global identifier (CGCSGID) X'02B90025'. (For details about TPUT, see *z/OS TSO/E Programming Services*.) You can specify one of the following values:

BASE Base English translation

EXTENDED
Extended English translation, as described in *3174 Character Set Reference*, GA27-3831.

NONE
No translation is performed

Value Range: BASE, EXTENDED, or NONE

Default: BASE

ACBPW

Specifies the optional password associated with all TSO/VTAM ACBs. If a password is specified on the PRTCT parameter of VTAM's APPL definition statement, ACBPW must be specified.

To prevent unauthorized disclosure of the ACB password when the CBPW keyword is specified, take the following steps:

- Place the TSOKEY00 member in a password protected data set that is compatible with SYS1.PARMLIB (for example SYS1.VTAMLST).
- Modify the parmlib DD statement in the TSO/VTAM start procedure to refer to the password protected data set, as show in the following example:

```
//PARMLIB DD DSN=SYS1.VTAMLST(&MBR),DISP=SHR,FREE=CLOSE
```

Value Range: Any 1 to 8 EBCDIC characters

Default: None

MODE

Specifies how 3276 and 3278 terminals are to be supported for TSO/VTAM. BREAK indicates that the terminal keyboard is unlocked whenever possible and does not necessarily correspond to the issuance of a TGET. NOBREAK indicates that the terminal keyboard is locked except when a TGET is issued. The BREAKIN option of TPUT applies only when BREAK is specified.

Value Range: BREAK or NOBREAK

Default: NOBREAK

MODESW
Specifies whether or not the TERMINAL command or the STBREAK macro can be used to change the mode of operation specified with the MODE parameter.

Value Range: YES or NO

Default: NO

RCFBDUMP
Specifies the values of the return code (xx) and the feedback field (yy) for a RPL-type request. When an abnormal termination occurs that produces a return code and a feedback value that matches those specified in RCFBDUMP, a dump is taken to the extent specified in z. The specification z=0, specifies that only the local address space is to be dumped; z=1 specifies a full storage dump. The RCFBDUMP parameter can be used to request dumps for I/O errors from which TSO/VTAM can recover.

Value Range: See return code and feedback values in *z/OS Communications Server: SNA Programming*.

Default: None

CONFTXT
Specifies whether the buffer output is to be confidential. YES indicates that the buffer output is to be confidential and, as such, not traceable; the data in the VTAM buffers is overwritten with zeros immediately after it is sent to the terminal.

Value Range: YES or NO

Default: YES

GNAME
Specifies a generic name by which one or more TSO/VTAM systems in a sysplex may be referenced. A generic name can also be specified on the START command to activate TSO/VTAM. If a generic name is not specified in TSOKEY00, or on the START command, then no generic name is used. If a generic name is used, TSO/VTAM registers its generic name by issuing the SETLOGON GNAMEADD macro instruction.

For example, three TSO/VTAM systems in a sysplex could have network names of TSO1, TSO2, and TSO3 respectively, and each defined with GNAME=TSOGR. A TSO user could logon to the name “TSOGR” and be transparently assigned to any one of the 3 TSO systems as determined by load balancing.

Value Range: Any 1 to 8 EBCDIC characters that must be unique from any real resource name in the network, and may not be “TSO”.

Default: None

BASENAME
Specifies a 1 to 4 character prefix to be used for the ACBNAME operand in the VTAM application program major node. As an example, if you specify BASENAME=tttt, you would use the names tttt0001, tttt0002, tttt0003, and so forth. See the *z/OS Communications Server: SNA Network Implementation Guide* for an example of how to use BASENAME.

Value Range: Any 1 to 4 EBCDIC characters, following the VTAM naming requirement specified in the *z/OS V2R1.0 Communications Server: SNA Resource Definition Reference*.

Default: TSO

CODEPAGE

Specifies to obtain the code page (CGCSGID) information for a TSO session from the terminal or the emulator. YES indicates that TSO/VTAM queries the terminal or the emulator to obtain the code page information.

Value Range: YES or NO

Default: NO

Chapter 80. VATLSTxx (volume attribute list)

VATLSTxx contains one or more volume attribute lists that predefine the mount and use attributes of direct access volumes. The *mount* attribute determines the conditions under which a volume can be demounted. The *use* attribute controls the type of requests for which a volume can be allocated.

The system programmer can predefine volume "mount" attributes as permanently resident or reserved, and can predefine volume "use" attributes as storage, public, or private. Therefore, critical direct access volumes can be controlled because the "mount" and "use" attributes determine the type of data sets that can be placed on a volume. During allocation, data sets on volumes marked permanently resident or reserved are selected first because they require no serialization, thus minimizing processing time.

You can ensure a faster initialization by specifying the volume attribute list(s) efficiently. Do not, for example, specify a list at a given IPL that contains entries for volumes that will not be mounted. Un-mounted volumes require operator intervention with resultant delay.

There are two ways to define the use and mount attributes for DASD volumes. You can define them in individual entries in the VATLSTxx member, one volume to each entry, or you can use one entry to define a group of volumes. In this second way, you specify generic volume serial numbers and device types for groups of volumes.

Specifying generic values makes it easier for you to maintain your VATLSTxx members. Give one generic name to a group of volumes mounted on devices that have the same device type and the same mount and use attributes. Specify an asterisk ("*") for the device type when groups of volumes have the same volume serial numbers, mount attributes, and use attributes, regardless of the devices the volumes are mounted on. See "Specifying a generic volume serial number" on page 780 and "Specifying a generic device type" on page 781 for more information.

For JES3, DEVICE and SETRES initialization statements, rather than the VATLST, can be used to specify permanently-resident volumes. In a JES3 complex, you must assign the same mount attribute to a volume mounted on every system in the complex. For example, if you use the mount attribute of "permanently resident" for the system residence volume on one system, you must use the same use attribute on all the other systems.

If VATLST members are used for volumes resident on devices that are shared among multiple systems, the mount attributes for a specific volume must be the same on all systems.

Definitions of the mount and use attributes

There are three mount attributes: permanently resident, reserved, and removable. The permanently resident or reserved attributes may be specified in a volume attribute list. The removable attribute automatically applies to any volume that VATLSTxx does not designate or default as permanently resident or reserved.

A *permanently resident* volume is either one that cannot be physically demounted (that is, a drum, 3344, or 3350) or one that cannot be demounted until its device is varied offline. Only direct access volumes can be made permanently resident.

The following volumes are *always* marked permanently resident by NIP. You should therefore specify only the use attribute of these volumes in a volume attribute list:

- Volumes that cannot be physically demounted (such as a 3350 volume).
- The system residence volume This volume includes the SYS1.SVCLIB and SYS1.NUCLEUS data sets.
- Volumes that contain these system data sets: SYS1.LINKLIB and data sets concatenated to it, SYS1.DUMPnn, VIO journaling data set, page data sets, and swap data sets.

A *reserved* volume remains mounted until the operator issues an UNLOAD or a VARY OFFLINE command. A volume is marked reserved when it is so designated in a volume attribute list, or when the operator issues a MOUNT command for the volume.

A *removable* volume can be demounted after its last use in a job, or when the device on which it is mounted is needed for another volume. Any volume not designated as either permanently resident or reserved is considered removable. The operator can change a removable volume to a reserved volume by issuing the MOUNT command for the volume.

The *use* attribute controls the type of request for which a volume can be assigned:

- a specific volume request.
- a temporary, non-private non-specific volume request.
- a non-temporary, non-private, non-specific volume request.

Three *use* attributes are used for allocating these types of volume requests, as follows:

- A *private* volume is allocated only to a specific volume request. For more information about this attribute, see *z/OS MVS JCL Reference*.
- A *public* volume is allocated to a temporary, non-specific volume request (or possibly to a specific volume request). Thus, a scratch data set would be placed on a public volume.
- A *storage* volume is allocated primarily to a non-temporary, non-specific volume request. (A storage volume can also be allocated to a specific volume request or a temporary non-specific volume request.)

Note: A storage volume is required by the SAVE subcommand of EDIT for a newly created data set. If a *storage* volume is not available, the SAVE subcommand cannot save the data set.

Processing the VATLSTxx members

The system reads the VATLSTxx members that were specified in the VAL parameter in IEASYSxx. If the system detects an incorrect VATLSTxx entry, the system issues informational message IEA855I, and continues processing the remaining entries.

If an I/O error occurs during the reading, the operator can choose to display informational message IEA850I, which lists the volumes, device types, and

attributes that will be processed. A second message (IEA853A) allows the operator to choose one of the following recovery options:

- Continue processing any remaining lists.
- Stop the processing of remaining lists.
- Specify a new VATLSTxx member by replying r 0,xx, where xx is the two-character identifier for VATLSTxx.
- If necessary, reIPL the system.

If an I/O error does not occur during the reading, the system lists the IPL and SYSTEM use attributes in message IEA168I.

Volume attribute processing compiles a list of all VATLSTxx entries. If a particular volume serial number appears on more than one entry, the system uses the volume attributes specified in the last entry for that volume serial.

Mount messages can be issued for unmounted volumes, up to the maximum number of processed entries. The system can process up to 64000 unique VATLSTxx entries at IPL.

The operator can respond to the mount messages by replying with the device number of the requested device type (that is, 3330-1). If the operator chooses not to mount a volume, the operator replies 'U' or ENTER.

If the operator enters an incorrect device number, or a path to a device is not available, the operator can reenter new device numbers.

To indicate that no more volumes will be mounted, enter 'U' or ENTER.

When message IEA860A lists the devices that need volumes, the operator should mount the required volumes on the replied devices. When all devices have become ready (green lights on), the operator replies 'U' or ENTER to message IEA860A. Volume attribute processing then scans for mounted volumes.

If a volume that did not appear in the mount message is mounted on a unit specified by the operator, it is unloaded. The volume is also unloaded if the operator mounts the requested volume on a device type other than the one specified in the volume attribute list, or on an un-requested unit.

If all the required devices do not become ready, volume attribute processing issues message IEA893A, which lists the devices that are not ready. If the operator intends to ready these devices, the operator may do so before replying 'U' or ENTER to the message. If a volume cannot be mounted on a device for some reason (such as a hardware problem), the operator should reply NO to the message, after all other required devices have been processed. This response indicates to volume attribute processing that no more volumes will be mounted.

If the system does not find a volume that matches a generic volume serial number entry, it issues message IEA166I.

Parameter in IEASYSxx (or supplied by the operator)

```
VAL={aa         }
-----------------
    {(aa,bb,...)}
```

Two alphanumeric characters (such as, A1 or 30) are appended to VATLST to specify the VATLSTxx member(s) of parmlib. If the parameter is not specified either in IEASYSxx or by the operator, the default member VATLST00 is used, if it exists. If the VAL parameter specifies multiple members, the members are processed in the order specified. If a particular volume serial number appears on more than one entry, the volume attributes specified in the last entry for that volume serial will be accepted. If the VAL parameter has an invalid format, or if it specifies a member that does not exist in parmlib, the operator is prompted to respecify the member or to reply 'U' to cause the member to be ignored.

Support for system symbols

You can specify system symbols in VATLSTxx. However, be aware that the fields in VATLSTxx are column-dependent (except for the VATDEF statement). The resolved substitution texts for the system symbols must conform to the rules described in "Statements/parameters for VATLSTxx" on page 782.

The following rules apply when you use system symbols in VATLSTxx:

- If a system symbol is used anywhere in the volume serial number, there must be a blank after the entire volume serial number. This blank delimits the end of the volume serial number, not the end of the system symbol, and does not become part of the volume serial number when resolved.
- A period at the end of the system symbol delimits the end of that symbol. The period is optional for the following cases:
  - between 2 system symbols. In this case, the & that begins the second system symbol acts as a delimiter.
  - before the blank delimiter at the end of the entire volume serial number.
  - before a generic character, such as % or *.
- The volume serial number field in the VATLSTxx entry must be exactly 6 characters long. If a combination of system symbols resolves to less than 6 characters, you must add blanks to pad the volume serial number.

Let us assume we have an IEASYMxx member with the following definitions:

- SYMDEF(&SYMB1.='PA')
- SYMDEF(&SYMB2.='GE')
- SYMDEF(&SYMB3.='%%')

With these assumptions, each of the examples that follow will resolve to a volume serial number of PAGE%%:

- This example leaves out periods before &.

  ```
  &SYMB1.&SYMB2.&SYMB3. ,0,0,3390...
  &SYMB1&SYMB2&SYMB3. ,0,0,3390...
  ```

- This example leaves out periods before %.

  ```
  &SYMB1&SYMB2.%% ,0,0,3390...
  &SYMB1&SYMB2%% ,0,0,3390...
  ```

- In this example, the blank that delimits the end of the volume serial number also delimits the end of the system symbol.

```
PAGE&SYMB3. ,0,0,3390...
PAGE&SYMB3 ,0,0,3390...
```

Using the same assumptions for the IEASYMxx member, the following VATLSTxx entry would be invalid:

```
&SYMB1GE%% ,0,0,3390...
```

Instead of appending "GE%%" to the end of &SYMB1, the system would look for a system variable &SYMB1GE, which does not exist in our example.

For information about how to use system symbols in shared parmlib members that require unique values, see Chapter 2, “Sharing parmlib definitions,” on page 33.

Creating a VATLSTxx member

The VATLSTxx member contains an optional VATDEF statement, which specifies a default use attribute, followed by entries that define the mount and use attributes of DASD volumes.

Use the VATDEF statement in VATLSTxx to specify a default use attribute. If you do not specify VATDEF, the system assigns a default of *public* to those volumes that are not specifically assigned a use attribute in the VATLSTxx member or in a MOUNT command. VATDEF must be the first entry in the first VATLSTxx member you specify in the VAL= operand; the system ignores any later specifications of VATDEF that might appear in other VATLSTxx members.

The following is the syntax for the VATDEF statement. Unlike DASD volume entries in VATLSTxx members, the VATDEF statement is not column-dependent but it also cannot be continued on another line.

```
VATDEF
                {PUBLIC }
          IPLUSE( {PRIVATE} )
                {STORAGE}

                {PUBLIC }
          SYSUSE( {PRIVATE} )
                {STORAGE}
```

VATDEF
The default for use attributes for DASD volumes. If you specify VATDEF without any operands, the system uses the operands IPLUSE(PUBLIC) and SYSUSE(PUBLIC).

IPLUSE
The use attribute default that the system applies to permanently resident volumes that are brought online during IPL (that is, have no VATLST entry, or whose use attribute is not specified correctly in VATLSTxx).

SYSUSE
The use attribute default that the system applies to volumes that are varied online after IPL and have no entry in a VATLSTxx member.

PUBLIC
Sets the default use attribute to *public*. This operand is the default for VATDEF IPLUSE and VATDEF SYSUSE.

PRIVATE
Sets the default use attribute to *private*.

STORAGE
Sets the default use attribute to *storage*.

Example of default use attributes

At an installation where the VATLSTxx members are defined as VAL=(00,05), member VATLST00 contains the following entries:

```
VATDEF SYSUSE(PRIVATE)
30565A,0,2,3330    ,    TSO
```

The system ignores any VATDEF entry in VATLST05. It uses PRIVATE as the default use attribute for any volume varied online after IPL and PUBLIC for any volume brought online during IPL. The optional information TSO appears in installation printouts, but is ignored by VATLST processing.

Syntax rules for VATLSTxx

The following rules apply to the creation of a DASD volume entry in the VATLSTxx member:

- Each record consists of 80 columns, although columns 22 through 80 are ignored.
- The fields are column-dependent, as shown in "Statements/parameters for VATLSTxx" on page 782 (except for the VATDEF statement).
- There are only two required fields: the volume serial number and the device types. All other fields have defaults.
- Use a comma to separate adjacent fields, except before the optional information field.
- Specify all characters in EBCDIC.
- Lines that begin with an asterisk in column 1 are comments.

"Statements/parameters for VATLSTxx" on page 782 includes a graphic description of all fields in the record and their lengths.

The following sections describe how to specify DASD volume entries.

Specifying a generic volume serial number

Generic volume serial numbers allow you to reduce the number of entries in the VATLSTxx members. The valid generic characters are % and *; you can use any combination of these characters in an entry. If a volume serial number does not contain any generic characters, it defines only the volume with the specified volume serial number. Rules for generic volume serial numbers follow, with examples of the generic numbers and the possible matching numbers.

Rules for Generic Volume Serial Numbers: The character "*" in the generic volume serial number indicates that any character in that position and all subsequent characters are a match.

Generic Volume Serial Number	Matching Volume Serial Numbers
TSO*	TSO01, TSO, and TSO123
TSO	XABTSO, ABTSO1, and TSO01
P*5	PROD05 and P5

Generic Volume Serial Number	Matching Volume Serial Numbers
*PROD	12PROD and PROD

The character "%" in the generic volume serial number indicates that any single character within the volume serial number is a match to that character position.

Generic Volume Serial Number	Matching Volume Serial Numbers
TSO%	TSO1
%TSO%	ATSO1
P%%5	P125
%PROD	1PROD

Both symbols can appear in the same number.

You can enter the generic volume serial numbers in any order. However, if more than one generic specification can be applied to a particular volume serial, the system will find the attributes specified in the last entry with such a generic specification.

For example, an installation has the volume serial numbers WK982A and WK9BBB, and the VATLSTxx has the following entries.

```
WK982%,1,0,3350     ,N   STORAGE VOLUME
WK9%%%,1,1,3350     ,N   PUBLIC  VOLUME
```

The system will find the second entry. WK9%%% matches both volume serial number WK982A and WK9BBB.

Generic Volume Serial Number	Matching Volume Serial Numbers
*TSO%	TSO1 and XXTSO2
P%5*	P05PRO

Specifying a generic device type

The character "*" identifies a generic device type, based on which DASD are defined at your installation. Use the generic device type when you have the same use attribute and the same mount attributes regardless of the device you want the volumes mounted on.

If you specify a generic volume serial number, you can specify a generic device type. For example, if you specify the generic volume serial as TSO*, the matching volume serial numbers for this are TSO01, TSO, and TSO123, as above. TSO01 is on a 3350, TSO is on a 3380, and TSO123 is on a 3390. If you want to specify the same mount and use attributes for these volumes, regardless of the device types, you can specify a generic device type and then the mount and use attributes.

Example of setting the generic device type

The following VATLSTxx entry:

```
TSO*  ,0,2,*
```

Is equivalent to the following three entries:

```
TSO01 ,0,2,3350
TSO   ,0,2,3380
TSO123,0,2,3390
```

These examples assume that only 3350, 3380, and 3390 DASD are defined.

Note:

1. Generic device types do not support specifications such as 33%% or 33*
2. A VATLSTxx statement with a specific volume cannot have a generic device type.

Statements/parameters for VATLSTxx

Figure 45 shows the entries in VATLSTxx members that define the mount and use attributes of direct access volumes. The fields are column-dependent; they are described in the section that follows.

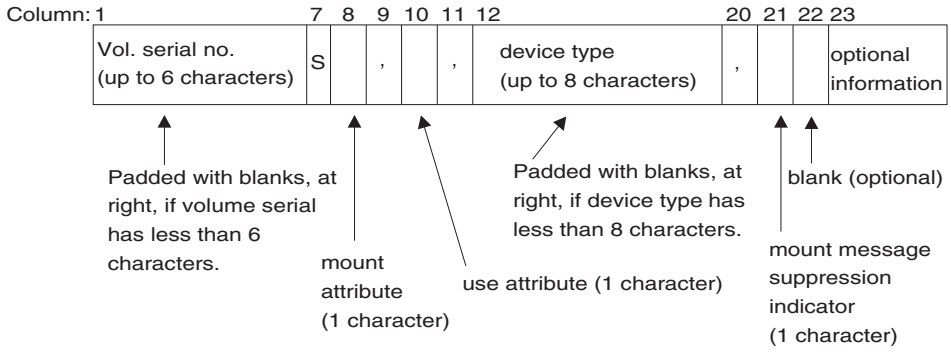

Figure 45. Entries n VATLSTxx members that define attributes of DASD

Examples of VATLSTxx entries

Figure 46 shows the entries in VATLSTxx that define the mount and use attributes. Table 40 on page 783 describes the fields in more detail.

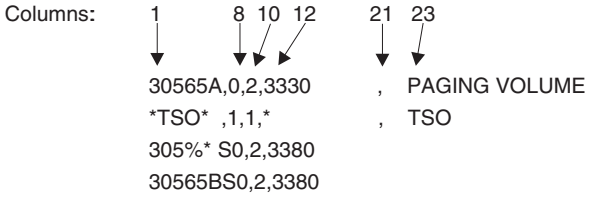

Figure 46. Entries in VATLSTxx members that define mount and use attributes of DASD

In the first entry in Figure 46, a volume whose serial number is 30565A is to be mounted on a 3330 and marked permanently resident. The volume's *use* attribute is to be *private*. A mount message is to be issued if the volume is demounted. The optional information, PAGING VOLUME, appears in installation printouts but does not affect VATLST processing.

In the second entry in the example, volumes with serial numbers that include the character string “TSO” are to be mounted on any DASD and reserved. The volumes' use attribute is to be *public*. Because you specified a generic volume serial

number, the system does not issue a mount message for un-mounted volumes. The optional information, TSO, appears in installation printouts but does not affect VATLST processing.

In the third entry, volumes with serial numbers included in the 305%* specification, are to be mounted on a 3380 and marked permanently resident. The use attribute is to be *private*.

In the fourth entry, a specific volser name, 30565B, contains an "S" in column 7, indicating that the volser is specific. However, error message IEA855I will be issued because the volser does not contain either of the special characters, "%" or "*". The "S" should only be coded for specific volsers containing special characters.

Table 40. Explanation of VATLSTxx entries

Parameter	Column	Meaning and Use	Value Range	Default Value
volume serial	1-6	Specifies either the direct access volume with *mount* and *use* attributes to be set, or a generic volume serial number. The volume serial number must begin at the first character position in the record.	1 to 6 alphameric characters plus special characters % and *, left justified in the field and padded with blanks at right to occupy six columns.	None
specific vol. ser. identifier	7	"S" specifies that the entry is a specific volume serial number, not a generic one. Code the "S" only when your volume serial number has an asterisk or percent sign in it and you do not want the system to process it as a generic entry. Code a comma when the volser is a specific volser without the special characters or the volser is a generic volser.	S or ,	None
mount attribute	8	Specifies whether the volume is to be permanently resident or reserved. Default is assigned if any character other than 1 is specified. **0** specifies permanently resident. **1** specifies reserved.	0 or 1	0
use attribute	10	Specifies whether the volume is to be defined as storage, public, or private. The default is assigned if any character other than 0, 1, or 2 is specified. **0** specifies storage. **1** specifies public. **2** specifies private.	0-2	1, unless the VATDEF operand specifies a different default.

Table 40. Explanation of VATLSTxx entries (continued)

Parameter	Column	Meaning and Use	Value Range	Default Value
device type	12-19	Specifies the device type, such as 3380. Up to eight characters may be specified, but the first character must start at column 12. Only supported device types that were defined through HCD are acceptable. This parameter indicates the basic device but does not denote special features, such as track overflow. An asterisk "*" in column 12 indicates a generic device type.	Up to 8 characters, left justified within the field, and padded with blanks at the right to occupy eight columns.	None
mount message suppression	21	Specifies whether mount messages should be issued for the volume if it is not already mounted. This parameter is ignored for volumes that are defined with generic VATLST entries. **N** specifies that mount messages should be suppressed. **x** where x, which is a blank or any character except N, specifies that mount messages should be issued.	N, blank, or any character.	Blank, indicating that mount messages should be issued.
optional information	23-80	Contains optional, installation-defined information (for example, an eye-catcher or brief comment) to be included in installation printouts. The system does not use this information.	Not applicable.	None

IBM-supplied default for VATLSTxx

No default member is supplied by IBM. The installation can, however, create its own default member, named VATLST00.

Chapter 81. XCFPOLxx (XCF PR/SM policy)

In a multisystem sysplex on PR/SM, the XCF PR/SM policy provides a way for the installation to obtain high availability for multisystem applications on the MVS systems in the sysplex. You specify the XCF PR/SM policy in a parmlib member. IBM suggests naming this member XCFPOLxx, where xx is a unique 2-character identifier. You activate (or deactivate) the policy with the SETXCF PRSMPOLICY command, and display the name of the active policy with the DISPLAY XCF,PRSMPOLICY command.

The XCF PR/SM policy allows you to specify the actions that XCF on a system in the sysplex on PR/SM is to take when a "status update missing" or "system gone" occurs on another system in the sysplex.

Some reasons a status update missing condition might occur are:

- The system has failed and entered a non-restartable wait state.
- The system is in a disabled spin loop.
- The operator stopped or reset the system.
- The system has failed and cannot dispatch normal work.
- The system entered a restartable wait state but the operator has not restarted the system.

A system gone condition occurs after a system reset is confirmed.

The XCF PR/SM policy reduces the need for operator intervention because XCF on a remaining system in the sysplex on PR/SM can automatically reset the failing system, deactivate the LPAR, and reconfigure processor storage for use by the remaining system.

Note: The XCF PR/SM policy in an XCFPOLxx parmlib member will not work unless the installation enables the system to do a reset of an MVS system on an LPAR and deactivate the LPAR on the PR/SM processor. See the *PR/SM Planning Guide* for information about the assignment of LPARs and processor storage.

For more information about using XCF, see *z/OS MVS Setting Up a Sysplex*.

Parameter in IEASYSxx (or supplied by the operator)

None.

Syntax rules for XCFPOLxx

The following syntax rules apply to XCFPOLxx:

- Use columns 1 through 71. Do not use columns 72 - 80 for data; these columns are ignored.
- At least one delimiter (space or comma) is required between a statement and keyword. Delimiters are not required between keywords.
- Use at least one delimiter to separate multiple keyword values within parentheses.

- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".

Syntax format of XCFPOLxx

```
NOSTATUS(failsys)
                { RESETTIME(nnnnn) }
                { DEACTTIME(nnnnn) }

SYSGONE(failsys)
                  SYSTEM(sysname)
                  DEACTIVATE(othersys|ALL)
                [ STORE(YES|NO)  ]
                [ ESTORE(YES|NO) ]
```

IBM-supplied default for XCFPOLxx

None.

Statements/parameters for XCFPOLxx

NOSTATUS(failsys)

The NOSTATUS statement specifies an action to take when the specified system appears to have failed (status update missing). You can choose to either reset or deactivate the failed system. The current system and the failing system must be running in LPAR mode on the same processor. If the processor is physically partitioned, then both LPARs must be on the same side.

RESETTIME(nnnnn)

Specifies that the failing system is to be reset after the specified number of seconds have elapsed. If you specify RESETTIME(10), this system will perform a system reset of the failing system 10 seconds after the status update missing condition is detected for the failing system.

If the failing system becomes active before the specified time interval elapses, the system reset is not performed.

Note: IBM suggests that you specify a time interval of less than 60 seconds, because longer intervals might cause SETXCF PRSMPOLICY commands to be delayed.

Value Range: 0 - 86400 seconds

DEACTTIME(nnnnn)

Specifies that the logical partition (LPAR) where the failing system resides is to be deactivated after the specified number of seconds have elapsed. If you specify DEACTTIME(15), this system will deactivate the LPAR of the failing system 15 seconds after the status update missing condition is detected for the failing system. Specifying DEACTTIME(nnnnn) also causes the system to be reset.

If the failing system becomes active before the specified time interval elapses, the LPAR is not deactivated.

Note: IBM suggests that you specify a time interval of less than 60 seconds, because longer intervals might cause SETXCF PRSMPOLICY commands to be delayed.

Value Range: 0 - 86400 seconds

SYSGONE(failsys)
Specifies an action to take when the specified system has been reset by the operator or by the policy specified by the NOSTATUS parameter.

SYSTEM(sysname)
Identifies the system in the sysplex that is to take the following actions when the specified system (failsys on the SYSGONE statement) failed. The system indicated by *sysname* is not necessarily the current system, instead it is the system you are designating to perform the deactivation of the LPAR. This allows a single XCFPOLxx parmlib member to be used on all systems in the sysplex.

DEACTIVATE(othersys|ALL)
Specifies either that the logical partition (LPAR) where the specified system (othersys) resides is to be deactivated or all LPARs within the current system's addressing range are to be deactivated. The specified system (othersys) must be in the same sysplex as the current system and on the same processor as the system specified with the SYSTEM(sysname) keyword.

Specifying DEACTIVATE(othersys) where *othersys* is the same system specified in the NOSTATUS statement with the RESETTIME keyword has no effect; instead, use the DEACTTIME keyword. The following example is a correct use of the policy.

```
NOSTATUS(SYS1) DEACTTIME(25)
SYSGONE(SYS1) SYSTEM(SYS2) DEACTIVATE(SYS1)
```

Default: None. This parameter is required.

STORE(YES|NO)
Specifies whether or not the specified system (SYSTEM(sysname)) is to acquire the storage freed up by the deactivated LPAR(s).

Default: NO

ESTORE(YES|NO)
Specifies whether or not the specified system (SYSTEM(sysname)) is to acquire the expanded storage freed up by the deactivated LPARs.

Note: ESTOR is not supported in the z/Architecture environment.

Default: NO

Part 3. Appendixes

Appendix A. IEFSSNxx (subsystem definitions) - positional parameter form

Note: This appendix describes the IEFSSNxx parmlib member in the positional parameter form. IBM suggests that you use the IEFSSNxx parmlib member in the keyword parameter form instead, which allows subsystems defined in the member to use the dynamic SSI services.

IEFSSNxx is a parmlib member that contains parameters defining the primary subsystem and the various secondary subsystems that are to be initialized during system initialization. IEFSSNxx allows you to name the subsystem initialization routine to be given control during master scheduler initialization. IEFSSNxx also allows you to specify the input parameter to be passed to the subsystem initialization routine. Using the PRIMARY keyword, you can specify a primary subsystem name. The NOSTART keyword used with the PRIMARY keyword indicates that the system is not to issue an automatic start for the primary subsystem.

For information about writing subsystems, see *z/OS MVS Using the Subsystem Interface*.

The order in which the subsystems are initialized depends on the order in which they are defined in the IEFSSNxx parmlib member on the SSN parameter. Unless you are starting the Storage Management Subsystem (SMS), start the primary subsystem (JES) first. Some subsystems require the services of the primary subsystem in their initialization routines. Problems will occur if subsystems that use the subsystem affinity service in their initialization routines are initialized before the primary subsystem. If you are starting SMS, specify its record before you specify the primary subsystem record.

The format of the IEFSSNxx record for SMS is described in "Defining SMS through the IEFSSNxx member" on page 537.

Parameter in IEASYSxx (or supplied by the operator)

The SSN parameter in IEASYSxx identifies the IEFSSNxx member that the system is to use to initialize the subsystems, as follows:

```
SSN  {aa         }
     {(aa,bb,...)}
```

The two-character identifier, represented by aa (or bb, and so forth) is appended to IEFSSN to identify IEFSSNxx members of parmlib. If the SSN parameter is not specified, the system will use the IEFSSN00 parmlib member.

The order in which the subsystems are defined on the SSN parameter is the order in which they are initialized. For example, a specification of SSN=(13,Z5) would cause those subsystems defined in the IEFSSN13 parmlib member to be initialized first, followed by those subsystems defined in the IEFSSNZ5 parmlib member. you specify duplicate subsystem names in IEFSSNxx parmlib member, the system issues message IEE730. For more information, see the section on handling errors in defining your subsystem in *z/OS MVS Using the Subsystem Interface*.

Syntax rules for IEFSSNxx

The following rules apply to the creation of IEFSSNxx:

- Each record in IEFSSNxx defines one and only one subsystem that is to be initialized.
- Parameters begin with the comma following the *init-routine* and end with the first blank or comma.
- Each record in IEFSSNxx is 80 bytes long and has the following format:

```
ssname[,init-routine[,parm]][,PRIMARY[,NOSTART]]  comments
```

ssname
The subsystem name. The name can be up to 4 characters long; it must begin with an uppercase alphabetic or national character (#, @, or $), and the remaining characters (if any) can be alphanumeric or national. The name begins with the first non-blank character in the record and continues to the first comma or blank. When a blank follows the name, the system assumes that no initialization routine exists and that only comments (if any) follow.

In general, the subsystem name should be the same as the started procedure name. Some products (such as IMS and CICS), however, require the subsystem name to be different from the started procedure name. For more information, refer to the section on coding initialization routines for subsystem in *z/OS MVS Using the Subsystem Interface* and the documentation for the particular product.

init-routine
The name of the subsystem initialization routine. This name can be 1-8 characters long, and the characters can be alphanumeric or national (#, @, or $). The name begins with the first character following the first comma after "ssname" and continues to the next comma or blank. When a blank follows the name, the system assumes that no initialization routine input parameters are supplied and that only comments (if any) follow.

For information about writing subsystem initialization routines, see *z/OS MVS Using the Subsystem Interface*.

parm
Input parameters to be passed to the subsystem initialization routine. The input parameters are variable in length for the remainder of the 80-byte record (for a maximum of 60 characters); they begin after the comma that ended the "init-routine" name and end with the first blank. If blanks, commas, or single quotation marks are included in the input parameters, then the entire parm field must be enclosed in single quotation marks. If the parm field is enclosed in single quotation marks, a single quotation mark within the field must be specified as a double quotation mark.

PRIMARY
Parameter indicating the primary subsystem name. The primary subsystem is typically a job entry subsystem (either JES2 or JES3). This keyword indicates that the subsystem name (ssname) will be the primary subsystem name. Initialize the primary subsystem before any secondary subsystem(s). If you specify PRIMARY more than once, the system will issue a message.

NOSTART
Parameter indicating that an automatic start is not to be issued. NOSTART

indicates that an automatic start of the primary subsystem is not to be issued. NOSTART cannot be used without the PRIMARY parameter.

comments
Comments can be specified after the first blank in the record following the input parameters, or PRIMARY and NOSTART, for as many characters as remain in the 80-byte record.

You can include any number of records in the IEFSSNxx parmlib member.

IBM-supplied default for IEFSSNxx

If you do not specify the SSN system parameter, the system will use the IEFSSN00 parmlib member. IEFSSN00 specifies JES2 as the primary subsystem.

If you specify a set of IEFSSNxx members that do not identify a primary subsystem, the system will issue a message that prompts the operator to specify the primary subsystem.

Statements/parameters for IEFSSNxx

None.

Appendix B. Symbolic Parmlib Parser

The Symbolic Parmlib Parser allows you to verify symbolic substitutions without doing an IPL. The Parser is in SYS1.SAMPLIB and is activated as follows:

Activation

Enter the following commands to install the Parser for the first time:

```
EX 'SYS1.SAMPLIB(SPPINST)' '''SYS1.SAMPLIB(SPPPACK)'''
```

Enter the following commands to install a new version of the Parser. The / R option indicates to replace the previous instance.

```
EX 'SYS1.SAMPLIB(SPPINST)' '''SYS1.SAMPLIB(SPPPACK)''' / R
```

Note: See SYS1.SAMPLIB(SPPINST' for more details about the (/ R) replace option.

Enter the following command to activate the Parser, where *myid* equals your user ID.

```
TSO ex 'myid.PARMLIB.EXEC(SYSPARM)'
```

Note: Browse SYS1.SAMPLIB(SPPINST' for more details about installation and activation syntax.

Once activated, you see a panel similar to that shown in Figure 47.

```
COMMAND ===>


  Member Name            ===> IEAYSYSTT (Enter '?' for supported member list)
  Browse, Edit or View   ===> V         (Optional. Default is VIEW for symbol
                                         substitution, and EDIT otherwise.)
  Dsn ===>PARMLIB                                      Dflt:'SYS1.PARMLIB'
  Volume of above Dsn    ===>         (Optional)
  SYS1.NUCLEUS Volume    ===>         (If different from one catalogued)
  FMID                   ===>         (Optional. Enter '?' for list)

  Do Symbol Substitution ===> Y       (Y or N)
  Symbol Substitution Values:
                                     (Use '*' for Current val
    LOADxx Member        ===> LOADTT  Default LOAD00. LOADF1      <===Current
    Hardware Name        ===>         (Optional)      PR9672A     <===Current
    LPAR Name            ===>         (Optional)      AQFT        <===Current
    VM Userid            ===>         (Optional)                  <===Current
    SYSRES Volume        ===>         (Optional)      PRIPK5      <===Current
    Master Cat Volume    ===>         (Optional - Substituted for *MCAT*    )
```

Figure 47. Symbolic parmlib parser panel

Help is available on all panels by pressing PF1 or typing help on the command line.

Capabilities

The 'myid.PARMLIB.NOTES' file, member WORKFLOWS, gives an overview of the tool's capabilities. The following is an enhanced version of that file.

The Parmlib Processing tool provides several functions.

The tool can be used as a LOADxx processor by specifying:

- "Member Name" = desired LOADxx value
- "Do Symbol Substitution" = N
- "LOADxx Member" = blank

This brings up the LOADxx processing panel, which allows the following types of data validation:

- Syntax Checking

  Syntax checking examines the LOADxx parmlib member, looking for data specification errors such as missing commas and illegal keywords. Syntax checking is performed when an existing supported LOADxx parmlib member is selected for EDIT or BROWSE mode. Any errors detected are listed before the EDIT or BROWSE panel is initialized or displayed. Only data that is valid and recognized by the tool is displayed on the EDIT or BROWSE panel.
- Range and Type Checking

  Range and Type checking examines the data that is extracted from the LOADxx parmlib member, looking for value and type specification errors such as alphabetic characters being used when numerical data is required and numerical data that is not within a required range. The ISPF Panel and associated REXX program perform range and type checking. No range or type checking is done during syntax checking.
- Data Verification

  Data verification is performed by typing VERIFY on the command line of the BROWSE or EDIT panels. Data verification takes the valid data from syntax, range and type checking and examines what they mean in the current system environment. For example, one of the fields in a LOADxx parmlib member may contain a pointer the another parmlib member. Data verification checks to see if that member exists in the specified parmlib.

GRSRNLXX PROCESSING: The tool can be used as a GRSRNLxx syntax checker by specifying:

- "Member Name" = desired GRSRNLxx value
- "Do Symbol Substitution" = Y - substitution must be specified in order to have syntax checking performed
- "LOADxx Member" = blank
- Syntax Checking

  Syntax checking examines the GRSRNLxx parmlib member, looking for data specification errors such as missing commas and illegal keywords, invalid QNAMEs, and duplication of entries. Syntax checking is performed when an existing GRSRNLxx parmlib member is selected for EDIT or BROWSE mode and symbol substitution is specified. Any errors detected are noted in ISPF EDIT or BROWSE messages that state the type of error and the record number containing the error.

Note: GRSCNFxx parmlib member syntax checking is not supported by the tool.

The tool can be used as a parmlib symbolic preprocessor by specifying:

- "Member Name" = the name of a parmlib member containing symbolics to be processed
- "Do Symbol Substitution" = Y
- "LOADxx Member" = Desired LOADxx member. LOADxx is where a Symbol set is specified via the IEASYM statement.

The tool performs the following steps as a Parmlib Preprocessor:

1. Verifies that the LOADxx member exists where is is supposed to.
2. Reads the LOADxx member and examines its statements.
3. Reads the IEASYMxx members and thoroughly processes each one using the same code that is used at IPL time. Any errors are noted with the same messages used during IPL. If no IEASYMxx members can be found, a message will be issued.
4. Reads the IEASYSxx members and thoroughly processes each one. Any errors are noted with appropriate error messages. If no IEASYMxx members can be found, a message will be issued.
5. Searches for the input member and opens it if found.
6. Processes the input member through the symbol table built from reading the IEASYMxx members and the panel input fields. Any symbols that cannot be substituted are highlighted.
7. If DOALL is invoked, each line of the IEASYSxx member will be read to see if other parmlib members are implied. If these members are not found, an error message will be issued. If these members are found, they will be processed as described in Step 6. Any apparent processing problems for IEASYSxx are also noted.

The tool can be used to obtain a parmlib member selection list by specifying:

- "Member Name" = blank
- "Do Symbol Substitution" = Y
- "LOADxx Member" = blank

This places you in a panel displaying the different types of available parmlib members. When you select a parmlib member type the tool shows you a list of all members of that type in your parmlib data set, as specified on the main panel.

The tool can be used to obtain a parmlib member selection list from a concatenation of parmlibs by specifying:

- "Member Name" = blank
- "Do Symbol Substitution" = Y
- "LOADxx Member" = LOADxx (where that LOADxx member has defined a concatenated parmlib via the PARMLIB keyword)

The parmlib processor places you in a panel displaying the different types of available parmlib members. When you select a parmlib member type the tool shows you a list of all members of that type in your data set concatenation, as determined by the LOADxx member specified on the main panel.

Limitations

1. The tool does not support editing of SYSn.IPLPARM data sets. You can use the tool to generate the member in SYS1.PARMLIB or high_level_qualifier.PARMLIB and then you can copy the member into the appropriate SYSn.IPLPARM data set.
2. 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.
3. If the tool is running on a partial ship release or a point release, in the Parmlib Processor Member Selection Panel under the FMID field, use the LATEST keyword or a full ship release FMID to select the parser release used to process or verify the SYS1.PARMLIB member that you selected.
4. 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.
5. The Master Catalog information is not verified.
6. When in EDIT mode, the data in a member is not altered until it is saved. If you are running in split screen mode, you will not see the effects of the changes until the EDIT mode changes are saved.
7. Syntax checking is not done generally for the parmlib members that the tool lists as supported.
8. When processing an IEASYMxx parmlib member, the tool manipulates the data into an internal format. If that internal format exceeds 320080 bytes, the tool will terminate, with a message indicating that the IEASYMxx member is too large. It takes an IEASYMxx member on the order of 10000 lines long to reach this limit. If the limit is exceeded, the tool user will have to specify a different IEASYMxx member, possibly created from the original member but with data for non-applicable systems removed.

Appendix C. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out

punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface information

This book is intended to help the customer initialize and tune the MVS element of z/OS. This book documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/us/en/copytrade.shtml.

Adobe, Acrobat, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Product Number: 5650-ZOS

Printed in USA

SA23-1380-04