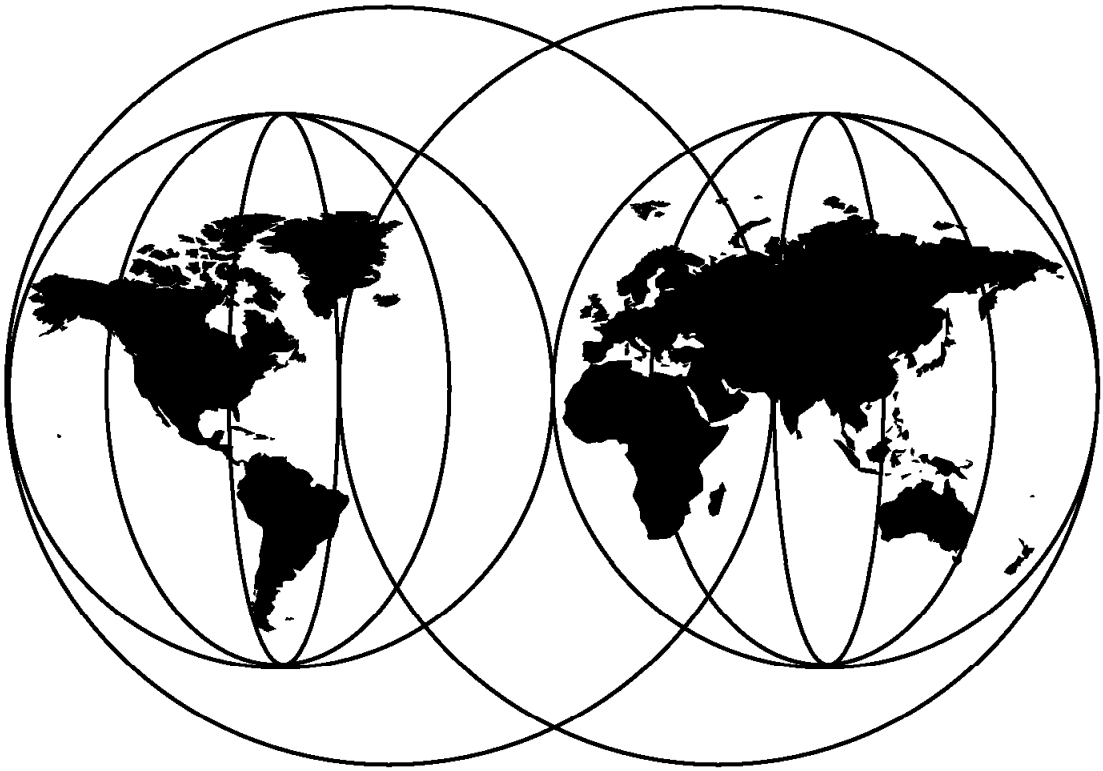
ABCs of OS/390 System Programming Volume 2

P. Rogers, G. Capobianco, D. Carey, N. Davies, L. Fadel, K. Hewitt,
J. Oliveira, F. Pita, A. Salla, V. Sokal, Y. F. Tay, H. Timm



International Technical Support Organization

www.redbooks.ibm.com

SG24-5652-00

International Technical Support Organization

SG24-5652-00

ABCs of OS/390 System Programming Volume 2

April 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 299.

First Edition (April 2000)

This edition applies to OS/390 Version 2 Release 8, Program Number 5647-A01, and to all subsequent releases and modifications.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, New York 12601-5400

Contents

Figures

Tables

Preface

This redbook is Volume 2 of a five-volume set that is designed to introduce the structure of an OS/390 and S/390 operating environment. The set will help you install, tailor, and configure an OS/390 operating system, and is intended for system programmers who are new to an OS/390 environment.

Chapter 1 provides an introduction to OS/390 implementation and daily maintenance, and how to IPL an OS/390 system. Described are the main parameters and system data sets necessary to IPL and run an OS/390 operating system and the main daily tasks that a system programmer performs in order to maximize the advantages that a well implemented operating system can offer to your I/T structure.

Chapter 2 describes subsystems and how to define them.

Chapter 3 describes the two job entry subsystems, JES2 and JES3.

Chapter 4 describes what factors should be considered when placing load modules or load libraries into storage.

Chapter 5 describes how data sets are managed in a catalog structure and the various types of catalogs.

The team that wrote this redbook

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

Paul Rogers is an OS/390 specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of OS/390. Before joining the ITSO 11 years ago, he worked in the IBM Installation Support Center (ISC) in Greenford, England as OS/390 and JES support for IBM EMEA.

Guillermo Capobianco is an IT Specialist in IBM Global Services PSS Argentina. He has five years of experience working with customers on MVS, MVS-related program products, and OS/390. He is currently leading a technical group providing on-site customer support for the OS/390 platform.

David Carey is a Senior IT Availability Specialist with the IBM Support Center in Sydney, Australia, where he provides defect and nondefect support for CICS, CICSPlex/SM, MQSeries, and OS/390. David has 19 years of experience within the information technology industry, and was an MVS systems programmer for 12 years prior to joining IBM.

T. Nigel Davies is a Systems Specialist in IBM Global Services Product Support Services (PSS) in the United Kingdom. He has 10 years of IT experience in various roles, ranging from operations to PC and LAN support to mainframe systems programming. He joined IBM in 1997 with eight years of experience as a VM/VSE systems programmer, and since joining IBM has cross-trained in OS/390 systems skills. His areas of expertise include VM and VSE systems

programming, installation, and technical support, and more recently, OS/390 installation and support.

Luiz Fadel - IBM Brazil

Ken Hewitt is an IT Specialist in IBM Australia. He has over 10 years of experience working with S/390 customers in a range of roles from CE to System Engineer. His areas of expertise include I/O and OSA configuration.

Joao Natalino Oliveira

Joao Natalino de Oliveira is a certified I/T consulting specialist working for the S/390 in Brazil providing support for Brazil and Latin America. He has 24 years of experience in large systems including MVS-OS/390. His areas of expertise include performance and capacity planning, server consolidation and system programming. He has a bachelor degree in Math and Data Processing from Fundação Santo André Brazil.

Fabio Chaves Pita - IBM Brazil

Alvaro Salla has 30 years of experience in OS operating systems (since MVT). He has written several redbooks on S/390 subjects. Retired from IBM Brasil, he is now a consultant for IBM customers.

Valeria Sokal is an MVS system programmer at Banco do Brasil. She has 11 years of experience in the mainframe arena. Her areas of expertise include MVS, TSO/ISPF, SLR, and WLM.

Yoon Foh Tay is an IT Specialist with IBM Singapore PSS (S/390). He has six years of experience on the S/390 platform, providing on-site support to customers.

Hans-Juergen Timm is an Advisory Systems Engineer in IBM Global Services PSS Germany. He has 20 years of experience working with customers in the areas of MVS and OS/390, software and technical support, and planning and management. He also worked six years as an MVS Instructor in the IBM Education Centers in Mainz and Essen, Germany. His areas of expertise include implementation support for OS/390, Parallel Sysplex, UNIX System Services, and Batch Management.

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 321 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. OS/390 implementation and daily maintenance

Regardless of which method you use to install the OS/390 operating system (ServerPac, CBPDO...) it is highly recommended that you understand the basic aspects of the OS/390 implementation, from installation, to preparing to do daily activities after IPL.

This chapter will give you an overview of the basics as well as the IPL process. We will describe the main parameters and system data sets necessary to IPL and run an OS/390 operating system and the main daily tasks that a system programmer performs in order to maximize the advantages that a well-implemented operating system can offer to your IT structure.

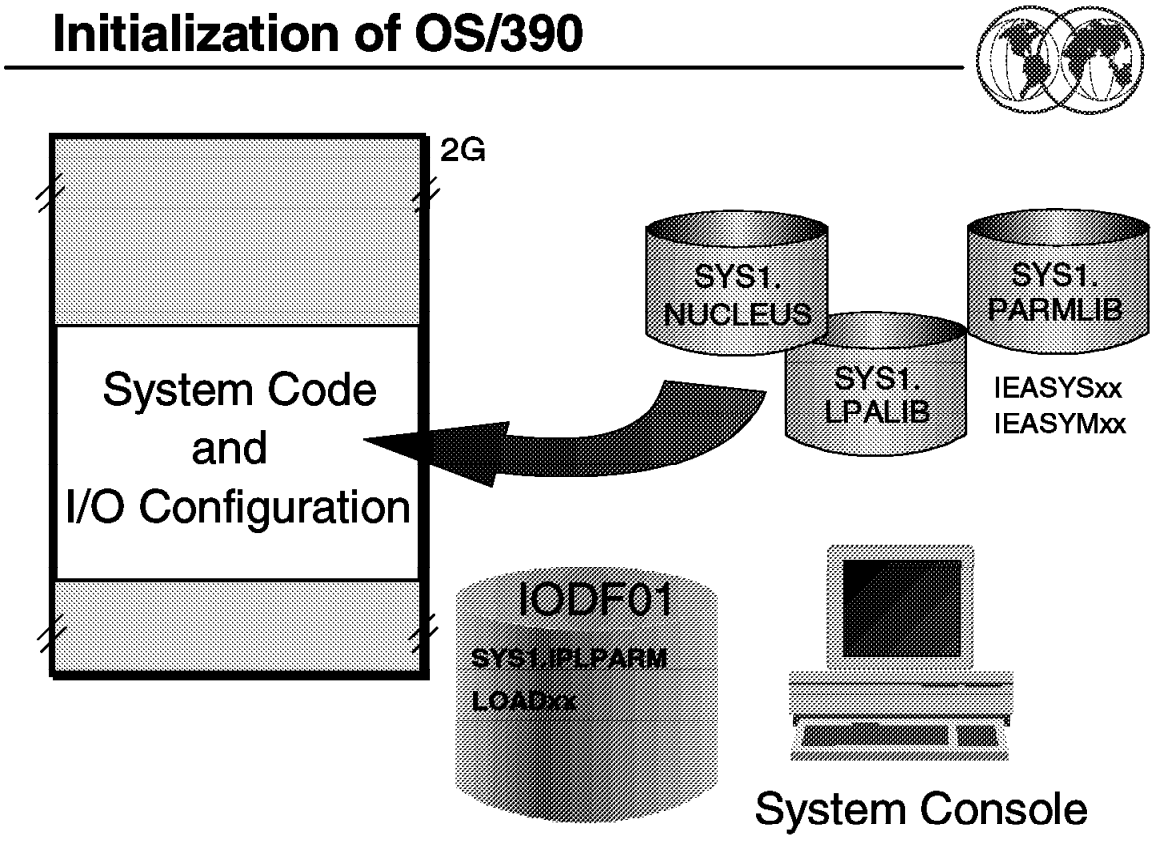


Figure 1. Initialization of OS/390

1.1 IPL of OS/390

When the system hardware is ready, you can use the system console to load the system software.

During initialization of an OS/390 system, the operator uses the system console, which is connected to the processor controller or support element. From the system console, the operator initializes the system control program during the nucleus initialization program (NIP) stage.

During the NIP stage, the system might prompt the operator to provide system parameters that control the operation of MVS. The system also issues informational messages that inform the operator about the stages the initialization process.

The LOADxx parmlib member allows your installation to control the initialization process. For example, in LOADxx, you specify IEASYSxx or IEASYMxx members that the system is to use; the system does not prompt the operator for system parameters that are specified in those members. Instead, it uses the values in those members.

The definition of these parameters are discussed in this chapter.

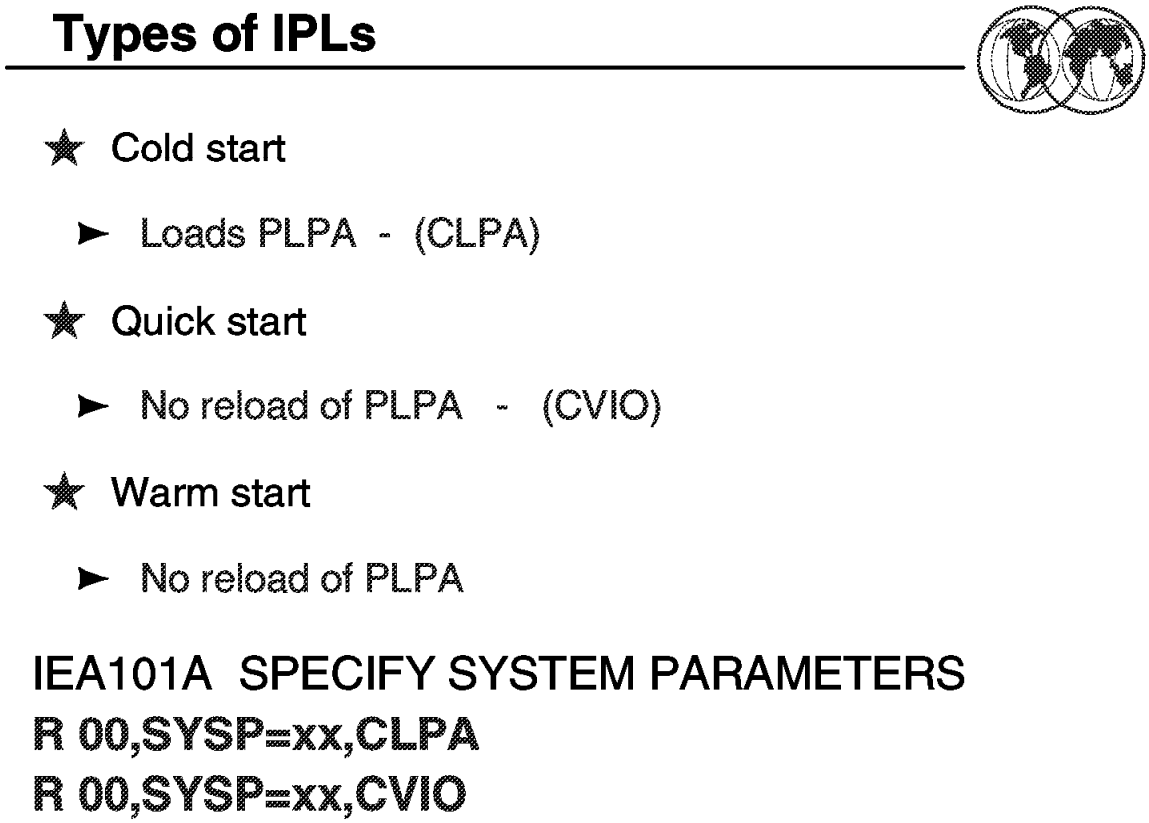


Figure 2. Types of IPLs

1.1.1 Types of IPL

It is possible that in this discussion we will use terms that are unfamiliar to you (like LPA, SQA, VIO, and so forth). The objective is to condense the IPL types and process (next foil) in order to give you an overview of the IPL and its interactions with the operator. Further, we will discuss the main aspects of the architecture as well as the mechanisms of OS/390 mentioned here.

Depending on the level of customization, a system IPL can bring up many different configurations, but there are only three basic types of IPL:

Cold Start Any IPL that loads or reloads the Pageable Link Pack Area (PLPA) and does not preserve VIO data sets. The first IPL after system installation is always a cold start because the PLPA must be initially loaded. At the first IPL after system installation the PLPA will automatically be loaded from the LPALST concatenation. The page data sets for this IPL will be those specified in IEASYSxx, plus any additional ones specified by the operator. Subsequent IPLs need only be cold starts if the PLPA has to be reloaded either to alter its contents or to restore it when it has been destroyed. The PLPA needs to be reloaded:

- At the first IPL after system initialization, when the system loads it automatically
- After modifying one or more modules in the LPALST concatenation
- After the PLPA page data set has been damaged and is therefore unusable, so its contents must be restored. The PLPA can be reloaded by responding *CLPA* (Create Link Pack Area) to the SPECIFY SYSTEM PARAMETERS prompt.

Quick Start Any IPL that does not reload the PLPA and does not preserve VIO data sets. The system will re-create page and segment tables to match the in-use PLPA. You would normally perform a Quick Start IPL after a power up. The PLPA from the previous session can be used without reloading it from the LPALST concatenation. A Quick Start can be initiated by replying *CVIO* (Clear VIO) to the SPECIFY SYSTEM PARAMETERS prompt.

Warm Start Any IPL that does not reload the PLPA and preserves journaled VIO data sets. The operator does not enter *CLPA* or *CVIO* at the SPECIFY SYSTEM PARAMETERS prompt. Any definitions of existing page data sets as non-VIO local page data sets are preserved.

1.1.1.1 System Parameters prompt

If the LOADPARM has been set to indicate the operator should be prompted for system parameters (A, P, S or T), NIP issues the following prompt on the NIP console:

```
IEA101A  SPECIFY SYSTEM PARAMETERS
```

At this point, the operator can respond in one of a number of ways:

- Pressing Enter will cause the system to use the parameters as coded in SYS1.PARMLIB(IEASYS00).
- Entering *SYSP=xx* will cause the parameters in SYS1.PARMLIB(IEASYS*xx*) to be used.
- Entering multiple IEASYSxx members, *SYSP=(xx,yy,..)*, for multiple concatenations to be used.

Specifying any of the individual parameters that appear in IEASYSxx will cause these parameters to be used instead of the ones coded in IEASYSxx except PAGE parameters. If multiple IEASYSxx members are searched, the system *always* reads IEASYS00 first, followed by the additional ones you specify. If the same parameters are found in one or more IEASYSxx members, the latest overriding parameter is used.

1.1.1.2 PAGE parameters

The PAGE 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, which includes the optional page data set specified by the DUPLEX parameter. 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.

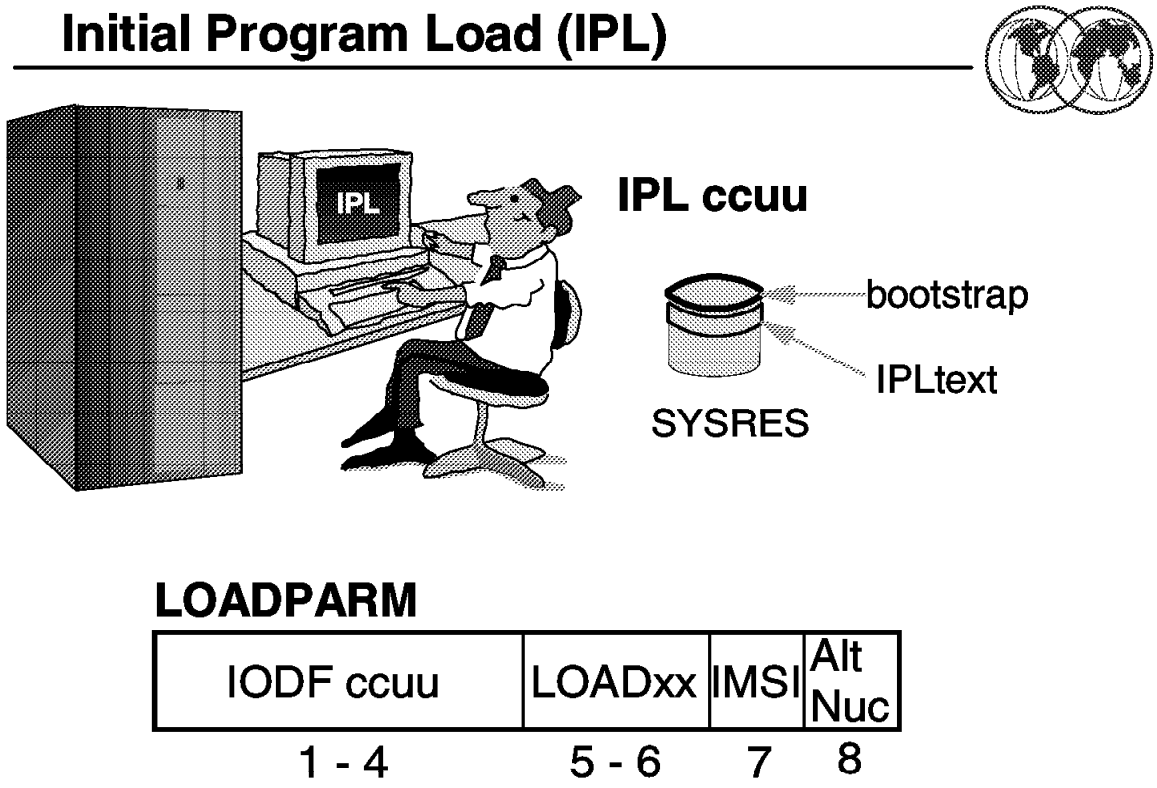


Figure 3. Initial program load

1.1.2 The IPL process

An Initial Program Load (IPL) is the act of loading a copy of the operating system from disk into the CPU's central storage and executing it. Not all disks attached to a CPU will have loadable code on them. A disk that does is generally referred to as an IPLable disk, and more specifically as the SYSRES volume.

IPLable disks will contain a bootstrap module at cylinder 0 track 0. At IPL, this bootstrap is loaded into storage at real address zero and control is passed to it. The bootstrap then reads the IPL control program IEAIPL00 (also known as IPL text) and passes control to it. This in turn starts the more complex task of loading the operating system and executing it.

Attempts to IPL from a disk that does not contain IPL text results in an error condition.

IEAIPL00 prepares an environment suitable for starting the programs and modules that make up the operating system. First, it clears central storage to zeros before defining storage areas for the master scheduler. It then locates the SYS1.NUCLEUS data set on the SYSRES volume and loads a series of programs from it known as IPL Resource Initialization Modules (IRIMs). These IRIMs will then start to construct the normal operating system environment of control blocks and subsystems that make up an OS/390 system. Some of the more significant tasks performed by the IRIMs are to:

- Read the LOADPARM information entered on the hardware console at the time the IPL command was executed. LOADPARM is discussed in 1.1.2.1, "The Load Parameter (LOADPARM)" on page 6.

- Search the volume specified in the LOADPARM as containing the IODF data set for the LOADxx member—the value of *xx* is also taken from the LOADPARM. IRIM will first attempt to locate LOADxx in SYS0.IPLPARM. If this is unsuccessful, it will look for SYS1.IPLPARM and so on up to and including SYS9.IPLPARM. If, at this point, it still has not been located, the search will continue in SYS1.PARMLIB.

  When a LOADxx member has been successfully located it will be opened and information including the nucleus suffix (unless overridden in LOADPARM), the master catalog name, and the suffix of the IEASYSxx member to be used, will be read from it.
- Load the MVS nucleus.
- Initialize virtual storage in the master scheduler address space for the System Queue Area (SQA), the Extended SQA (ESQA), the Local SQA (LSQA), and the Prefixed Save Area (PSA). At the end of the IPL sequence, the PSA will replace IEAIPL00 at real storage location zero, where it will then stay.
- Initialize real storage management, including the segment table for the master scheduler, segment table entries for common storage areas, and the page frame table.

The last of the IRIMs then loads the first part of the Nucleus Initialization Program (NIP), which then invokes the Resource Initialization Modules (RIMs), one of the earliest of which starts up communications with the NIP console defined in SYS1.NUCLEUS.

1.1.2.1 The Load Parameter (LOADPARM)

The most basic level of control is through the Load Parameter (LOADPARM). This is an eight-character value made up of four fields that the operating system will refer to as it IPLs, causing it to take the specified actions. The LOADPARM is made up of the following components:

IODF ccuu The IODF device address. This is also the device on which the search for the LOADxx member of SYSn.IPLPARM or SYS1.PARMLIB begins. The first four characters of the LOADPARM specify the hexadecimal device address for the device that contains the I/O Definition File (IODF) VSAM data set. This is also the device on which the search for the LOADxx member of SYSn.IPLPARM or SYS1.PARMLIB begins. This device address can be in the range X′0000′ to X′FFFF′. If the address is less than four digits, pad it with leading zeros—for example, a device address of *c1* should be entered as *00c1*. If you do not specify the device address, the system uses the device address of the system residence (SYSRES) volume.

LOADxx The xx suffix of the LOADxx parmlib member. The LOADxx member contains information about the name of the IODF data set, which master catalog to use, and which IEASYSxx members of SYS1.PARMLIB to use.

The default for the LOADxx suffix is zeroes. The system reads the LOADxx and NUCLSTxx members from SYSn.IPLPARM or SYS1.PARMLIB on the volume specified on the LOAD parameter (or the SYSRES volume, if no volume is specified). After the system opens the master catalog, the system reads all other members from the SYS1.PARMLIB data set that is pointed to by the master catalog. This SYS1.PARMLIB might be different from the SYS1.PARMLIB to which the LOAD parameter points.

IMSI x The prompt feature character specifies the prompting and message suppression characteristics that the system is to use at IPL. This character is commonly known as an initialization message suppression indicator (IMSI).

Some IMSI characters suppress informational messages from the system console, which can speed up the initialization process and reduce message traffic to the console. It can also cause you to miss some critical messages, so you should always review the hardcopy log after initialization is complete.

Alt Nuc x The last character specifies the alternate nucleus identifier (0–9). Use this character at the system programmer's direction. If you do not specify an alternate nucleus identifier, the system loads the standard (or primary) nucleus (IEANUC01), unless the NUCLEUS statement is specified in the LOADxx member.

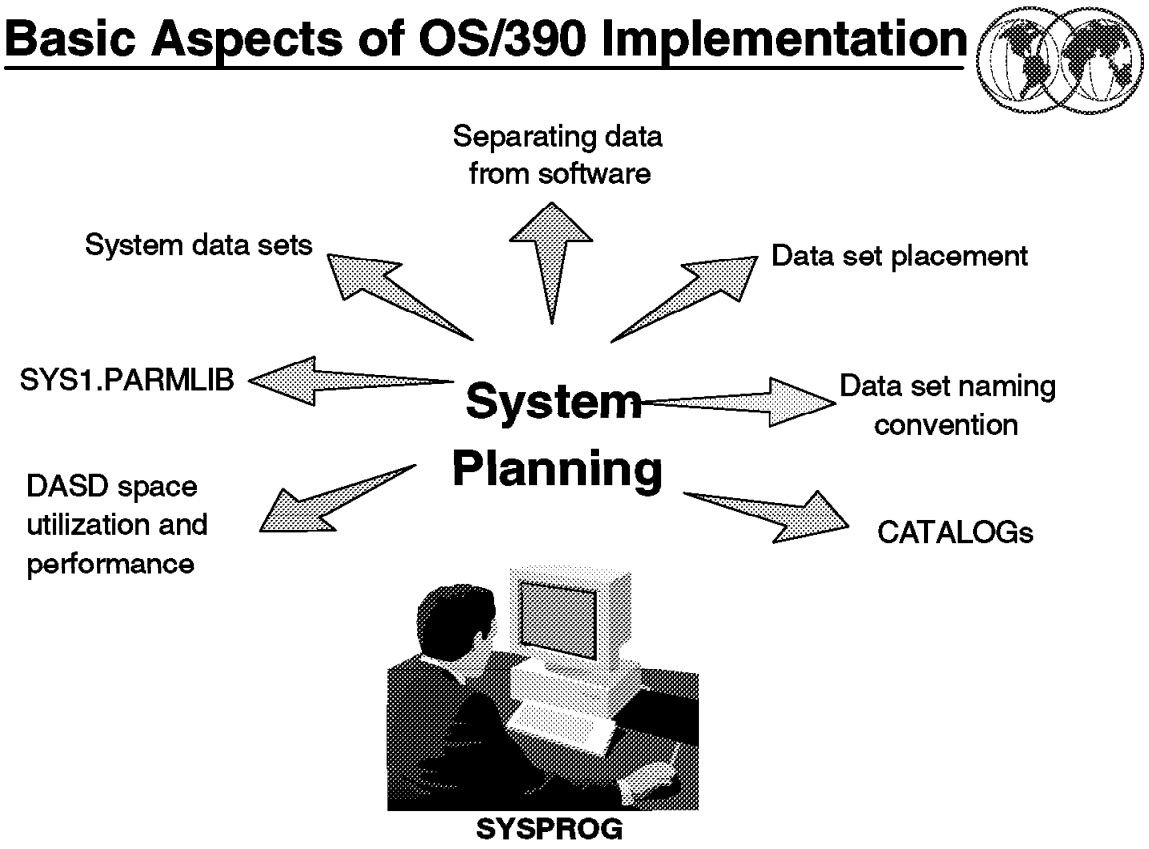


Figure 4. System planning

1.2 Basic aspects of OS/390 implementation

When you build an OS/390 system, you must balance the needs of your installation to build one that meets its needs well. While this will sometimes mean compromise, it more often means finding ways to build a flexible system that is easy to install, easy to migrate, easy to extend, and most important, easy to manage. When applied well, using a well-planned structure, this flexibility can be used to control the time it takes to install and migrate new systems or new software levels throughout an installation.

A phased approach will often prove most feasible and can be begin to control the installation and migration workload in the least time. This provides you benefits, starting with the next installation and migration cycle, while controlling the work involved in implementation.

Some aspects you might have to consider in adopting a structured approach to installation are:

- SYS1.PARMLIB and parmlib concatenation (logical parmlib)
- Catalogs and indirect catalog entries
- Separating data from software
- Placing data sets on specific volumes
- Choosing a naming convention for data sets
- DASD space utilization and performance
- System and installation requirements

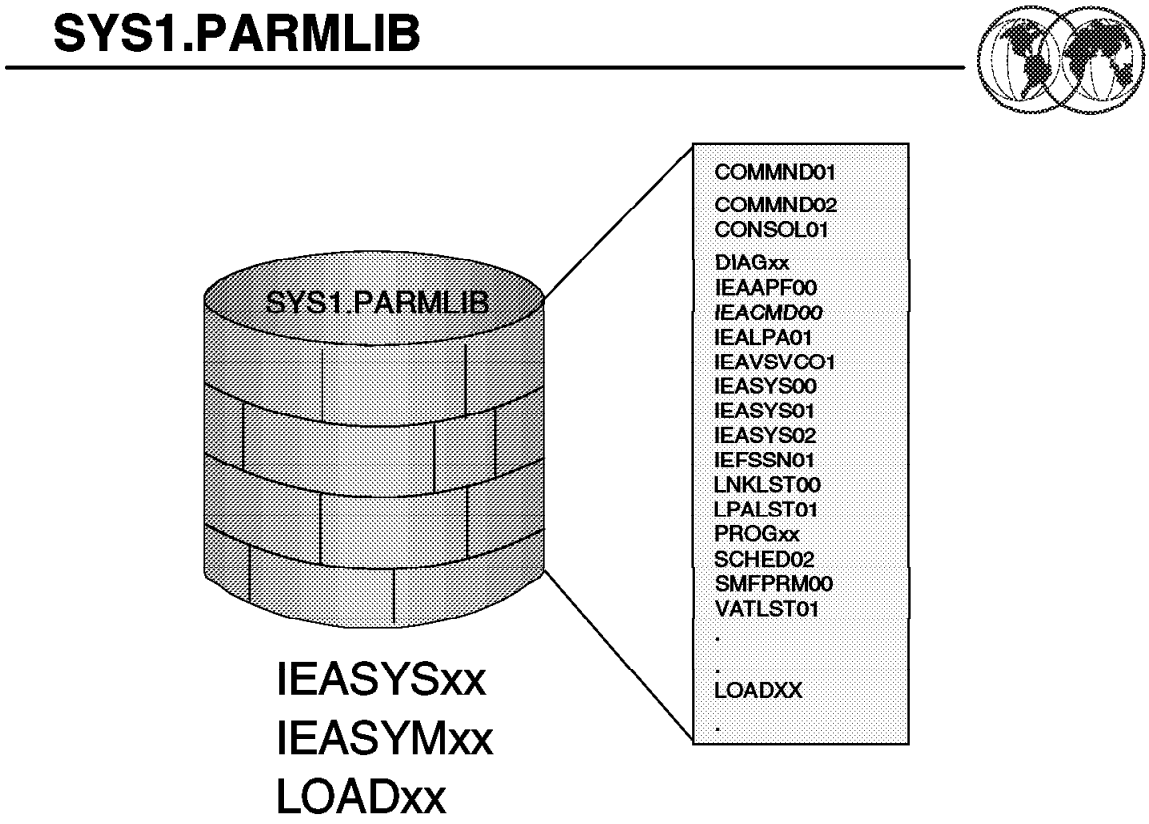


Figure 5. Overview of parmlib members

1.3 Overview of parmlib members

SYS1.PARMLIB is a required partitioned data set that contains IBM-supplied and installation-created members, which contain lists of system parameter values. You can include user-written system parameter members in SYS1.PARMLIB before installing a product.

SYS1.PARMLIB is read by the system at IPL, and later by such components as the system resource manager, the TIOC, and GTF, which are invoked by operator commands. The purpose of parmlib is to provide many initialization parameters in a pre-specified form in a single data set, and thus minimize the need for the operator to enter parameters. The SYS1.PARMLIB data set can be blocked and can have multiple extents, but it must reside on a single volume.

Note: This is the most important data set in an OS/390 operating system.

All parameters of the SYS1.PARMLIB data set are described in Appendix A, “OS/390 installation and customization” on page 285. Three of the most important parmlib members are discussed in this chapter. They are:

IEASYSxx This parmlib member allows the specification of system parameters that are valid responses to the IEA101A SPECIFY SYSTEM PARAMETERS message; see 1.1.1.1, “System Parameters prompt” on page 4. 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.

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.

LOADxx Contains information about the name of the IODF data set, which master catalog to use, and which IEASYSxx members of SYS1.PARMLIB to use.

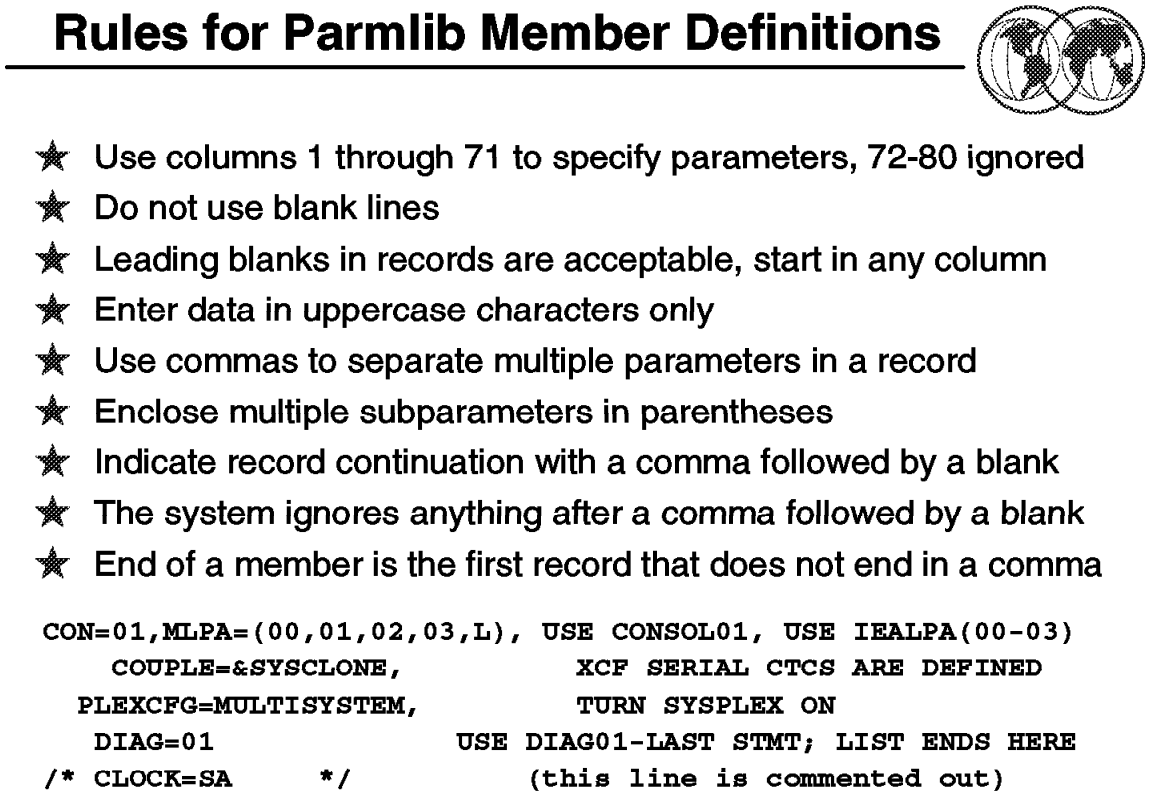


Figure 6. Syntax rules for parmlib members

1.3.1 Syntax rules for parmlib members

The following rules apply to the creation of parmlib members:

- Use columns 1 through 71 to specify parameters. The system ignores columns 72 through 80.
- Do not use blank lines.
- Leading blanks in records are acceptable. Therefore, a parameter need not start at column 1.
- 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.
- The system ignores anything after a comma followed by one or more blanks. You can use the remainder of the line for comments.
- 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.

The visual shows an example of the syntax used in creating a parmlib member.

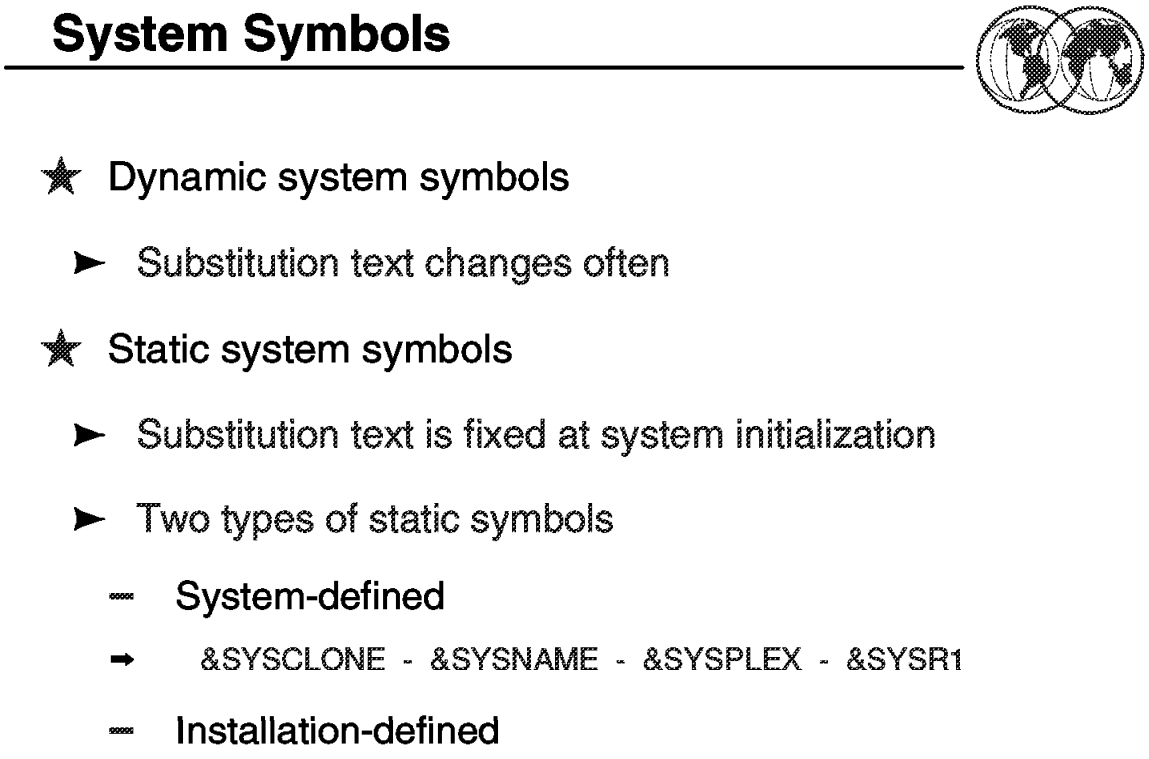


Figure 7. System symbols

1.3.2 Types of system symbols

The following terms describe the types of system symbols:

Dynamic 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 A symbol whose substitution text is defined at system initialization and remains fixed for the life of an IPL. One exception, &SYSPLEX, has a substitution text that can change at one point in an IPL. Static system symbols are used to represent fixed values such as system names and sysplex names.

There are two types of static system symbols:

System-defined 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
- SC68
- &SYSPLEX
- &SYSR1

Note: Your installation *cannot* define substitution texts for &SYSR1.

Installation-defined 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.

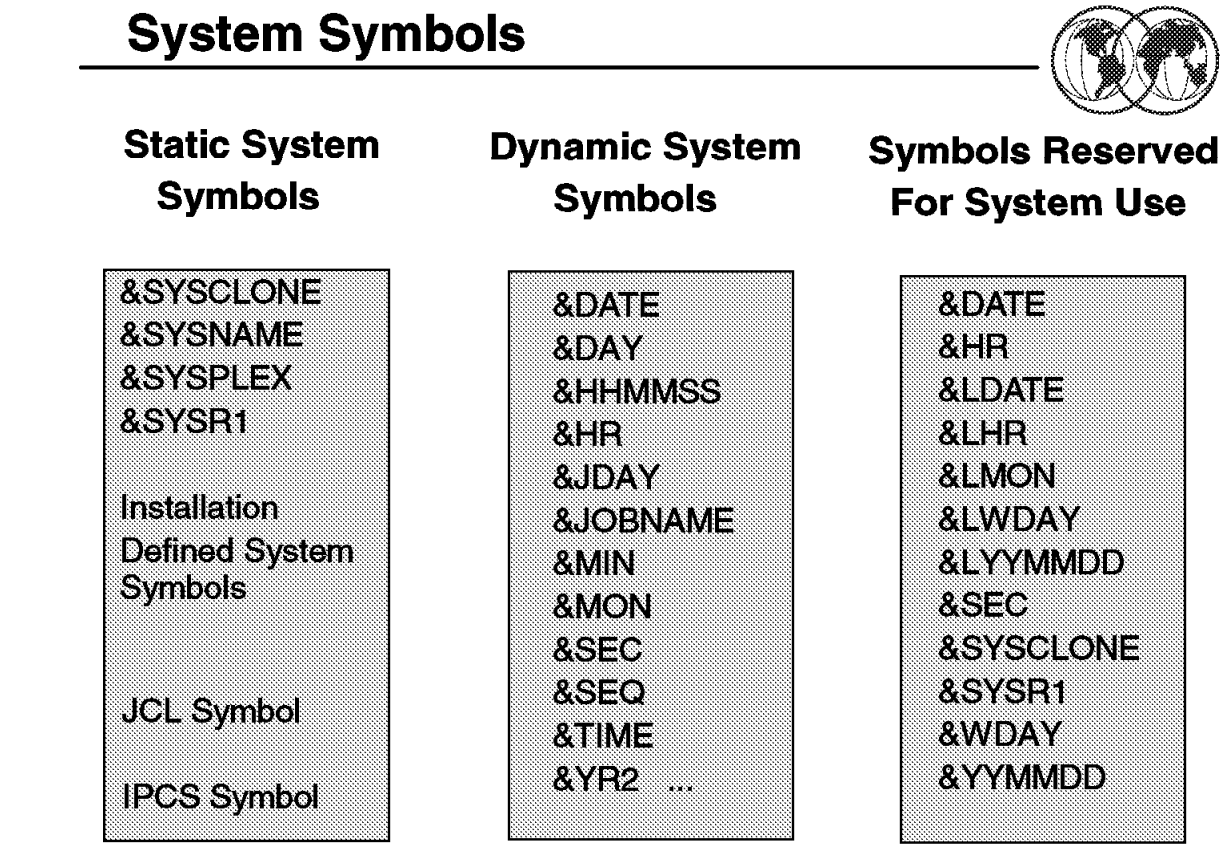


Figure 8. System symbols

1.3.3 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 up to 99 additional static system symbols. You can enter the DISPLAY SYMBOLS operator command to display the static texts that are in effect for a system. See *OS/390 MVS System Commands*, GC28-1781, for information about how to enter DISPLAY SYMBOLS. You can use the SYMDEF interactive problem control system (IPCS) subcomm static system symbol definitions. See *OS/390 MVS IPCS Commands*, GC28-1754, for information about the SYMDEF subcommand.

In addition to the system symbols listed above, the system allows you to define and use symbols in:

JCL symbol Symbols can be used 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 *OS/390 MVS JCL Reference*, GC28-1757.

IPCS symbol Symbols can be use by IPCS to represent data areas in dumps that are processed with IPCS subcommands.

For more information about using IPCS symbols, see *OS/390 MVS IPCS User's Guide*, GC28-1756.

Note

Although IBM recommends the use of ending periods on system symbols, the text of this chapter does not specify them, except in examples, out of consideration for readability.

1.3.3.1 Dynamic system symbols

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 reflects 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 is the name of the job assigned to the address space that calls the symbolic substitution service, not the address space of the started task.

1.3.3.2 Symbols reserved for system use

When you define additional system symbols in the IEASYMxx parmlib member, ensure that you do not specify the names reserved for system use.

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

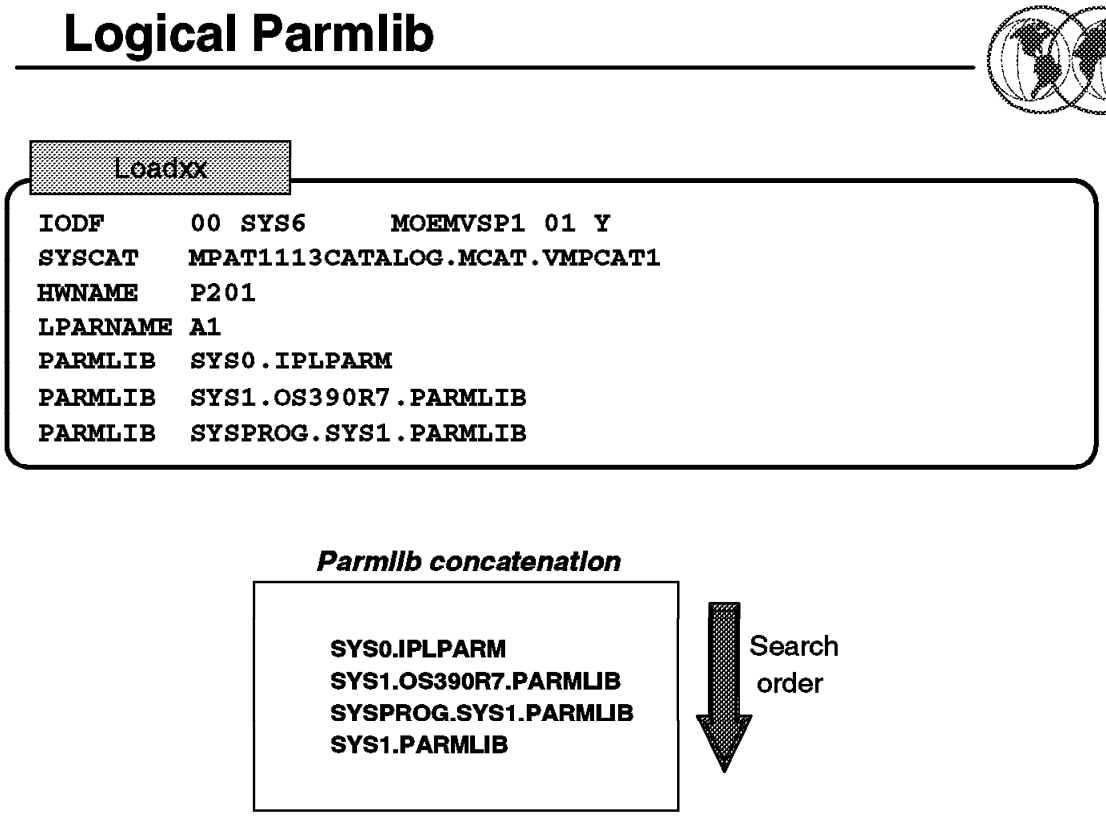


Figure 9. Logical parmlib

1.3.4 Using parmlib concatenation

As of OS/390 R2, you can concatenate up to 10 data sets to SYS1.PARMLIB, in effect creating a *logical parmlib*. You define the concatenation in the LOADxx member of SYS1.PARMLIB or SYSn.IPLPARM. 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. You can also use the SETLOAD operator command to switch from one logical parmlib to another without an IPL.

The benefits of using parmlib concatenation is that it gives you greater flexibility in managing parmlib and changes to parmlib members.

Note: If you do not specify at least one PARMLIB statement in the LOADxx, 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. However, if there is no PARMLIB statement in the parmlib concatenation and there is no IEFPARM DD statement, Master Scheduler processing will use only SYS1.PARMLIB.

For more information on logical parmlib, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

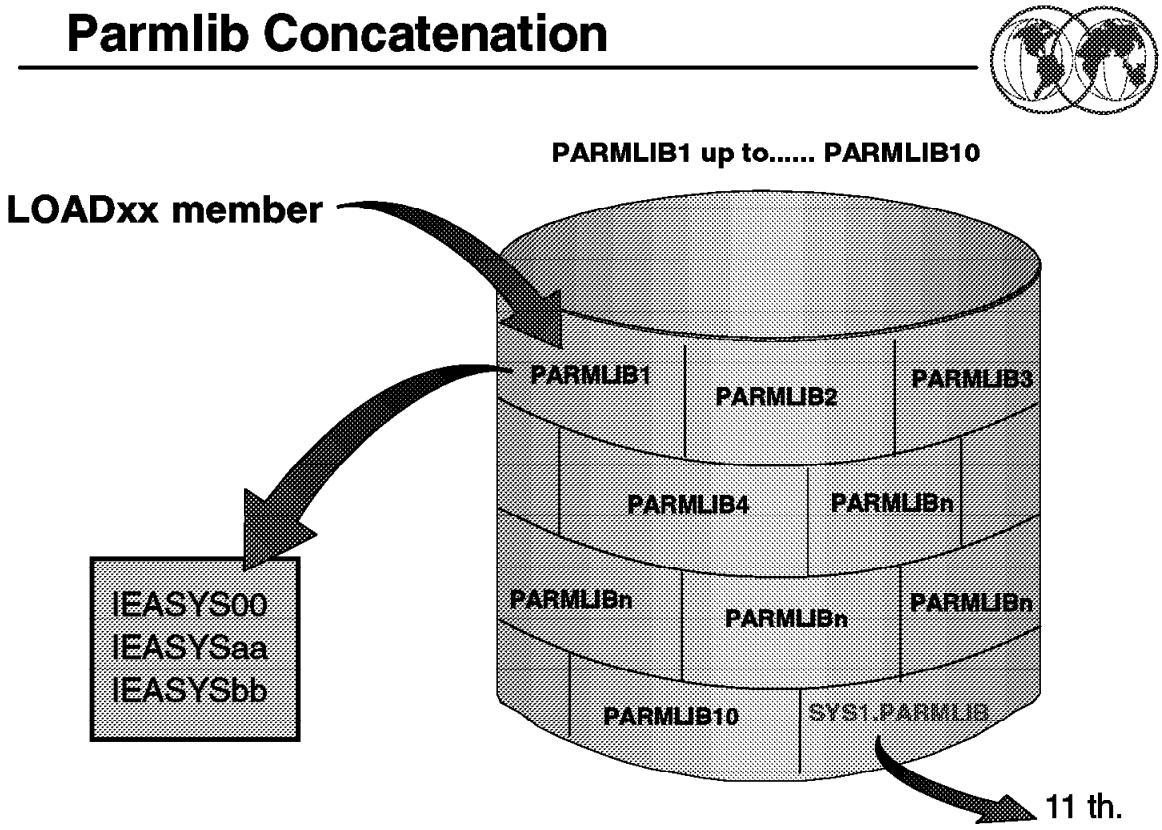


Figure 10. Parmlib concatenation

1.3.5 Description and use of the parmlib concatenation

This section 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).

The parmlib concatenation is a set of up to 10 partitioned data sets defined through PARMLIB statements in the LOADxx member of either SYSn.IPLPARM or SYS1.PARMLIB. These members contain 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 eleventh or last data set in the concatenation and is the default parmlib concatenation if no PARMLIB statements exist in LOADxx. For specific information on how to define a logical parmlib concatenation, see 1.3.11, “LOADxx (system configuration data sets)” on page 28. 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 an IPL you may issue a SETLOAD command to switch to a different parmlib concatenation which contains PDSEs. For information on processing of concatenated data sets see *DFSMS/MVS Using Data Sets*, SC26-4922.

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.

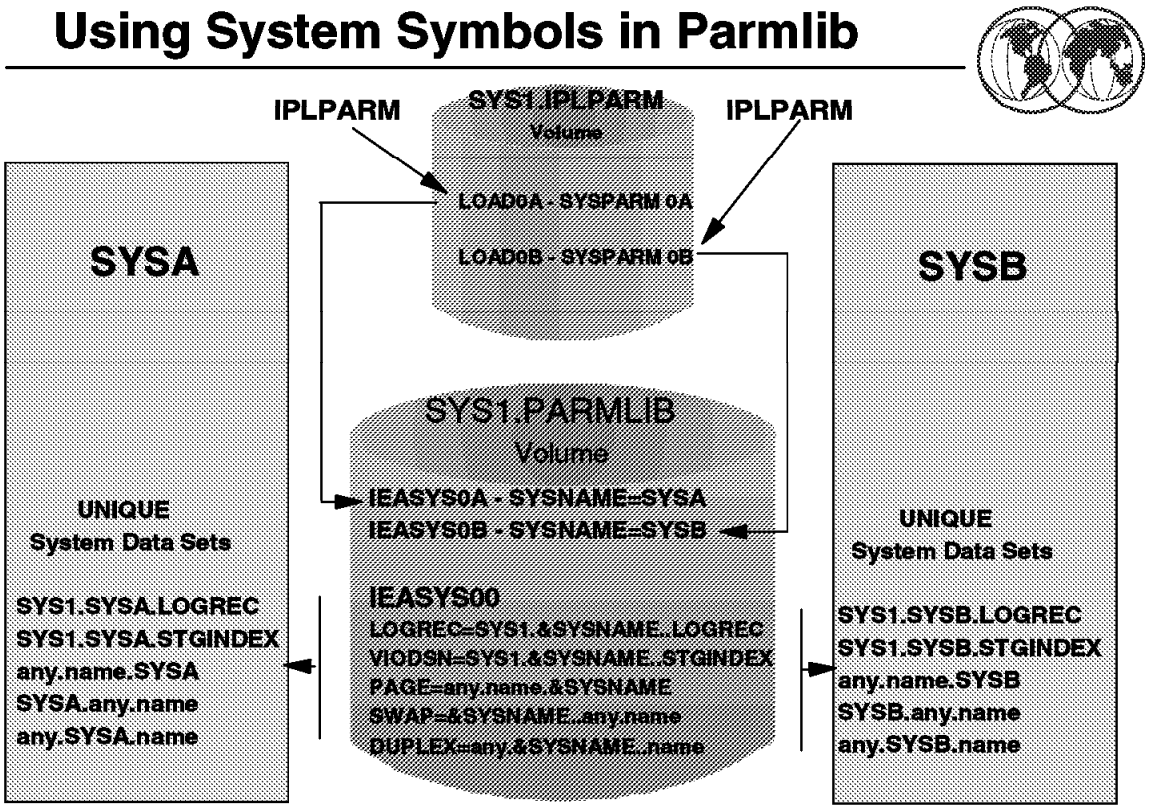


Figure 11. Using system symbols in parmlib

1.3.6 Using system symbols in parmlib

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

1. Know the rules for using system symbols in parmlib.
2. Determine where to use system symbols in parmlib.
3. Verify system symbols in parmlib.

1.3.6.1 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 to 8 characters between the ampersand and the period (or the next character, if you do not specify a period)

   If the system finds a system symbol that does not end with a period, it substitutes 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 processes the (2) as substring syntax for the system symbol, regardless of how you intended to use the string in the command.

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 SC68 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 the following command is to start CICS:

   ```
   S CICS,JOBNAME=CICSSC68,...
   ```

   The resolved substitution text for SC68 cannot exceed four characters because jobnames 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 jobname CICSSYS1. However, a substitution text of SYSTEM2 is not valid because it resolves to the jobname of CICSSYSTEM2, which exceeds the allowable maximum of eight characters.
6. If you use &SYSCLONE, ensure that the LOADxx parmlib member indicates that the system is to check for a unique &SYSCLONE substitution text on each system.
7. If you use &SYSNAME, ensure that its substitution text is unique on each system.
8. Do not specify system symbols in the values on the OPI and SYSP parameters in the IEASYSxx parmlib member.
9. Do not specify system symbols that were introduced in MVS/ESA SP 5.2 in parmlib members that are processed by pre-MVS/ESA SP 5.2 systems.
10. Do not specify any system symbols in parmlib members that do not support system symbol substitution.

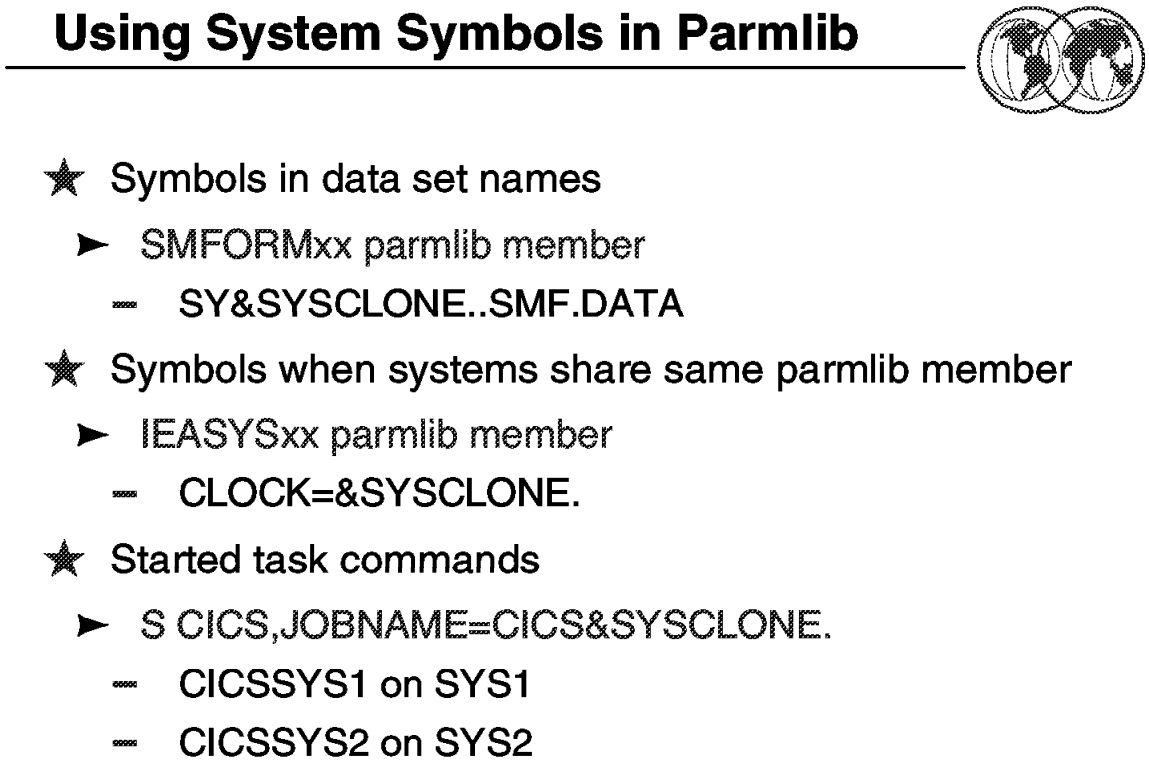


Figure 12. Using system symbols in parmlib

1.3.7 Using 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 could 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 produces 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.

Shared 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 could 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 could specify the SC68 system symbol on a START command in COMMNDxx to start unique instances of CICS:

```
S CICS,JOBNAME=CICSSC68,...
```

When each system in the sysplex initializes with the same COMMNDxx member, SC68 resolves to the substitution text that is defined on each system. If SC68 is defined to SYS1 and SYS2 on the respective systems, the systems start CICS with the following jobnames:

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

Note that the resolved substitution text for SC68 is eight characters long, which is the maximum length for jobnames.

1.3.7.1 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.

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.

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

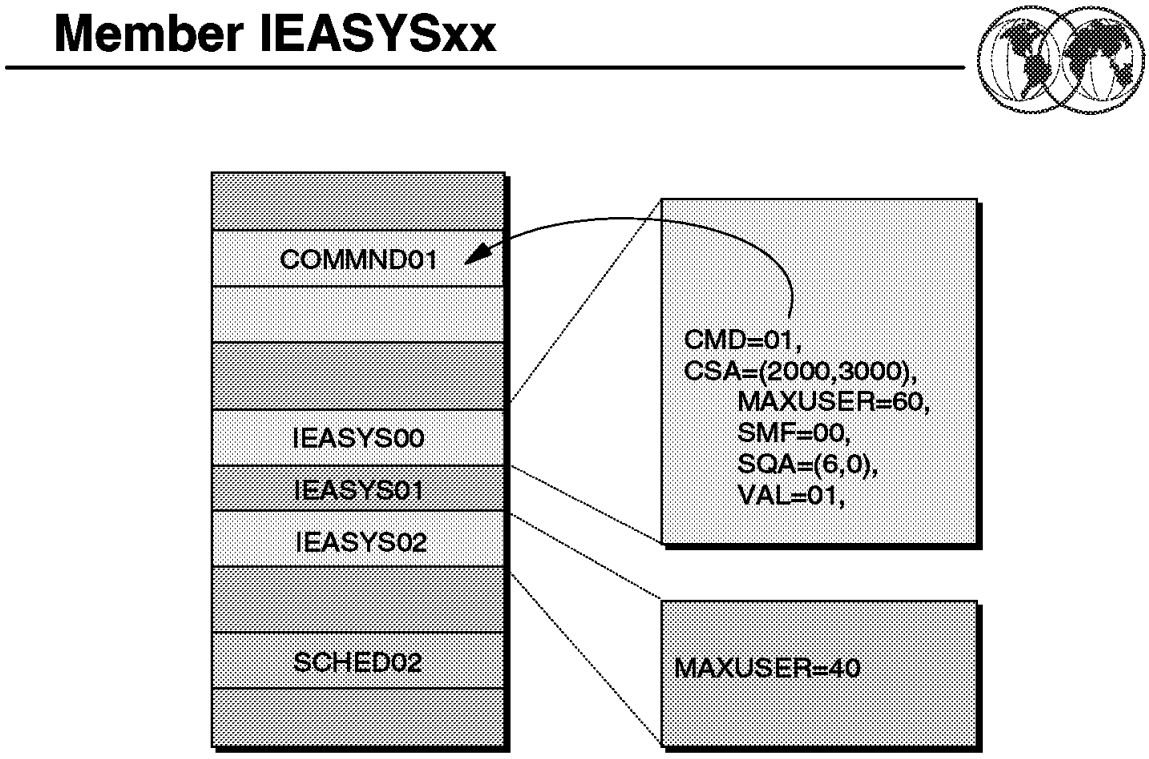


Figure 13. IEASYSxx member

1.3.8 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.

```
R 00,SYSP=xx
```

The operator specifies this parameter to specify an alternate system parameter list in addition to IEASYS00.

- Specify one or more suffixes of alternate IEASYSxx members on the SYSPARM parameter in the LOADxx or in the IEASYMxx parmlib member.

1.3.8.1 Overview of IEASYSxx parameters

See A.2, “IEASYSxx SYS1.PARMLIB parameters” on page 287 for a list of all the 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 other 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.

1.3.8.2 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.

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 on the LOAD parameter, see the section on loading the system software in *OS/390 MVS System Commands*, GC28-1805.

1.3.8.3 Statements/Parameters for IEASYSxx

For detailed information about statements/parameters, see *OS/390 Initialization and Tuning Reference*, SC28-1752.

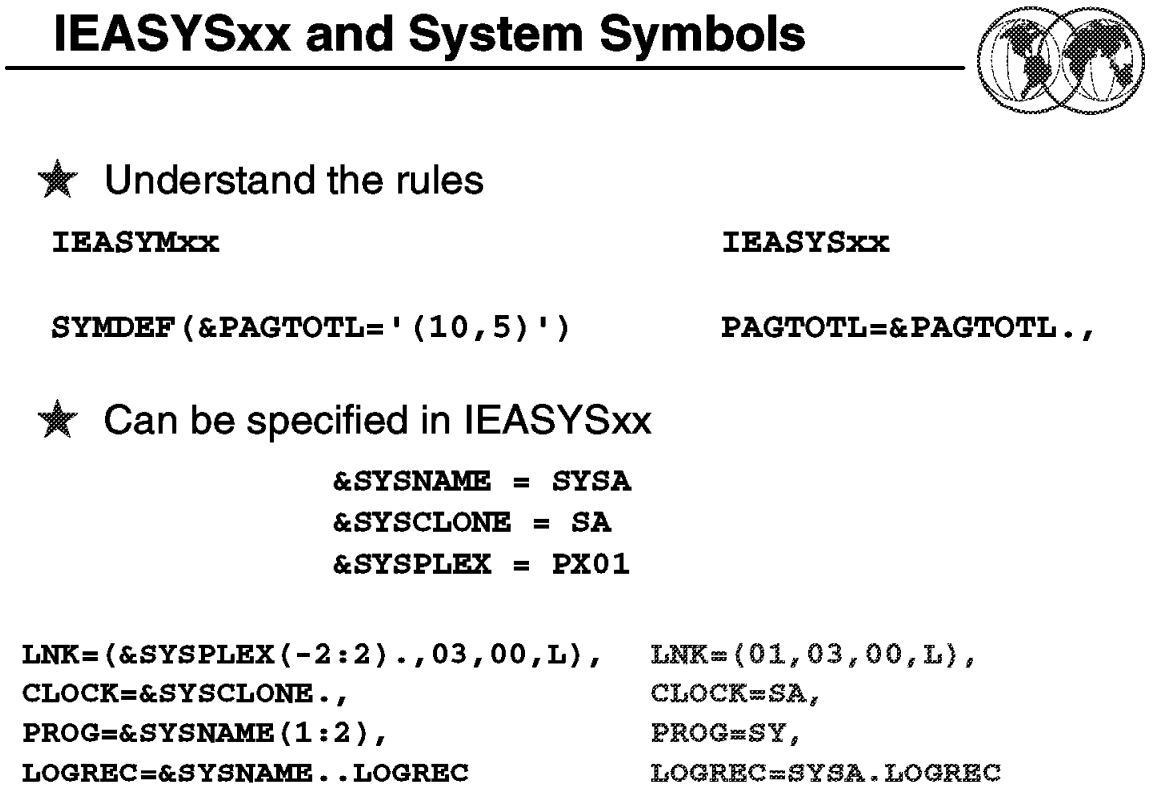


Figure 14. IEASYSxx and system symbols

1.3.9 IEASYSxx and 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.

For example, the following notation for IEASYMxx and IEASYSxx is valid, because the left and right parentheses both appear in the system symbol definition:

```
        IEASYMxx                        IEASYSxx
        --------                        --------
  SYMDEF(&PAGTOTL='(10,5)')          PAGTOTL=&PAGTOTL.,
```

The following notation is not valid, because the parentheses are split between the system symbol definition and the hard-coded definition in IEASYSxx:

```
        IEASYMxx                        IEASYSxx
        --------                        --------
   SYMDEF(&PAGTOTL='10,5)')           PAGTOTL=(&PAGTOTL.,
```

Example of using system symbols in IEASYSxx: Suppose the following system symbols have the values:

```
SC68 = SYSA
&SYSCLONE = SA
&SYSPLEX = PX01
```

Then assume that you want to do the following 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=SC68.LOGREC
```

The values of the parameters resolve to:

```
LNK=(01,03,00,L),
CLOCK=SA,
PROG=SY,
LOGREC=SYSA.LOGREC
```

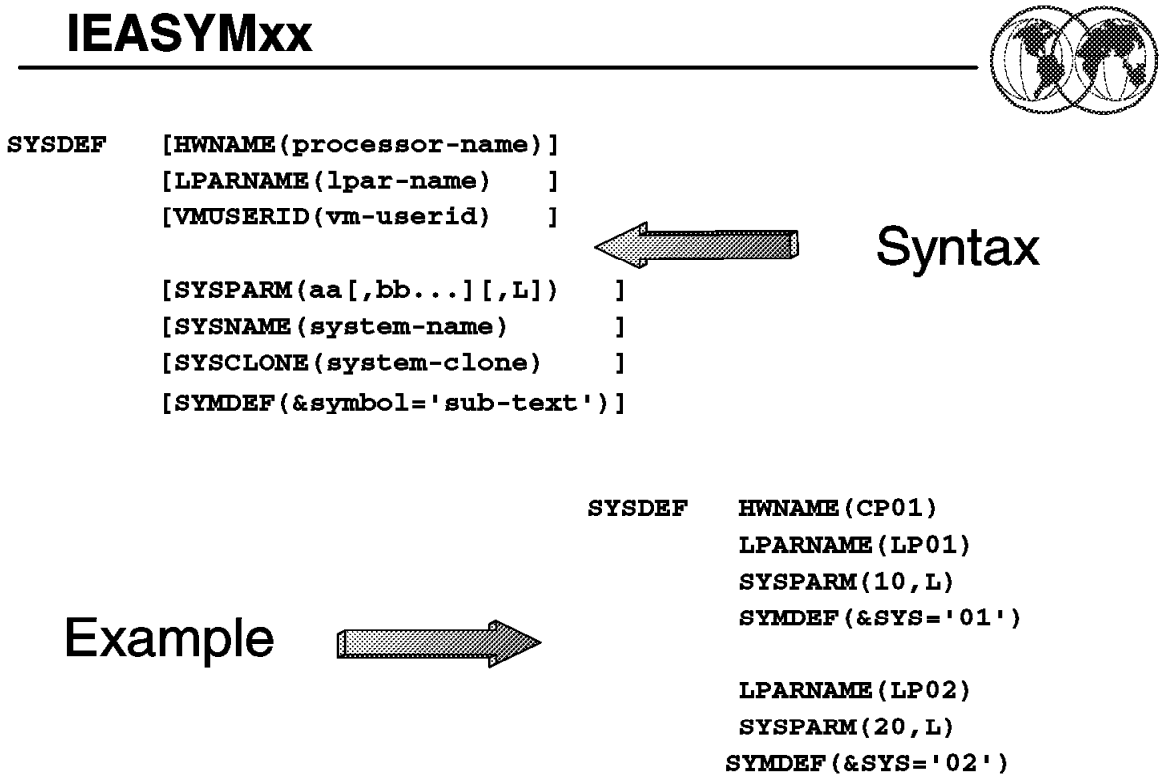


Figure 15. IEASYMxx parmlib member

1.3.10 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. The IEASYMxx parmlib member contains statements that do the following:

- 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 up to 99 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 no more than 99 of those system symbols apply to a particular system at any time during an IPL.

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 the next visual.

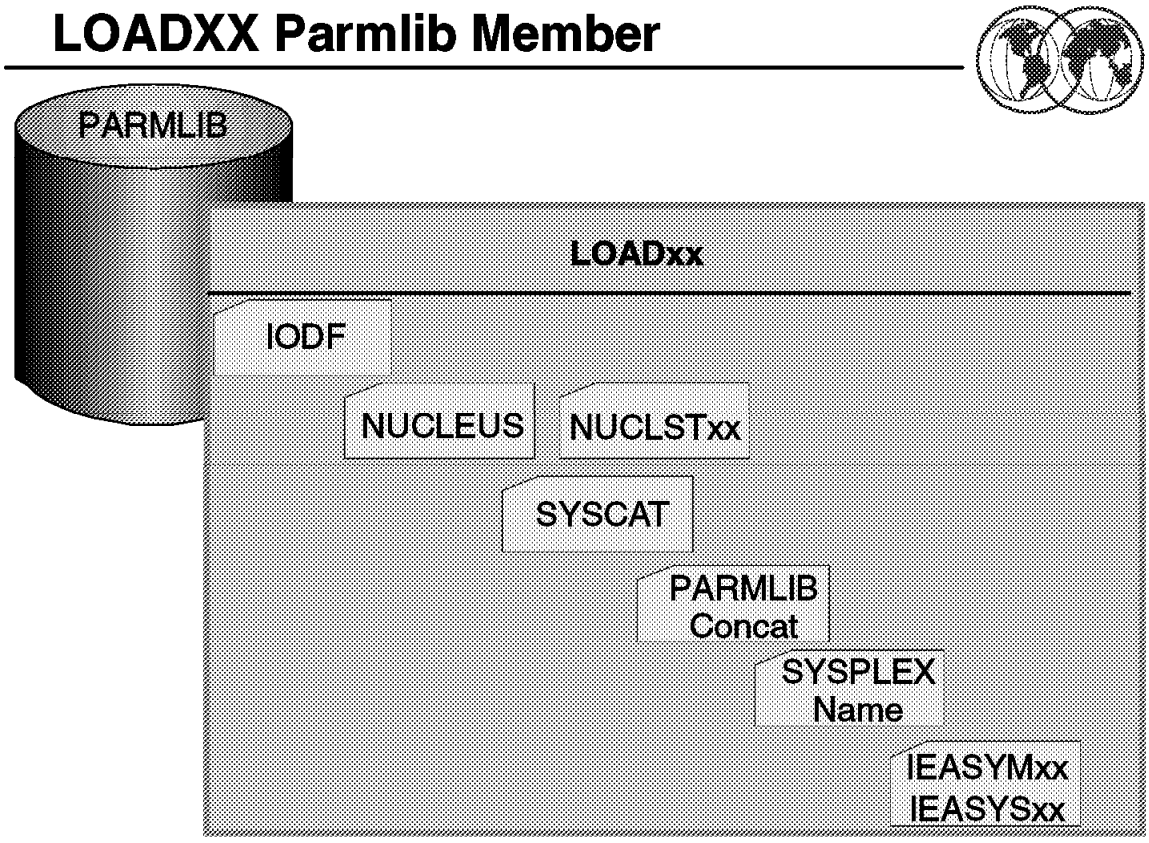


Figure 16. LOADxx parmlib member

1.3.11 LOADxx (system configuration data sets)

The system must have access to a LOADxx member which specifies:

- Information about your I/O configuration as specified by the IODFxx suffix to be used.
- An alternate nucleus ID.
- The NUCLSTxx member that you use to add and delete modules from the nucleus region at IPL-time.
- The name of 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.

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 *OS/390 MVS System Commands*, GC28-1781. If the operator does not select a LOADxx member on the system console, the system uses LOAD00.

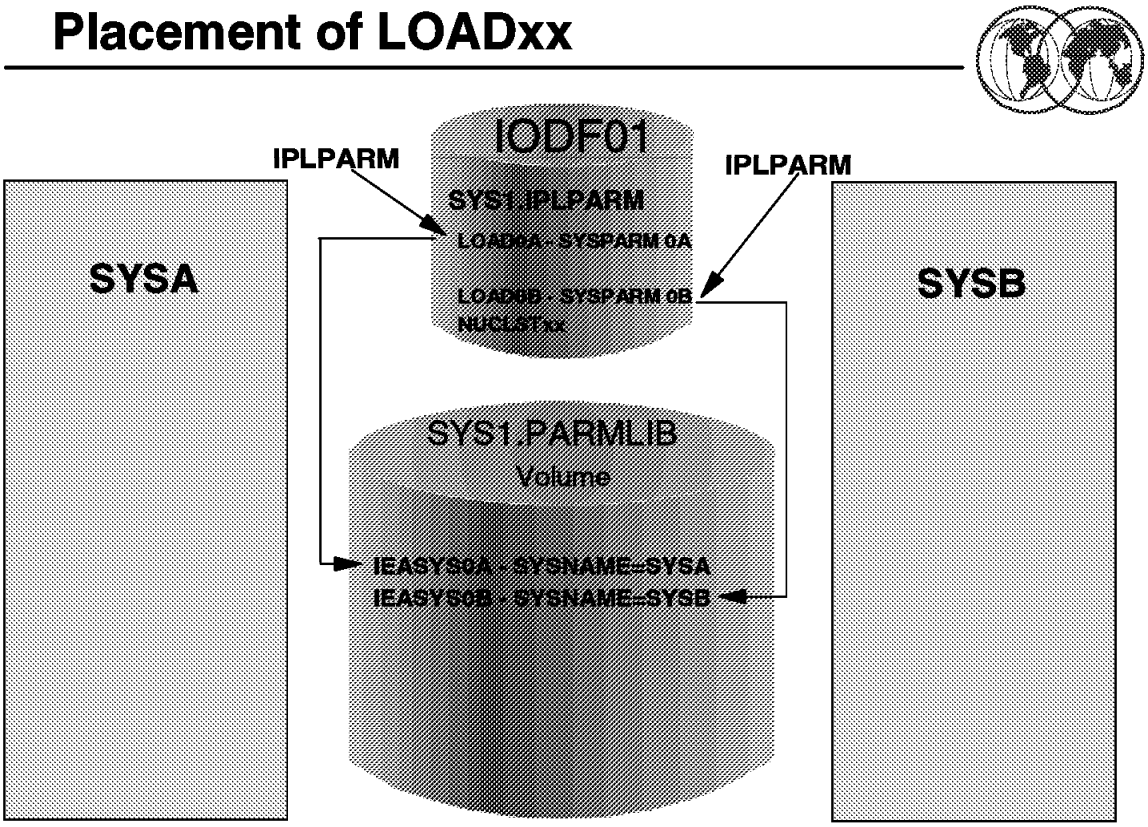


Figure 17. Placement of LOADxx

1.3.12 Placement of LOADxx

You can place the LOADxx member in one of the following system data sets:

SYSn.IPLPARM SYSn.IPLPARM is an optional partitioned data set that contains LOADxx members that point to I/O definition files (IODFs) that reside on the same volume as SYSn.IPLPARM. SYSn.IPLPARM is particularly useful in a multisystem environment, as shown in the visual.

You can use one SYSn.IPLPARM data set for each system. On each system, the SYSn.IPLPARM data set must be on the volume where the production IODF for that system resides.

For more information about the relationships between a SYSn.IPLPARM data set and its associated IODFs, see *OS/390 Planning for Installation*, GC28-1726.

Note: This system data set must reside on a direct access volume. This volume should contain only one SYSn.IPLPARM data set and its associated IODFs. (The character *n* is a single numeral, 0 through 9.)

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

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. If the LOADxx member specified on the LOAD parameter is not in the data set, the system loads a wait state.

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.

Member SYSCATLG of SYS1.NUCLEUS can contain a pointer to the master catalog. However, IBM recommends that you use the SYSCAT statement of the LOADxx member of SYS1.PARMLIB or SYSn.IPLPARM to identify the master catalog.

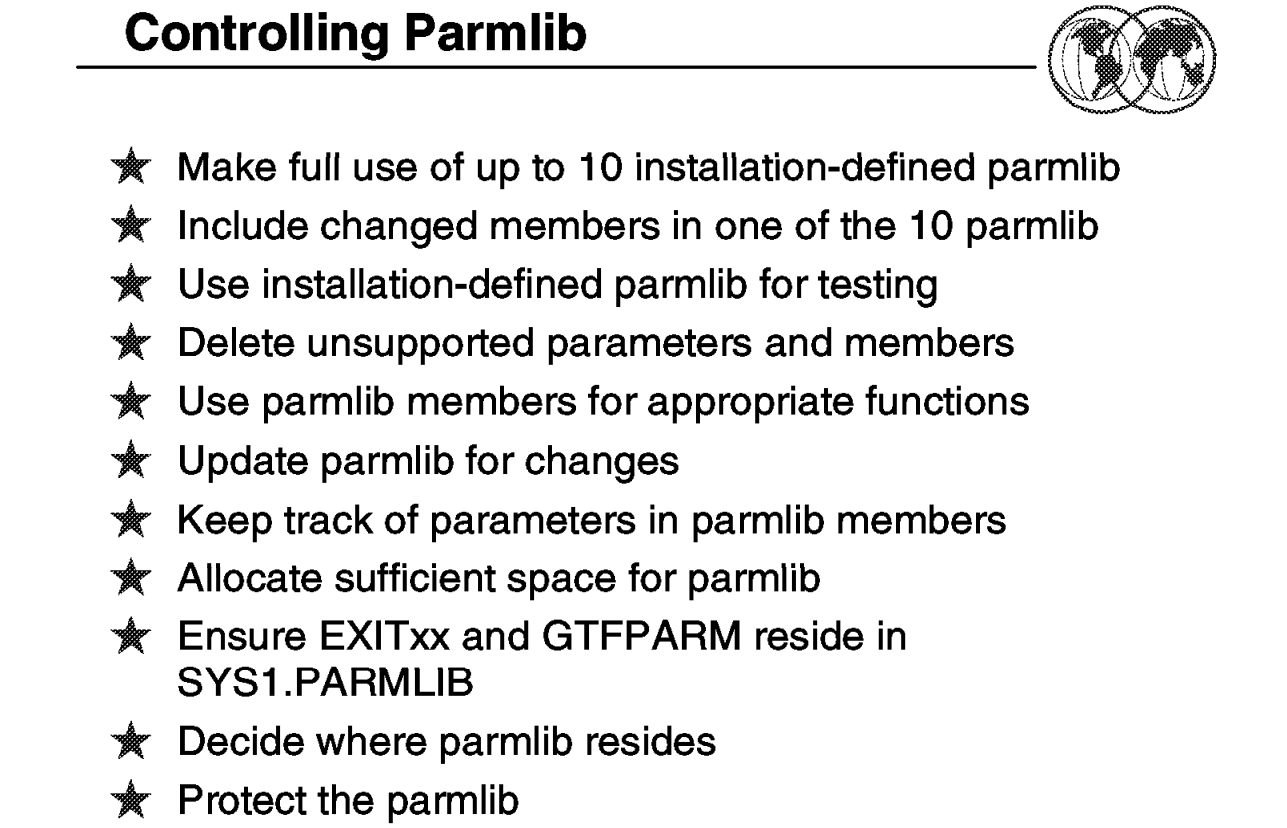


Figure 18. Controlling parmlib

1.3.13 How to control parmlib

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

- Use the ability to have up to 10 installation-defined parmlib data sets to 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 10 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 should probably remove the old parameters or build parmlib from scratch. This action will minimize 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 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 doesn't keep track of parmlib members and their parameters. 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, etc.). 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 blocksize of the data set. Then add a suitable growth factor (for example, 100–300 percent) to allow for future growth of alternate members. To recapture space occupied by deleted members, use the *compress* function of IEBCOPY. However, should the data set run out of space, you may 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 and GTFPARM reside in SYS1.PARMLIB since they can only be accessed from SYS1.PARMLIB.
- Decide which volume(s) and device(s) should hold the parmlib concatenation. The data set must be cataloged, unless it resides on SYSRES or its volume serial number is included on the PARMLIB statement in LOADxx. The data set could be placed on a slow or moderate speed device. For information about the placement of parmlib data sets and the IODF data set, see *OS/390 MVS System Data Sets Definition*, GC28-1782.
- Use a security product (like RACF) to protect the data sets. The purpose is to preserve system integrity by protecting the appendage member (IEAAPP00) and the authorized program facility members (IEAAPFxx and PROGxx) from user tampering.

1.3.13.1 General syntax rules for the creation of members

The following general syntax rules apply to the creation of most parmlib members. Exceptions to these rules are described under specific members. The general rules are:

- Logical record size is 80 bytes.
- Blocksize must be a multiple of 80.
- Any columns between 1 and 71 may contain data.
- Statements are entered in uppercase characters.
- Suffix member identifiers can be any combination of A to Z and 0 to 9, though some member identifiers may allow other characters.
- Columns 72 through 80 are ignored.
- 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 are most often indicated by using /* and */ as the delimiters in columns 1–71, for example:

```
/*comment*/
```

However, some parmlib members require other methods. Check specific parmlib members for information about specifying comments.

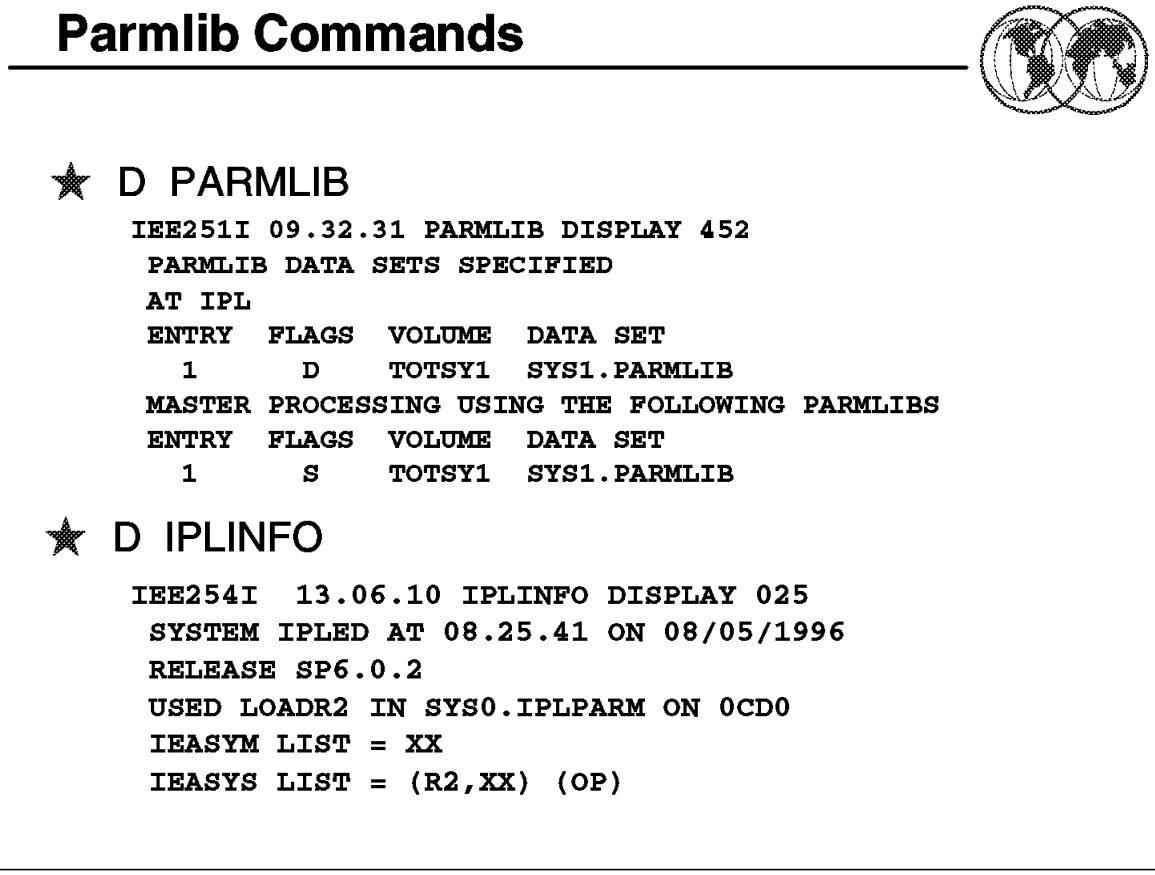


Figure 19. Parmlib commands

1.3.14 PARMLIB commands

The use of commands enable installations to display the current logical parmlib, display general IPL information, and change the current logical parmlib settings.

The commands are:

- DISPLAY PARMLIB

  This command displays the logical parmlib setup for the IPLed system. The output of this command includes the parmlib data set name(s) and volser(s) that was defined by LOADxx PARMLIB statement(s), and if used, MASTER JCL IEFPARM DD statements. When the errors option on the command is used, the display shows any parmlibs that were defined in LOADxx but were not found. This command is only valid before a SETLOAD command is issued. A sample output is shown in the visual.

- DISPLAY IPLINFO

  This command displays the general IPL information used by the system. The output includes the date and time of the IPL, release level, LOADxx information, and what IEASYSxx and IEASYMxx parmlib members were used. A sample output of the command is shown in the visual.

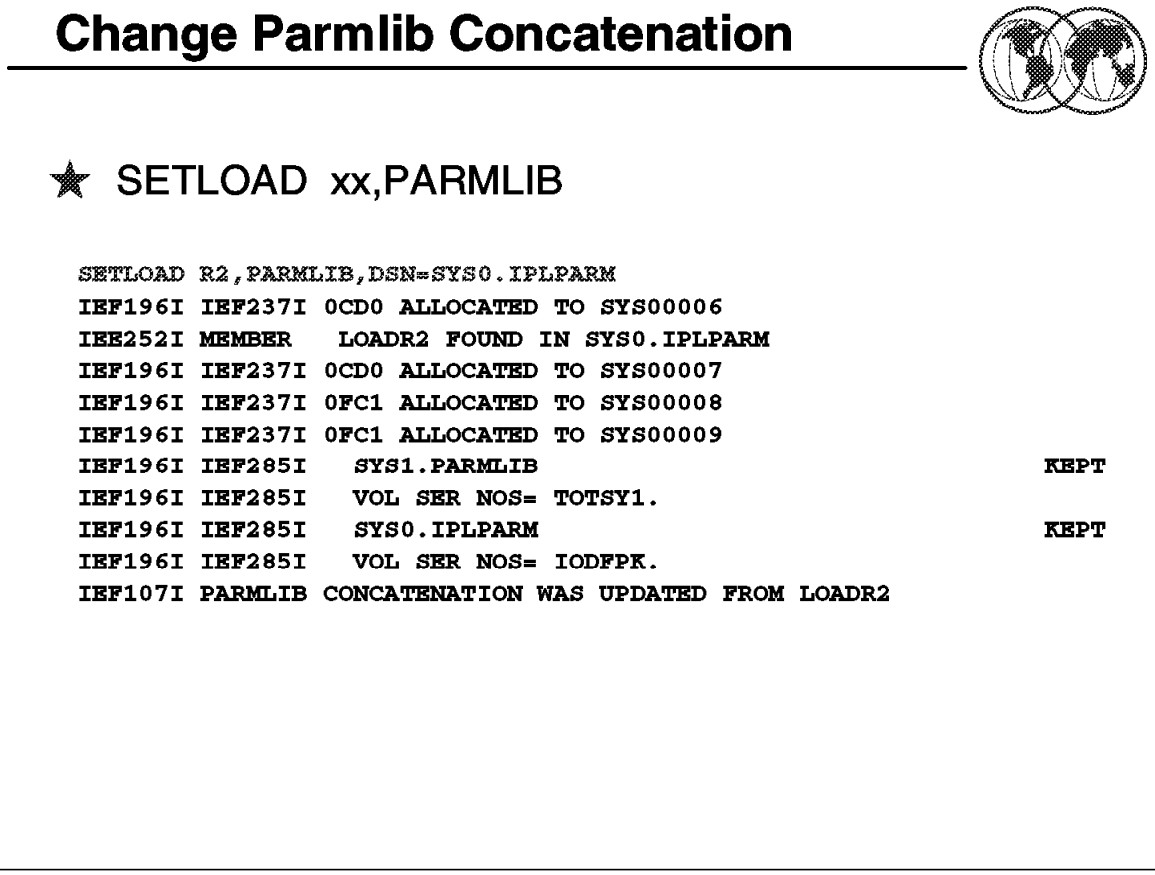


Figure 20. Parmlib commands

1.3.15 Parmlib commands

The following command (which is also shown in the visual) allows the installation to dynamically change a parmlib concatenation without having to IPL. A sample output of the SETLOAD command is shown in the visual.

```
SETLOAD R2,PARMLIB,DSN=SYS0.IPLPARM
```

D PARMLIB Command

★ After SETLOAD issued

```
D PARMLIB
IEE251I 09.36.52 PARMLIB DISPLAY 470
 PARMLIB DATA SETS SPECIFIED
 AT 09.34.11 ON 08/03/1996
 LOADR2 DATA SET=SYS0.IPLPARM
        VOLUME=CATALOG
 ENTRY  FLAGS  VOLUME  DATA SET
   1      S    CATALOG SYS0.IPLPARM
   2      S    CATALOG SYS1.OS390R2.PARMLIB
   3      S    CATALOG SYS1.PARMLIB
```

★ After SETLOAD issued and an IPL

```
D PARMLIB
IEE251I 13.10.57 PARMLIB DISPLAY 027
 PARMLIB DATA SETS SPECIFIED
 AT IPL
 ENTRY  FLAGS  VOLUME  DATA SET
   1      S    IODFPK  SYS0.IPLPARM
   2      S    TOTSY1  SYS1.OS390R2.PARMLIB
   3      D    TOTSY1  SYS1.PARMLIB
```

Figure 21. Parmlib commands output

1.3.16 Use of SETLOAD command

The sample output shown in visual is a result after the SETLOAD command was issued.

The second display shows the output of a DISPLAY PARMLIB command after an IPL had taken place using LOADR2. PARMLIB statements were specified in LOADR2. Note the difference in the FLAGS column shown in the visual.

The FLAGS describe how the parmlibs were specified:

- S denotes the LOADxx PARMLIB statement.
- D denotes the default (SYS1.PARMLIB).

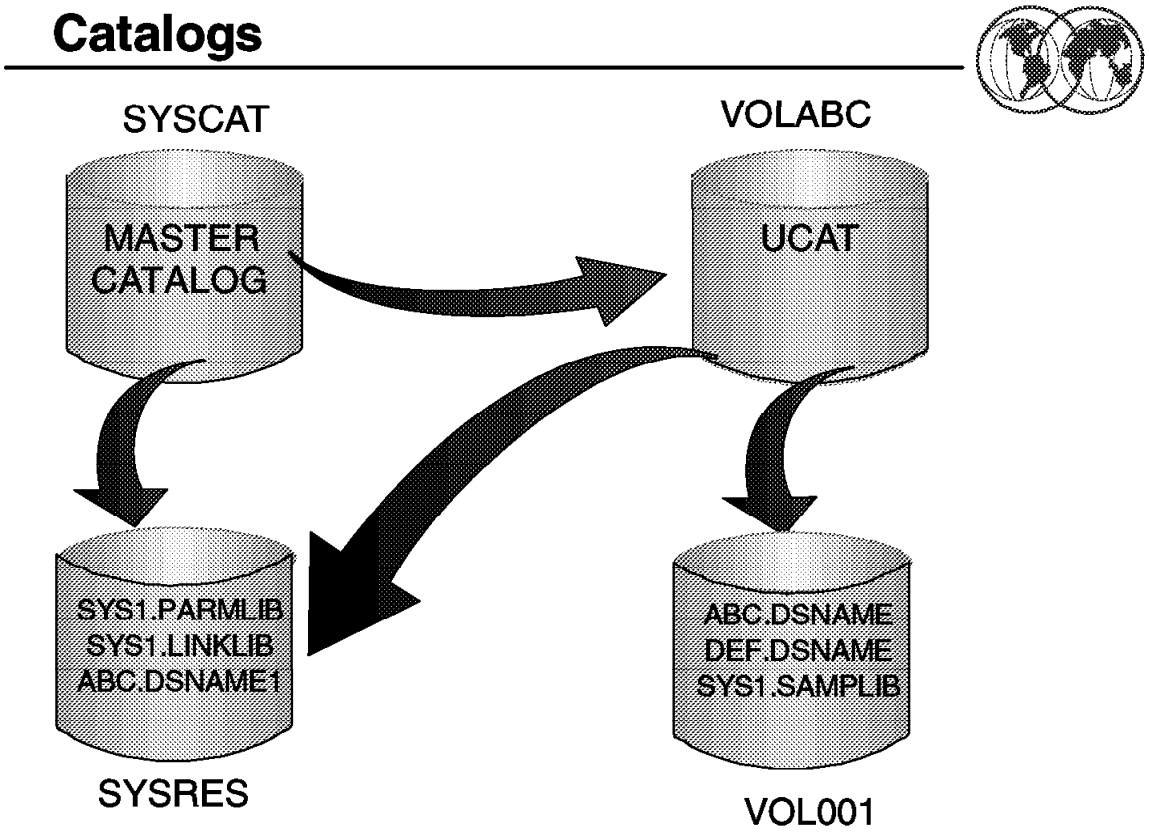


Figure 22. Catalogs

1.4 Catalogs

A catalog is a data set which contains information about other data sets. It provides users with the ability to locate a data set by name, without knowing where the data set resides. By cataloging data sets, your users will need to know less about your storage setup. Thus, data can be moved from one device to another, without requiring a change in JCL DD statements which refer to an existing data set.

Cataloging data sets also simplifies backup and recovery procedures. Catalogs are the central information point for VSAM data sets; all VSAM data sets must be cataloged. In addition, all SMS-managed data sets must be cataloged.

1.4.1 Using indirect catalog entries

Indirect cataloging, also known as indirect volume serial support, allows the system to dynamically resolve volume and device type information for non-VSAM, non-SMS managed data sets that reside on the system residence (IPL) volume when accessed through the catalog. This allows you to change the volume serial number or the device type of the system residence volume without also having to recatalog the non-VSAM data sets on that volume.

The extended indirect volume serial support that was introduced in OS/390 R3 allows catalog entries to be resolved using system symbols defined in the IEASYMxx parmlib member, so that indirect references can be made to one or more logical extensions to the system residence volume. Therefore,

you can have multiple levels of OS/390 data sets residing on multiple sets of volumes with different names and device types, and use them with the same master catalog.

Using indirect catalog entries, together with the indirect volume serial enhancements, allows you to share the master catalog among multiple images that use different volumes with different names for the system residence volumes and their extensions.

For more information on indirect volume serial support, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

1.4.2 Catalog management

Ensuring that your catalogs are effectively managed is therefore a crucial aspect of DASD management. Some of the most common catalog management tasks that a system programmer may have are:

- Defining and maintaining the master catalog
- Defining the alias and user catalog
- Protecting the catalogs
- Cleaning up catalogs
- Backing up catalogs and performing catalog recovery

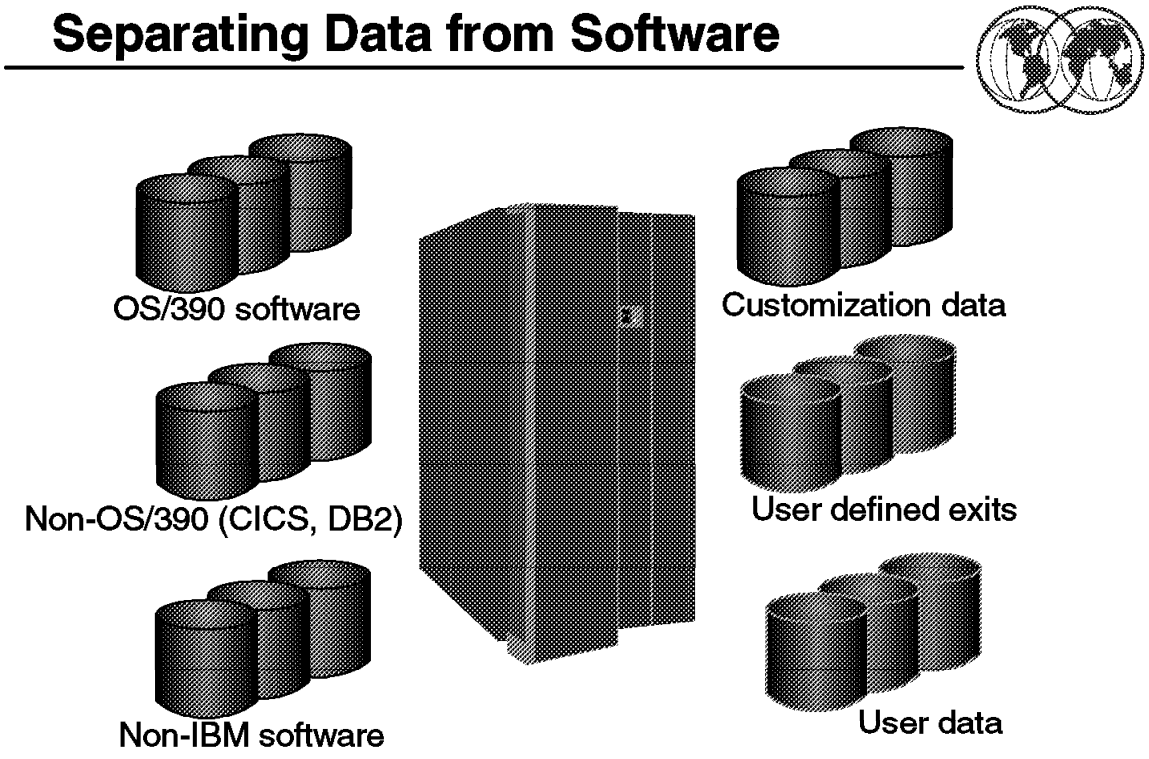


Figure 23. Separating data from software

1.5 Separating data from software

When you separate your data from your system software, you eliminate many tasks that would otherwise need to be performed each time you upgrade or replace your system software. One effective way to achieve this is to use dedicated pools of DASD volumes for each.

The kinds of data you should separate from OS/390 software are:

- Customization data, including most system control files
- Non-IBM software
- IBM non-OS/390 products, for example CICS and DB2
- User-defined exits
- User data

Your goal is to make it easier to replace the volumes that contain OS/390 software, which allows you to keep the other software and data you will need to use with OS/390 across migrations.

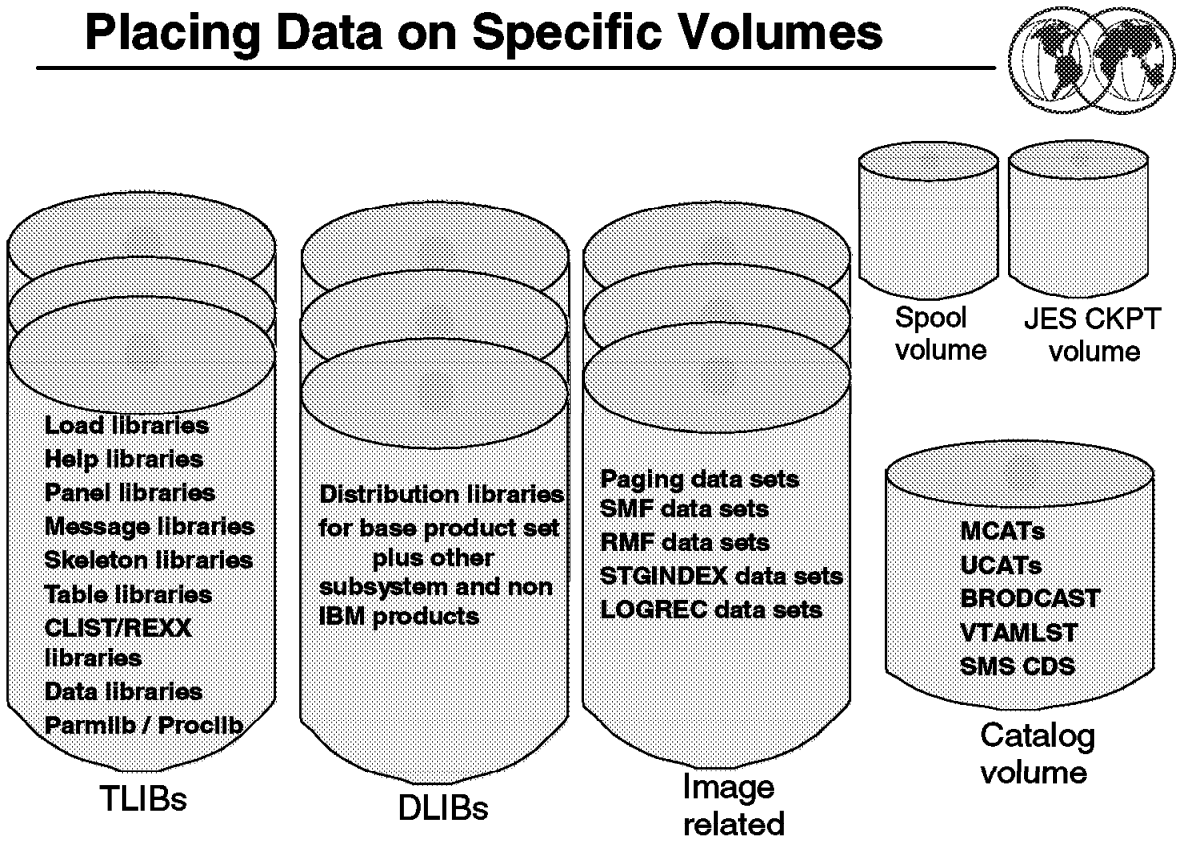


Figure 24. Data set placement

1.6 Placing data sets on specific volumes

Some SYSRES volume types, such as the 3390-3, are not big enough to hold all the target libraries for an OS/390 system. Therefore, you have to move some data sets to SYSRES logical extension volumes (or overflow volumes). The considerations you should take into account for placing data sets on specific volumes are:

- Your ability to use a system (or subsystem) replacement
- Data set and system availability
- System performance
- System cloning and servicing techniques
- Sysplex/multisystem operations
- Sharing data sets
- Backup and recovery
- Disaster recovery

Based on these considerations, you should determine which data sets to place on each volume based on data set type, not based on element, feature, or product. There are basically five types of data sets, which each type can be placed on a separate (logical) volume:

- SMP/E global-shared data sets

- Target libraries (TLIBs) for product sets
- Distribution libraries (DLIBs) for product sets
- Image-related data sets
- Cluster-related data sets

1.6.1 SMP/E global-shared data sets

These data sets should be placed on a volume shared by all systems in the complex that need SMP/E global information. Sharing these data sets will provide an ease of backup and recovery. The recommended types of data sets for this volume are:

- SMP/E global CSI
- SMPPTS
- SMP/E global logs (SMPLOG and SMPLOGA)

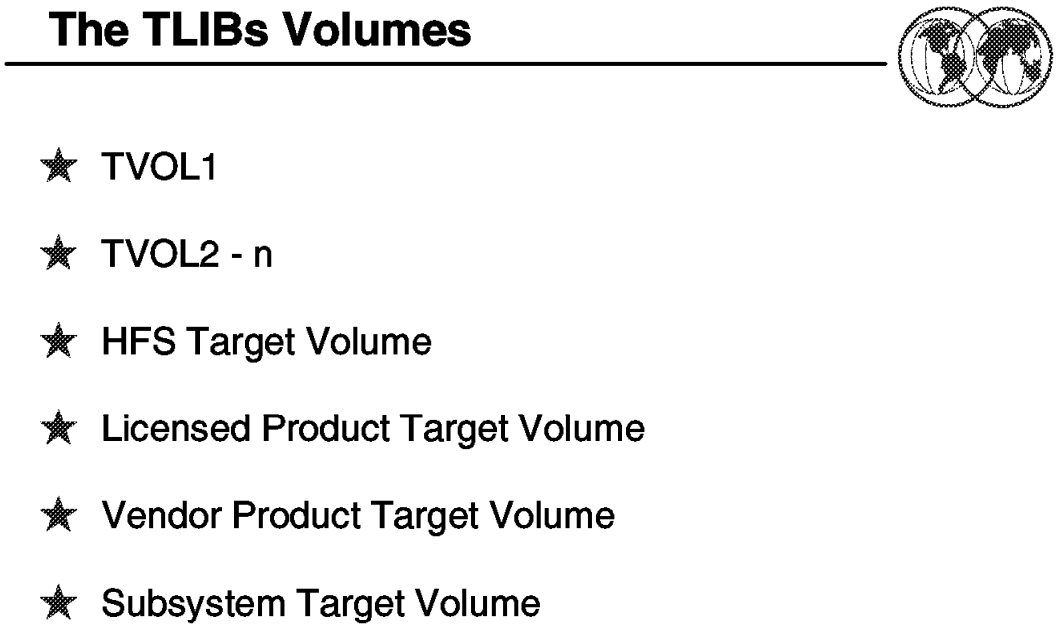


Figure 25. The TLIBs volumes

1.6.2 Target libraries (TLIBs) for product sets

These target libraries spread across several volumes, from primary to secondary volumes:

- Primary target volume (TVOL1)

  The TVOL1 is the first target library volume and the system residence (IPL) volume; it contains many of the OS/390 target libraries. Make sure you leave enough free space to allow for future growth. The recommended types of TVOL1 data sets are:

  - *Load libraries* which are data sets containing load modules.
  - *Change migration libraries* that are used during migration from one level of software to another.
  - *Help libraries* are data sets that contain help information.
  - *Panel libraries* are data sets that contain ISPF panels.
  - *Message libraries* are OS/390 libraries that contain ISPF messages and MMS source messages.
  - *Skeleton libraries* are OS/390 libraries that contain ISPF skeletons.
  - *Table libraries* are OS/390 libraries that contain ISPF tables.
  - *Fixed-block CLIST and EXEC libraries* are data sets that contain CLIST and REXX EXECs.
  - *Data libraries* are OS/390 libraries that contains data parts which include workstation information, header files, or other various types of data.
  - *SMP/E-managed PARMLIB* which is the data set that is pointed to by the PARMLIB DDDEF, which will be used to store parmlib members supplied by products you install.

- *SMP/E-managed PROCLIB* which is the data set that is pointed to by the PROCLIB DDDEF, which will be used to store JCL procedures supplied by products you install.

- Secondary target volumes (TVOL2 – TVOLn)

  The TVOL2 through TVOLn are volumes used for data sets that do not fit on TVOL1. They are for the OS/390 product set and the recommended types of TVOL2 through TVOLn data sets are:

  - *Fixed-block CLIST and EXEC libraries* are used only if variable-block CLIST and EXEC libraries were used on TVOL1.
  - *Sample and JCL libraries* are OS/390 libraries that contain samples, header files, and JCL jobs.
  - *Source libraries* are OS/390 libraries that contain source code.
  - *Macro libraries* are OS/390 libraries that contain assembler macros, header files, and other information identified in SMP/E as the element type MACRO.
  - *Workstation libraries* which can be combined with the data libraries.
  - *Softcopy libraries into which SMP/E installs* are OS/390 libraries that contain books, bookshelves, and bookindexes.
  - *Font and printing libraries* are OS/390 libraries that contain fonts and data sets required for printing.
  - *Flat files that SMP/E cannot manage* include interface repositories and so forth, excluding books.
  - *SMP/E target CSI*
  - *SMP/E target data sets* including SMPLTS, SMPMTS, SMPSTS, and SMPSCDS.
  - *User catalog for the SMP/E target CSI and MVS-supplied data sets*

- HFS target volume

  The recommended types of data sets (filesystems) for this volume are:

  - HFS data sets for OS/390 elements or features that install into an HFS.
  - Any non-OS/390 HFS data set, except those containing customization data.

- Licensed product target volume

  The libraries on this volume consist of the licensed product set that you might not have in a system-replacement order and you want to keep separate. The recommended types of data sets for this volume are:

  - Licensed program target libraries.
  - SMP/E target CSI.
  - SMP/E target data sets: SMPLTS, SMPMTS, SMPSTS, and SMPSCDS.
  - User catalog where all the licensed program libraries and the SMP/E target CSI are cataloged.

- Vendor product target volume

  The libraries on this volume consist of the vendor product set that you might not have in a system-replacement order and you want to keep separate. The recommended types of data sets for this volume are:

  - Vendor target libraries.
  - SMP/E target CSI.
  - SMP/E target data sets: SMPLTS, SMPMTS, SMPSTS, and SMPSCDS,
  - User catalog where all the vendor product target libraries including the SMP/E target CSI are cataloged.

- Subsystem target volume

  The libraries on this volume consist of the subsystem product sets (for example, CISCS, DB2, IMS, or NCP). The recommended types of data sets for this volume are:

  - Subsystem target libraries.
  - Alternate subsystem SMP/E global CSI, if applicable.
  - SMP/E target CSI.
  - SMP/E target data sets: SMPLTS, SMPMTS, SMPSTS, and SMPSCDS.
  - User catalog where the subsystem targets libraries including SMP/E CSIs are cataloged.

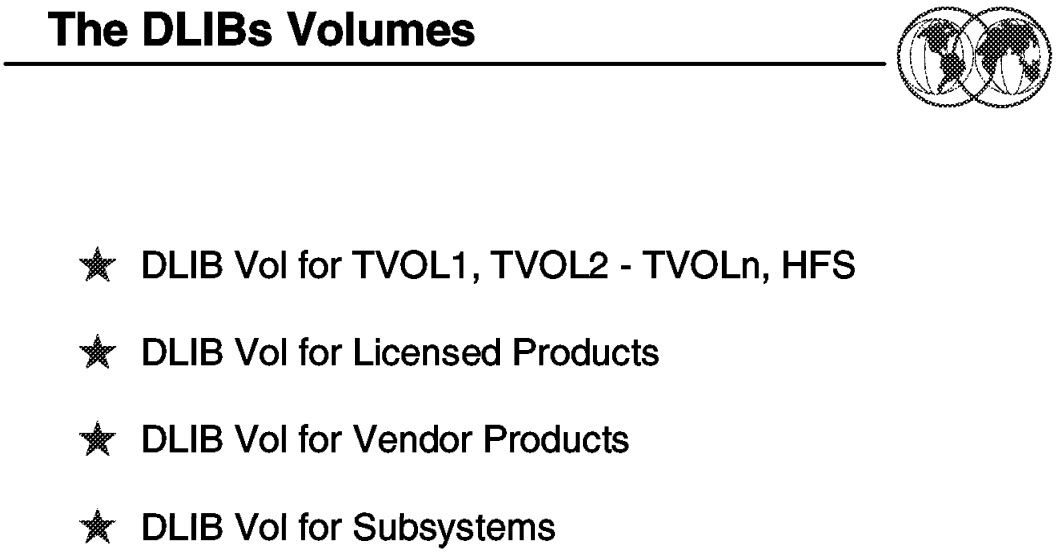


Figure 26. The DLIBs volumes

1.6.3 Distribution libraries (DLIBs) for product sets

You should place data sets on the DLIB volumes wherever they fit. However, keep in mind how other systems will use the distribution libraries when you are deciding where to place them. There are cases where you don't want or need a set of distribution libraries available on certain packs, for example having multiple target zones connect to a DLIB zone. The DLIB volumes like the target volumes can be divided into primary and secondary volumes:

- DLIB volumes for TVOL1, TVOL2 through TVOLn, and HFS

  These distribution libraries are the ones that are placed by ServerPac for your OS/390 product sets. By keeping the distribution libraries on the same volumes it will be easier to avoid overlaying data sets.

- DLIB volumes for licensed products

  These are the distribution libraries that correspond to the target libraries for the licensed product set. These are data sets that would not be overlaid in a system replacement.

- DLIB volume for vendor products

  These are the distribution libraries that correspond to the target libraries for the vendor product set. These are data sets that would not be overlaid in a system replacement.

- DLIB volumes for subsystems

  These distribution libraries are the ones that are placed by ServerPac for subsystem product sets. By keeping the distribution libraries on the same volumes it will be easier to avoid overlaying data sets.

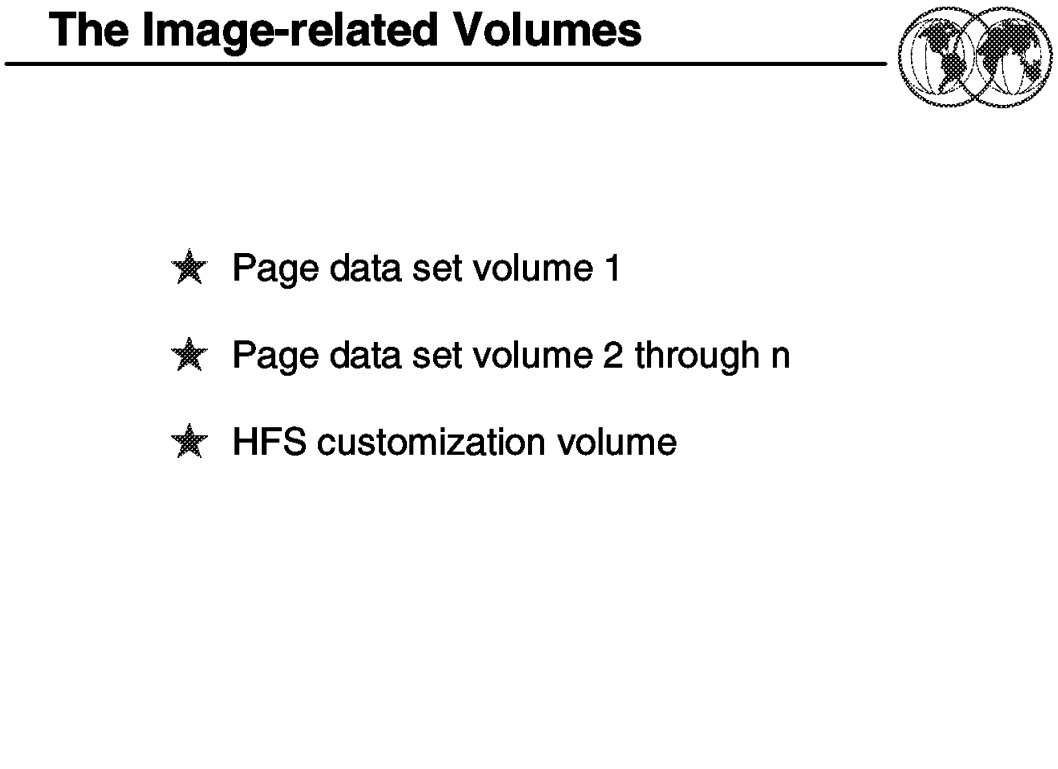


Figure 27. The image-related volumes

1.6.4 Image-related data sets

These data sets contain non-shareable system image information. Although the recommendation is that they be put on separate volumes, if DASD space is scarce you can combine them at the expense of performance or availability, or both. The image-related data sets are placed in the following recommended volumes:

- Page data sets volume 1

  The recommended types of data sets for this volume are:

  - PLPA (one-cylinder allocation)
  - COMMON

    Note: Unless your system is central-storage constrained, and has significant PLPA paging activity, there is little or no performance impact to combining the PLPA and COMMON page data sets. The PLPA data set should be allocated first, as a one-cylinder data set, with the COMMON data set allocated second, immediately following the PLPA data set on the same volume. The size of the COMMON data set should be large enough to contain both PLPA and COMMOND pages.

    This causes the vast majority of PLPA pages to be written to the COMMOND page data set during IPL. This allows the operating system to use the chained CCWs within a single data set and improves performance when both data sets are on the same volume.

    The message (ILR005E PLPA PAGE DATA SET FULL, OVERFLOWING TO COMMON DATA SET) during IPL can be ignored when the PLPA and COMMON page data sets are on the same volume.

- Page data set volumes 2 to n

The recommended types of data sets for these volumes are:

- Local
- SMF
- RMF reporting
- STGINDEX data set (if used)
- Image-related LOGREC data set (if used)

- HFS customization volume

  This is an installation maintained volume that contain data sets that will not be overlaid by system replacement. This volume is separate from the HFS target volume because it contains unshareable HFS files that will generally need to be mounted MODE(RDWR). The recommended types of data sets for this volume are:

  - HFS data sets that must be in write mode (for instance, /tmp, /etc, /dev, /u) and contain customized information.
  - User catalog where the HFS data sets are cataloged.

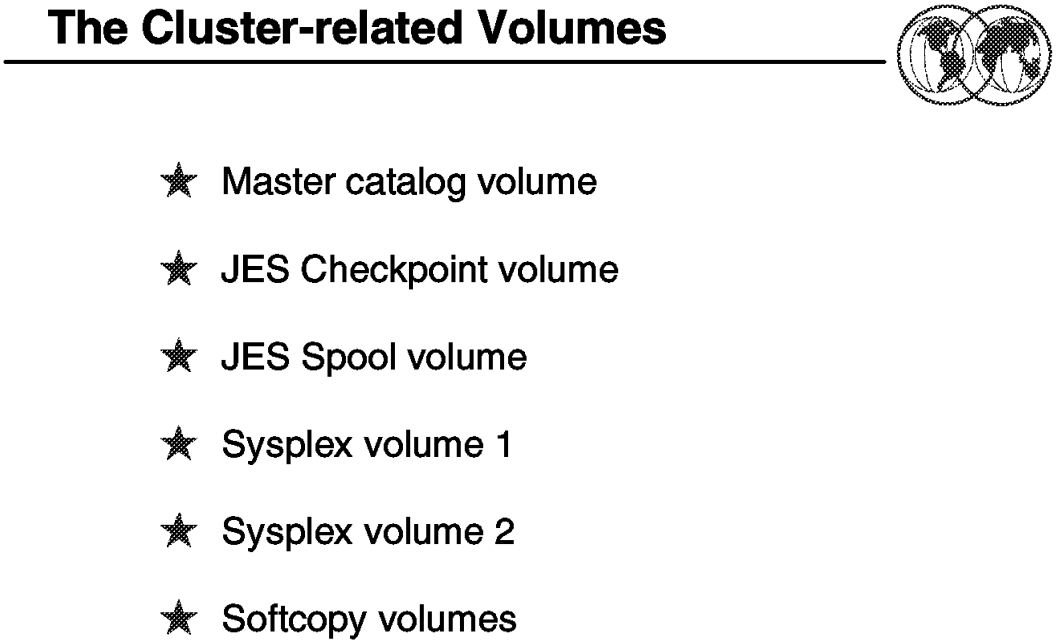


Figure 28. The cluster-related volumes

1.6.5 Cluster-related data sets

These are shareable data sets used in a multisystem environment. Cluster-related data sets should use system symbolics in their names for easier maintainability. While all cluster-related data sets can be combined on the same volume, it is usually preferable to separate certain data sets from others for performance or availability reasons. You can group the cluster-related data sets into the following volumes:

- Master catalog volume

  The recommended types of data sets for this volume are:

  - Master Catalog
  - BROADCAST data set
  - Customer parmlib concatenation (not the SMP/E DDDEFed PARMLIB)
  - Customer proclib concatenation (not the SMP/E DDDEFed PROCLIB)
  - UADS data set (if used)
  - VTAMLST data set
  - SMS control data sets (ACDS, SCDS, and COMMDS), HSM, RMM, and so forth
  - APPC VSAM data set
  - System control files (TCPI/P configuration and so forth)
  - Primary RACF database
  - IODF data set

  - SYS0.IPLPARM
  - UCATs
  - SYS1.DDIR sysplex dump directory data set
  - DAE data set
- JES checkpoint volume

  For maximum performance and reduced contention, place the primary JES checkpoint data set on its own dedicated volume. The JES checkpoint primary data set may be on a Coupling Facility.
- JES spool volume

  You can place the JES duplex checkpoint data set together with the JES spool to reduce I/O and improve performance. Having the duplex data set on the same volume as one of the JES spool data sets will take advantage of the fact that JES checks, during every write to the JES spool, whether a checkpoint data set resides on the same volume.
- Sysplex-related volume 1

  The recommended types of data sets for this volume are:
  - SYSPLEX primary
  - CFRM alternate
  - ARM primary
  - WLM primary
  - LOGR primary

    Note: The CFRM primary and SYSPLEX primary should be on different volumes attached to different control units. All other primary couple data sets can reside on the same volume, and all other alternate couple data sets can reside on a different volume.
- Sysplex-related volume 2

  The recommended types of data sets for this volume are:
  - SYSPLEX alternate
  - CFRM primary
  - ARM alternate
  - WLM alternate
  - LOGR alternate
  - Secondary RACF database
- Softcopy volume

  This volume holds softcopy books and related data sets. The recommended types of data sets for this volume are:
  - Books
  - Bookshelves
  - Bookindexes

Many volumes on your system will contain data sets that are not supplied by ServerPac. Keeping such volumes separate from those that ServerPac will replace, or that you will replace when migrating the new system to other system images, makes it easier to prevent overlaying data sets that you want to keep.

For more information on data sets placement, see *OS/390 Planning for Installation*, GC28-1726.

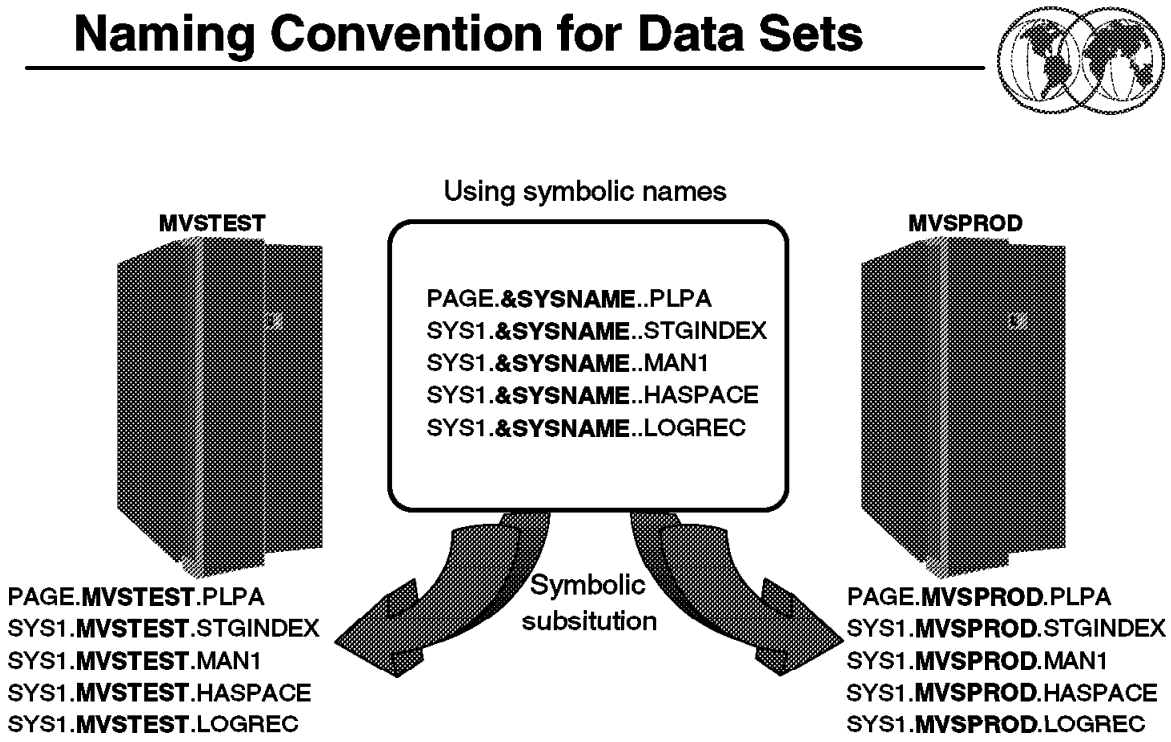


Figure 29. Naming conventions for data sets

1.7 Choosing a naming convention for data sets

Choosing the right naming conventions for system software data sets can save you considerable time during installation and migration.

Some data sets are associated with only one system in a multisystem environment. Choose names for these data sets that reflect the name of the specific system. Names of system operational data sets, such as page and swap data sets, should include the system name. You can accomplish this using the IBM-supplied system symbol *SC68*.

Using symbolic substitution involves carefully establishing naming conventions for data sets, such as parmlib and proclib members, system images, HCD definition, and so forth. Remember that once your system goes into production with a set of naming conventions, you cannot easily change them.

IBM has been removing the level-dependent qualifier (such as V1R1M0) from default data set names for OS/390 elements in order to ease your migration to a new OS/390 release. This preserves much of your existing JCL, system procedures, parmlib and other system control file contents, and security system definition across upgrades, saving you time and reducing the opportunity for error because updates will be limited to just the data sets in which the low-level qualifiers were changed.

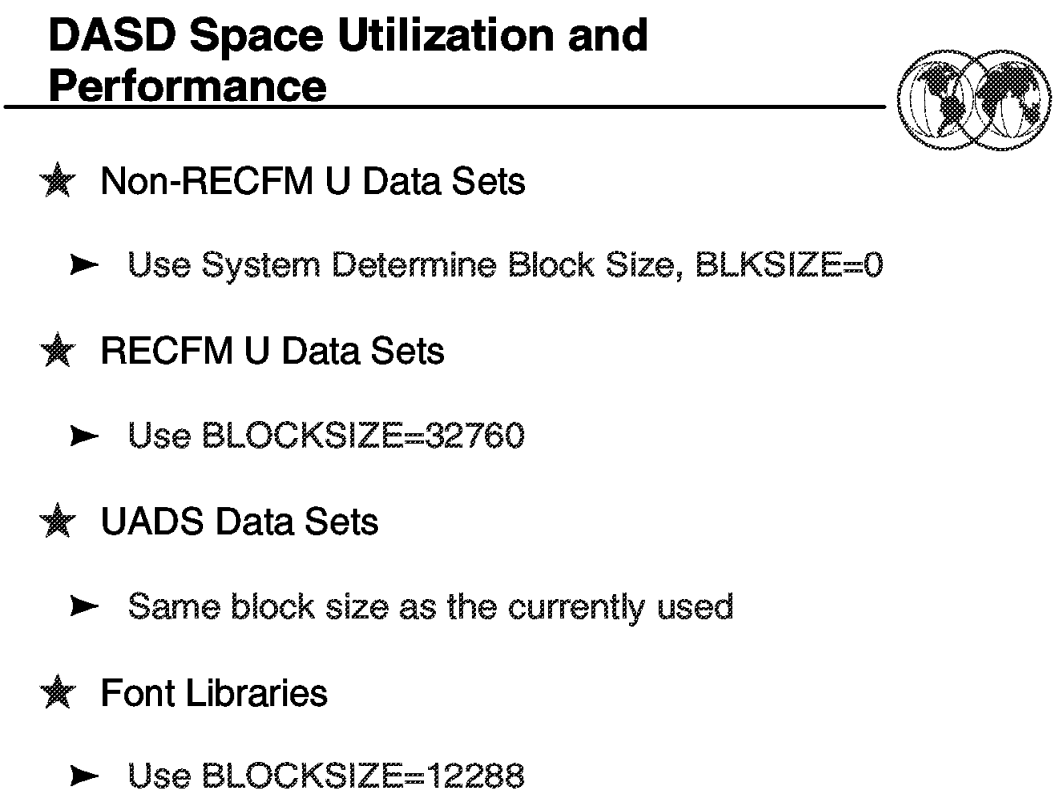


Figure 30. DASD space utilization and performance

1.8 DASD space utilization and performance

The space required by system software data sets, except for PDSE data sets, is affected by the block sizes you choose for those data sets. Generally, data sets with larger block sizes use less space to store the same data than those with smaller block sizes.

The exception to this general rule are fixed block (FB) record format data sets. They should not be allocated with block sizes larger than half the track length of the DASD they are allocated on. Doing so will cause considerable DASD space to be wasted, because current DASD track lengths are less than twice the maximum block size of 32760 bytes.

Generally, system-determined block sizes (SDB) are the best choice for block size for fixed block (FB), variable blocked (VB), and variable block spanned (VBS) record format data sets. You should use SDB for all system software data sets with these record formats except those for which IBM specifically recommends other block sizes. One way to do this is by specifying BLKSIZE=0 in the DCB parameter of a DD statement when allocating new data sets using JCL. For more information, see *OS/390 MVS JCL Reference*, GC28-1757.

Data sets with undefined (U) record formats do not follow the same rules as those with other record formats. In particular, most load libraries in partitioned data sets (not PDSEs) will require less space and offer better performance at increasing block sizes right up to the block size limit of 32760 bytes. The reason is because the program management binder, linkage editor, and IEBCOPY's COPYMOD command use the data set block size only to set the maximum block length they will use.

Allocate all load libraries using a block size of 32760 bytes unless you plan to move your system software data sets from the device types on which they were originally allocated to device types with

shorter track lengths, or plan to move them between device types having different track lengths without using IEBCOPY COPYMOD.

For most efficient use of DASD, it is recommended that you allocate OS/390 data sets using the following block sizes:

- Use the SDB for most of non-RECFM U data sets.
- For RECFM U data sets, use BLKSIZE=32760.
- For UADS data sets for TSO/E, use the same block size you currently use to allocate a new one. Do not use SDB, as this will result in very poor DASD space utilization.
- You should not use the SDB for font libraries too. The correct block size for the font libraries is 12288.

For more information on the usage of SDB, see *DFSMS/MVS Using Data Sets*, SC26-4922.

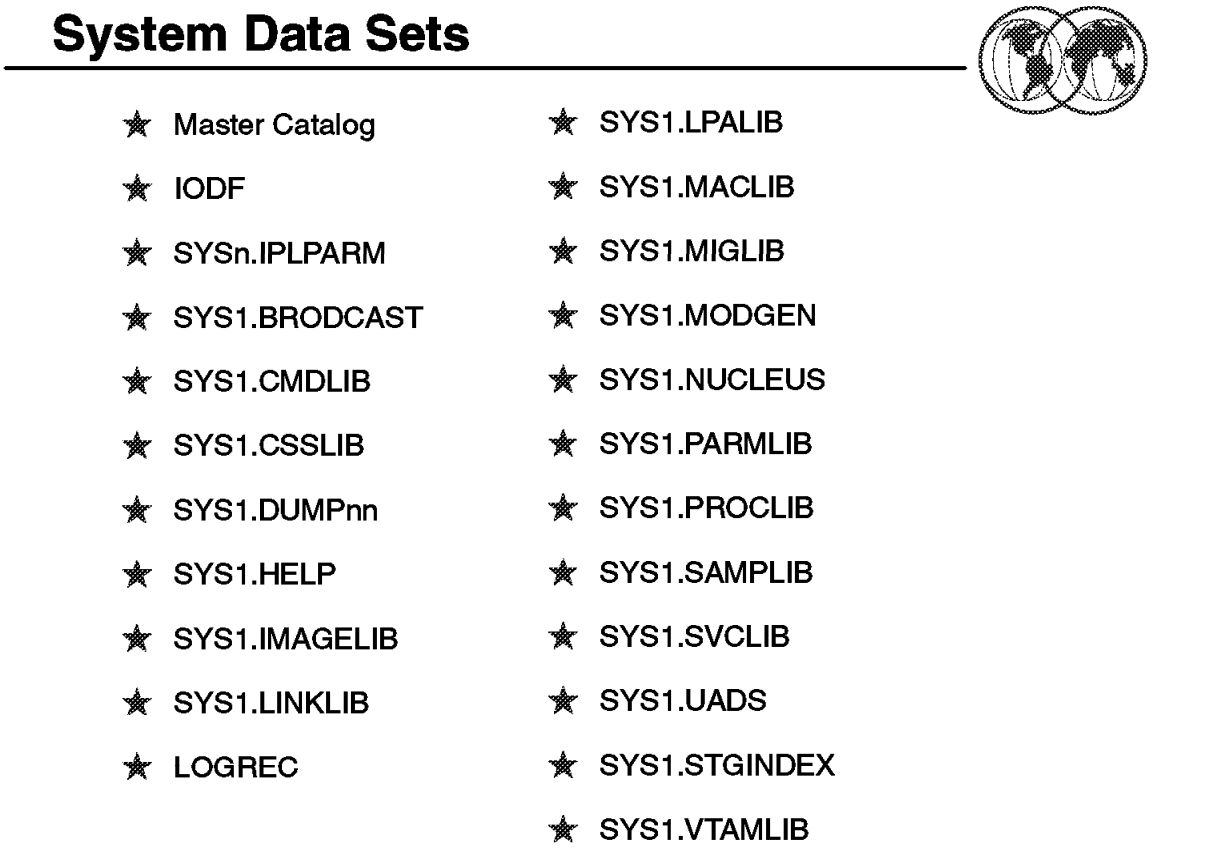


Figure 31. System data sets

1.9 System data sets

Before you install OS/390, you must select and define the system data sets that you need.

Before you include components from the distribution libraries (DLIBs) and user-defined data sets in the system, use JCL and access method services to define the system data sets.

This visual contains some of the common system data sets. For a complete list of system data sets, see *OS/390 MVS System Data Sets Definition*, GC28-1782.

A description of the data sets can be found in A.3, “System data sets” on page 291.

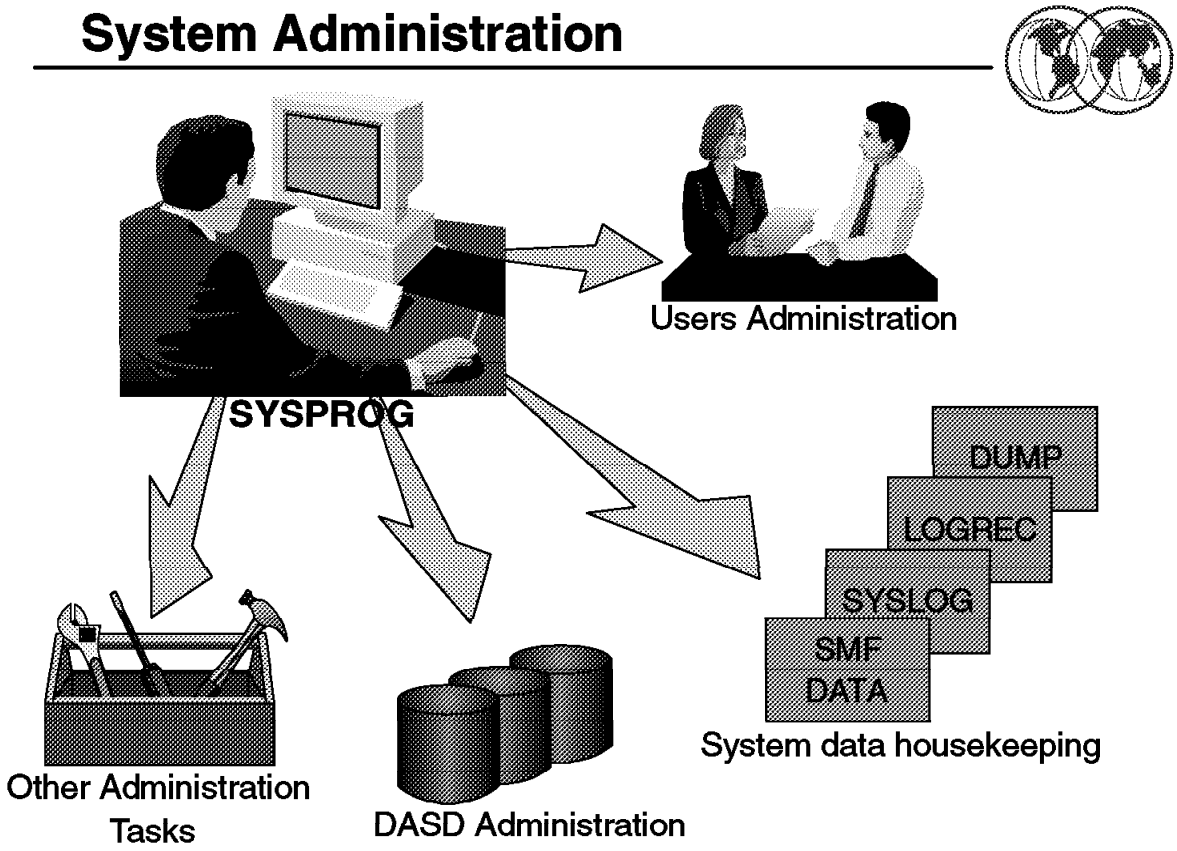


Figure 32. System administration

1.10 System administration tasks

Once you have the system up and running, you will spend most of your time maintaining the well-being of the system at the same time providing technical support to your users. The following sections in this chapter discuss some of the administrative tasks performed by a system programmer.

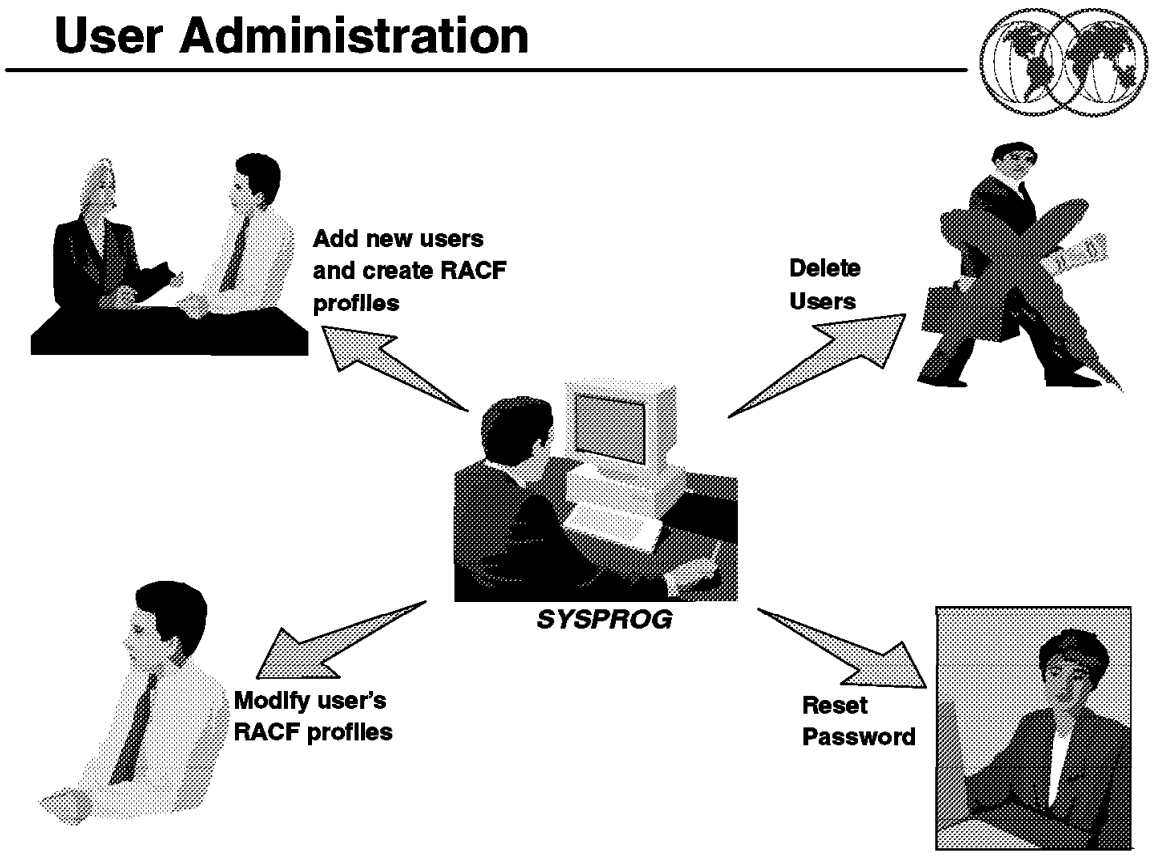


Figure 33. User administration

1.11 User administration

A system is never complete without users using it. Most of the time, you will find yourself having to perform certain user administration tasks. The next few sections provide examples of typical processes used when adding, deleting, or administrating users.

Some of the common user administrative tasks discussed in this section are:

- Add new users
- Create RACF profiles for new users
- Modify RACF profiles for existing users
- Reset password
- Delete users and their RACF profiles

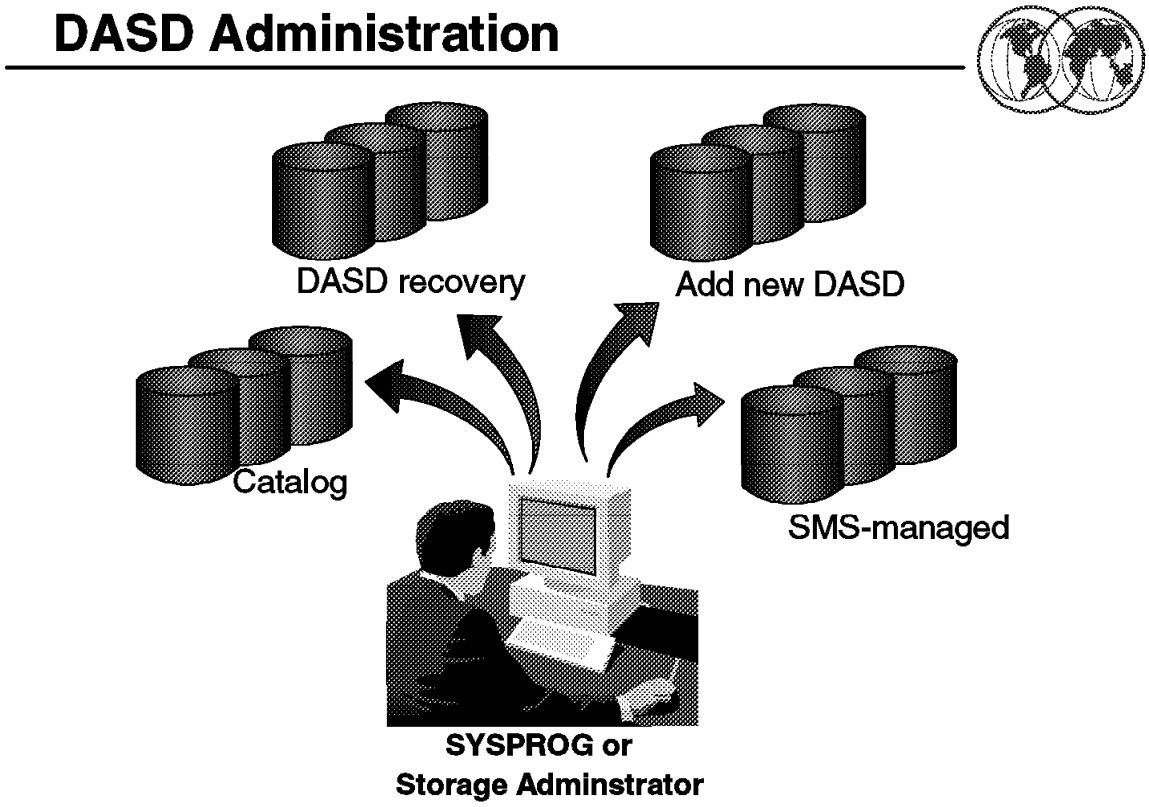


Figure 34. DASD administration

1.12 DASD administration

On a larger installation, you might have a team of storage administrators to take care of all DASD administration tasks and you as a system programmer may only be required to help with obscure problems and exit routines. In that environment, you may never have to deal with the hardware side of DASD management, other than modifying IODF and IOCDS using HCD. On the other hand, your installation may be of a smaller scale that doesn't warrant having a dedicated team of storage administrators and so you may have to take on far more of these roles.

1.12.1 DASD management

As a system programmer taking on the additional responsibility of a storage administrator, you might have to handle these aspects of DASD management:

- Adding new DASD devices and volumes to your system
- Implementing SMS
- Effectively managing catalogs
- Dealing with DASD problems

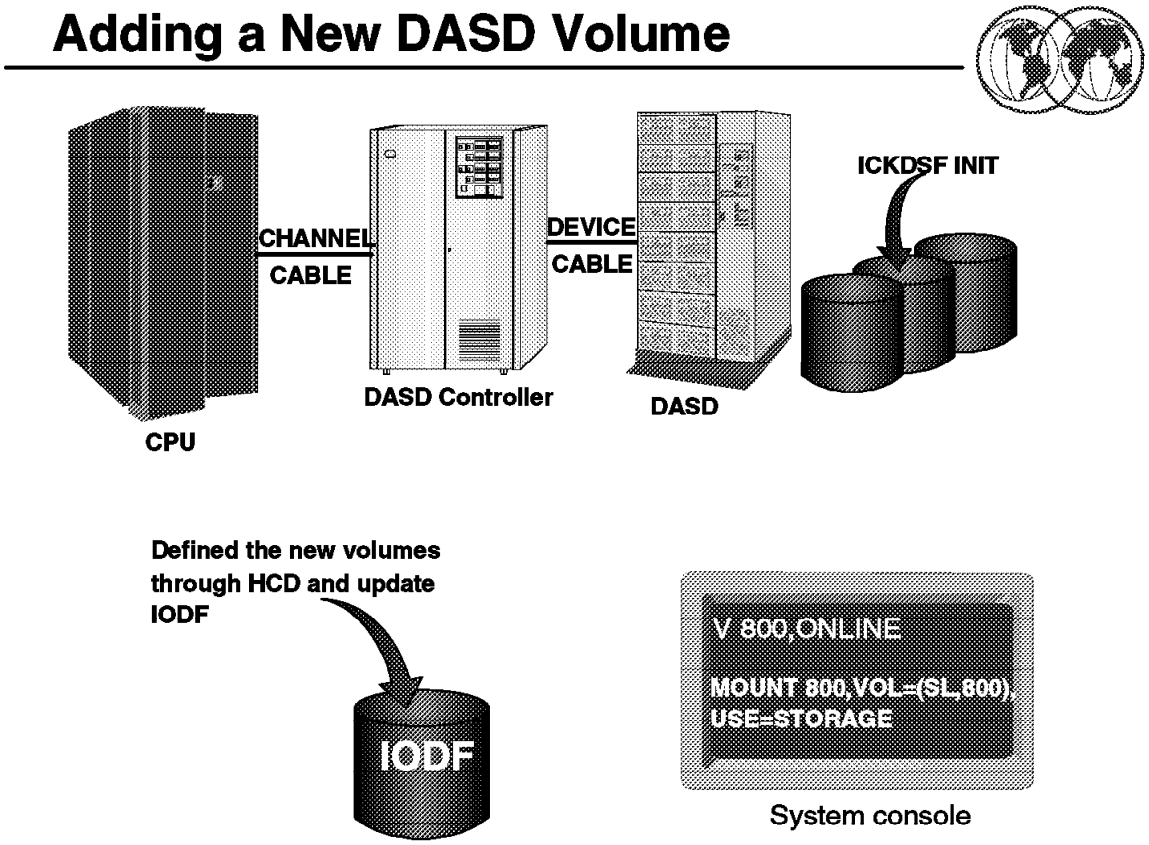


Figure 35. Adding a new DASD volume

1.12.2 How to add additional DASD volumes

The following list outlines the steps of making a new DASD volume available to your system:

1. Physically connect the new device to the storage controller, which is connected to the system through an available channel. Normally, this task is performed by an IBM Customer Engineer who takes care of all the hardware installation.
2. Using HCD, you have to update the IODF and IOCDS to include the new devices. You might want to define the device as belonging to one or more eligible device tables defined in your current IODF, so that data sets can be allocated on it using the UNIT parameter with the associated esoteric device name. For more information on how to use HCD, see *OS/390 Hardware Configuration Definition User's Guide*, SC28-1848.
3. Using ICKDSF, initialize the new volume with a volume serial label, a volume table of contents (VTOC), and a VTOC index. For more information on ICKDSF, see *ICKDSF R16 Refresh, User's Guide*, GC35-0033.
4. Vary the device online, and issue an appropriate MOUNT command, or re-IPL MVS to cause the new volume to be mounted according to the VATLST entries.

   A disk volume is mounted with one of the following use attributes except DFSMS-managed volumes:

   - *PRIVATE*

New data sets will be created on this volume only if the user (by using JCL or the TSO ALLOCATE command or an ISPF menu specification) specifies the volume serial number of this disk volume.

- *PUBLIC*

  MVS may place a temporary data set on this volume, if the user (by using JCL or otherwise) did not specify a volume serial number for the temporary data set. Data sets may also placed on this volume by specifying the volume serial number, as with the PRIVATE volumes. A temporary data set is one with a DSNAME beginning with an ampersand and/or with a disposition equivalent to NEW,DELETE. For more information about the DSNAME parameter in JCL, see *OS/390 MVS JCL Reference*, GC28-1757.

- *STORAGE*

  MVS places permanent data sets on this volume if the user did not supply a volume serial number for the data sets. In addition, temporary data sets, and data sets placed by volume serial number, may also be placed on this volume.

You can set the volume use attributes by:

- Using the MOUNT operator command
- Using the VATLSTxx parmlib member

For more information on the MOUNT command, see the section "Operation" in this chapter and *OS/390 MVS System Commands*, GC28-1781.

The VATLSTxx parmlib member defines the default mount and use attributes for disk volumes found during IPL. The mount attribute determines the conditions under which a volume can be demounted, while the use attribute controls the type of request for which a volume can be allocated.

Example of VATLSTxx

```
VATDEF IPLUSE(PRIVATE),SYSUSE(PRIVATE)
MPCAT1,1,0,3390    ,Y
MPRES1,1,2,3390    ,Y
MPRES2,1,2,3390    ,Y
```

The highlighted column specifies the use attribute. STORAGE is denoted by 0, PUBLIC by 1, and PRIVATE by 2.

For more information on the VATLSTxx parmlib member, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

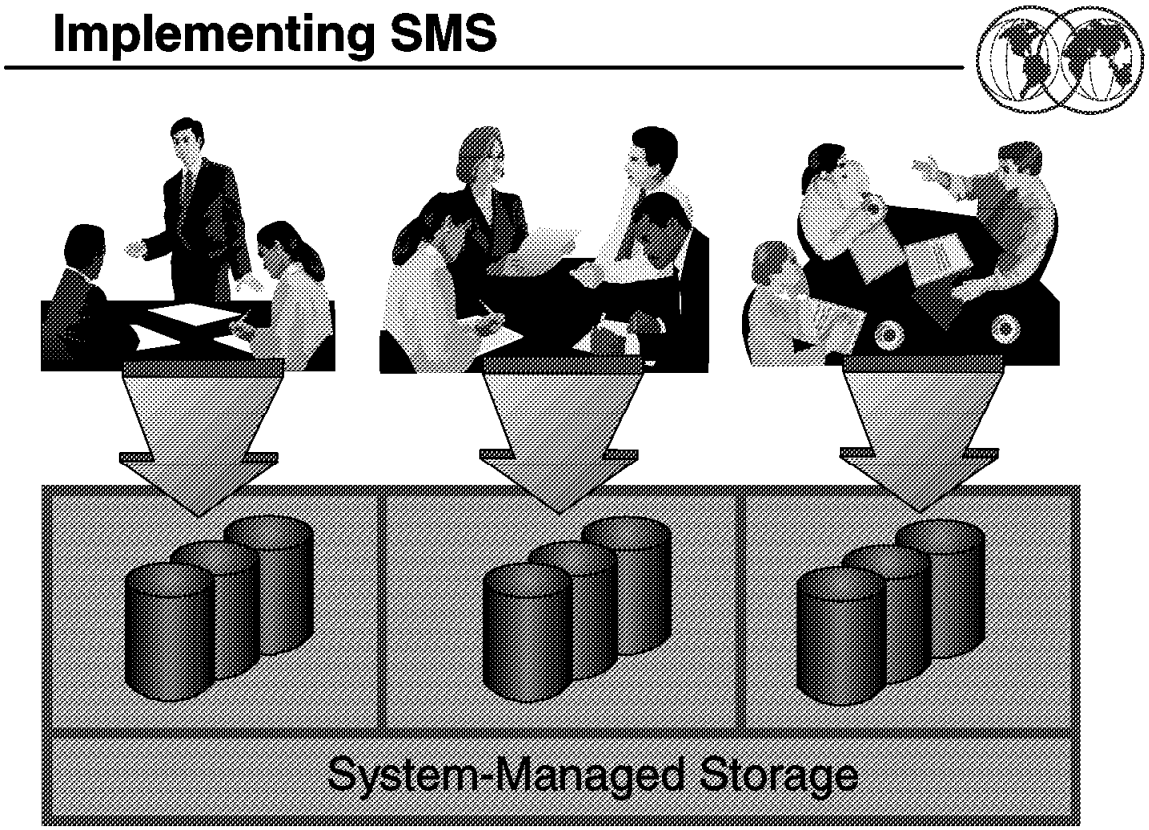


Figure 36. Implementing SMS

1.12.3 Implementing System Managed Storage (SMS)

After you have added the new volumes to the system, you should not make all the volumes available to the users and allow them to use them as they wish. With no restriction on which volumes they can use or how much space they can allocate, your new disk volumes will be filled up fairly quickly. You could spend most of your time providing a new supply of disk volumes for other essential data, tracking down the owner of each data set to establish their importance, and trying to negotiate for space reduction.

One way to solve this problem is to allocate individual volumes to each application, usage type, or group of users. Each of these volumes are mounted PRIVATE and allocation can be done by specifying the volume serial number. However, this method of control has some disadvantages:

- The requirement of specifying a volume serial number in JCL during allocation can lead to inflexibility; for example, a space allocation on one volume might fail even though there is lot of space on other volumes available to the user.
- It is time-consuming and difficult to change the volumes available to the user as this requires changing the hard-coded volume serial number in the JCL.

All these problems can be avoided by implementing System Managed Storage (SMS). It uses software to help you to automate the management of storage, for example data security, data set placement, data set migration and recall, data set backup, data set recovery and deletion, to ensure that current data is available when needed and obsolete data is removed from storage. Some of the benefits of system managed storage are:

- Simplified data allocation
- Ensured data integrity on new allocation
- Improved allocation control

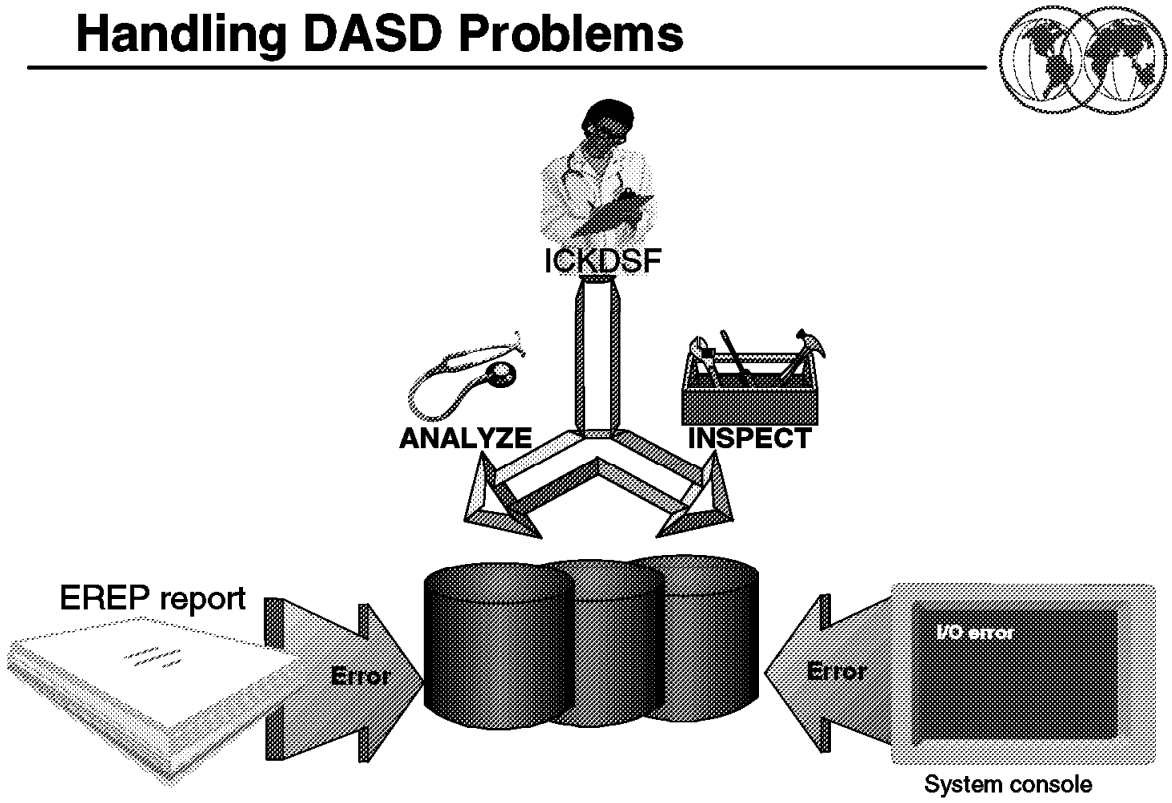


Figure 37. Handling DASD problems

1.12.4 How to deal with DASD problems

The most common DASD problem that you will encounter in your role as a system programmer is I/O error.

The normal error handling processing by the storage subsystem (that is, the storage control and its attached storage devices) and by the operating system, OS/390, includes various functions that recognize errors and recover from them whenever possible.

The storage subsystem performs the following functions:

- Adds ECC (error checking and correction) information to each field of a record when it is written.
- Detects error in reading data, in performing control operations, in functioning of the hardware, and in programming.
- Retries I/O operations for certain error conditions.
- Assembles usage and error information in the form of sense information.
- Maintains counts of disk storage errors.
- Performs ECC correction activity or sends the ECC data to the operating system for correction.

The operating system, MVS, provides standard error recovery procedures to handle errors detected by the storage subsystem:

- *Implements recovery actions*

The specific recovery actions depend on the particular error condition that was defined in the sense information sent from the storage subsystem; for example, retrying an operation when an equipment check is reported in the sense information.

- *Logs usage and error information records*

  The error data that is sent to the system is stored in the Error Recording Data set (ERDS), that is SYS1.LOGREC. The system processes the sense information to produce and supplement data records describing the conditions under which the error occurred. The EREP program formats error reports and may also perform error analysis based on information it obtains from the SYS1.LOGREC. See 1.15.1, “How to clear SYS1.LOGREC” on page 73.

- *Issues system messages at the operator console*

  System console messages may be the initial notification of hardware or media problem. Most information messages contain data on the type and location of an error, and give sense information in hexadecimal format.

The errors shown on the EREP report should be followed up, particularly as temporary errors tend to lead to permanent errors. Normally, you will call an IBM Customer Engineer or your hardware engineer to look into the problem if it turns out to be a hardware error. However, when the problem is caused by a disk media failure, you may have to perform some form of media recovery actions using ICKDSF.

You can use the ANALYZE command to detect and differentiate recording surface and drive-related problems on a volume. It can also scan data to help detect possible media problems. Figure 38 shows sample JCL for the ANALYZE command:

```
//ICKDSF1 JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//ANALYZE  EXEC PGM=ICKDSF
//VOLUME   DD UNIT=3390,VOL=SER=MPDLB2,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   ANALYZE DDNAME(VOLUME) SCAN DRIVETEST
/*
```

Figure 38. Sample JCL for the ANALYZE command

Note: When diagnosing media problems, you should always use the ANALYZE command with the DRIVETEST operand. This will ensure that the device hardware can perform basic operations, such as seek, reads, and writes.

When the ANALYZE report shows that there are potential media errors, you can use the INSPECT command to check the surface of a track to determine if there is a defect, than flag the track defective and assign an alternate track. The PRESERVE operand allows you to save the data on the inspected tracks to the assigned alternate tracks. Figure 39 on page 64 shows sample JCL for the INSPECT command.

```
//ICKDSF2 JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//INSPECT  EXEC PGM=ICKDSF
//VOLUME   DD UNIT=3390,VOL=SER=MPDLB2,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   INSPECT DDNAME(VOLUME) ASSIGN CHECK PRESERVE
/*
```

Figure 39. Sample JCL for the INSPECT command

For more information on ANALYZE and INSPECT, and other ICKDSF commands, see *ICKDSF R16 Refresh, User's Guide*, GC35-0033.

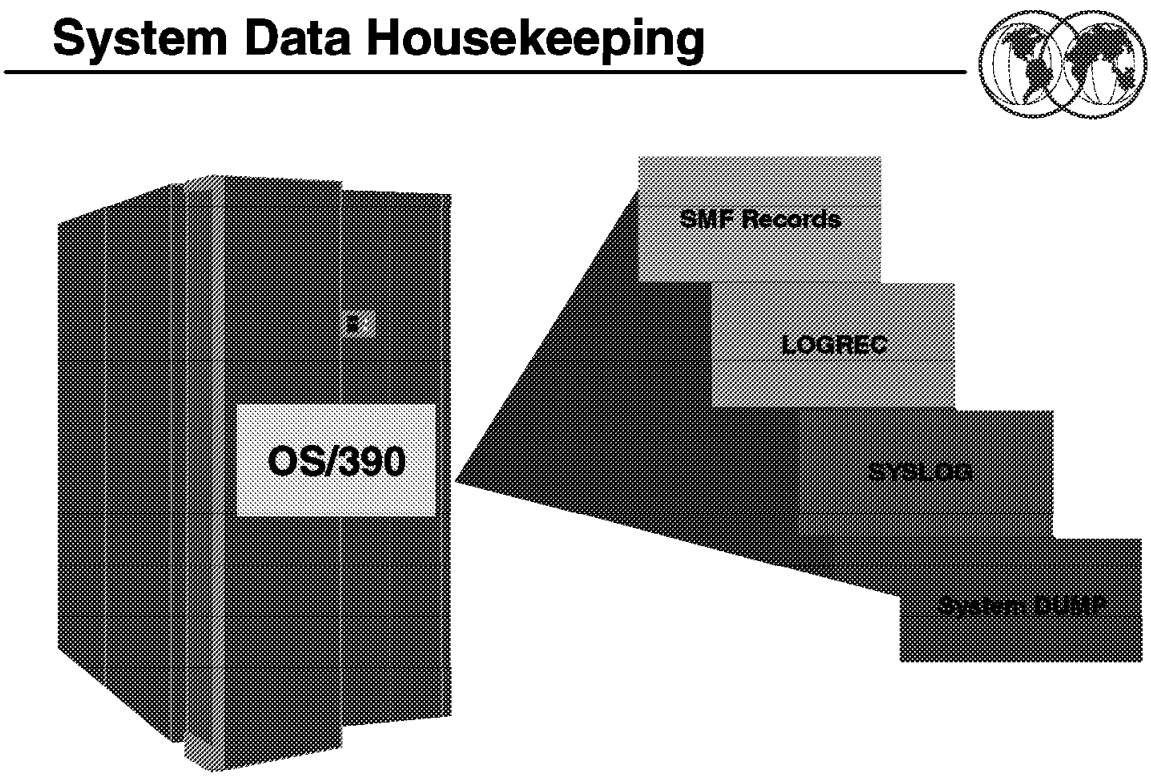


Figure 40. System data housekeeping

1.13 System data housekeeping

To help you to manage your system properly, MVS generates lots of useful management data of various types and stores them in system data sets. There are four types of system data which you will have to deal with:

- *SMF records*

  The SMF records contain a variety of information that enables you to produce many types of analysis reports and summary reports so that you can evaluate changes in configuration, workload, or job scheduling procedures by studying the trends in the data.

  You can also use SMF data to determine system resources wasted because of problems such as inefficient operational procedures or programming conventions

- *LOGREC data*

  The data from the LOGREC data set contains statistical data about machine failures (processor failures, I/O device errors, channel errors). It also contains records for program error recording, missing interrupt information, and dynamic device reconfiguration (DDR) routines.

- *SYSLOG data*

  This data set resides in JES2′s spool space. It can be used by application and system programmers to record communications about problem programs and system functions. It also contains a record of console messages and operator commands for audit and diagnosis purposes.

- *System DUMP data sets*

These system data sets are sequential data sets which contain system dumps, that record areas of virtual storage in case of system task failures. For more information on SYS1.DUMPx data sets, see Chapter 9, “Debugging OS/390.”

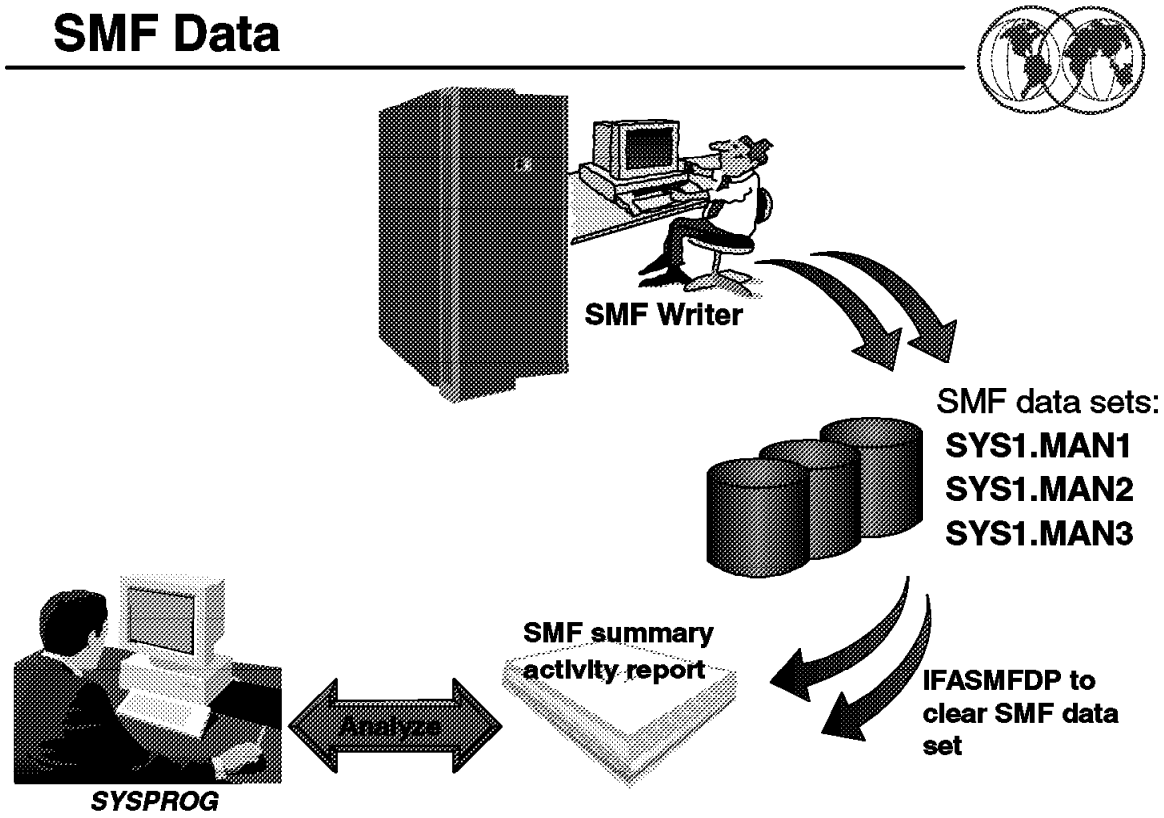


Figure 41. SMF data

1.14 How to collect SMF data

To record SMF data, you have to allocate direct access space and catalog the SMF data sets. You should allocate at least two data sets for SMF use (with at least one on a high-performance device).

Notes:

1. A high-performance device is needed because, if the I/O rate is too slow, data will have to be buffered. The buffers will eventually fill up, which could result in lost data.
2. You also have to consider the following factors when determining which device type when allocating SMF data sets:
   - Your system configuration
   - The amount of SMF data to be written
   - The size of SMF buffers (the control interval size)
   - Your installation's report program requirement

The naming convention used for SMF data sets is SYS1.MANx, such as SYS1.MAN1, SYS1.MAN2, SYS1.MAN3, and so forth. You can use the sample JCL shown in Figure 42 on page 68 to allocate new SMF data sets:

```
//DEFINE  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   DEFINE CLUSTER( -
       CONTROLINTERVALSIZE(4096) -
       CYLINDERS(40) -
       NAME(SYS1.MAN1) -
       NONINDEXED -
       RECORDSIZE(4086,32767) -
       REUSE -
       SHAREOPTIONS(2) -
       SPANNED -
       SPEED -
       VOLUME(MTCAT1) ) -
     DATA( -
       NAME(SYS1.MAN1.DATA) )
/*
```

Figure 42. Sample JCL to allocate new SMF data sets

SMFPRMxx Parmlib Member

```
ACTIVE                          /*ACTIVE SMF RECORDING*/
 DSNAME(SYS1.MAN1,              /*SMF DATA SET NAMES*/
        SYS1.MAN2,              /*SMF DATA SET NAMES*/
        SYS1.MAN3)              /*SMF DATA SET NAMES*/
 NOPROMPT                       /*DON'T PROMPT THE OPERATOR*/
 REC(PERM)                      /*TYPE 17 PERM RECORDS ONLY*/
 INTVAL(10)                    /* INTERVAL RECORDING EVERY 10 MIN */
 MAXDORM(3000)                 /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
 STATUS(010000)                /* WRITE SMF STATS AFTER 1 HOUR*/
 JWT(2400)                     /* 522 AFTER 24 HOURS   */
 SID(&SYSNAME(1:4))            /* SYSTEM ID IS &SYSNAME  */
 LISTDSN                       /* LIST DATA SET STATUS AT IPL*/
 LASTDS(MSG)                   /*DEFAULT TO MESSAGE          */
 NOBUFFS(MSG)                  /*DEFAULT TO MESSAGE          */
 SYS(TYPE(0,6,7,15,17,21,26,30,43,45,61,64,65,66,70:79,80,81,110),
     EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,
           IEFUSI,IEFUJI,IEFUTL,IEFU29),NOINTERVAL,NODETAIL)
```

Figure 43. SMFPRMxx parmlib member

1.14.1 SMFPRMxx parmlib member

Once you have allocated all the required SMF data sets, you have to update the SMFPRMxx in SYS1.PARMLIB to reflect the new SMF data sets. In order to enable SMF recording, make sure that the first parameter in SMPFPRMxx is set to ACTIVE as shown in the visual.

To activate the new changes to SMF parameters, you can use the SET command to make them effective (assuming that you have added SYS1.MAN3 to SMFPRM00) as shown in Figure 44 on page 70.

```
SET SMF=00
IEF196I IEF237I 0800 ALLOCATED TO IEFPARM
IEE252I MEMBER SMFPRM00 FOUND IN SYS1.PARMLIB
IEF196I IEF237I 0802 ALLOCATED TO SYS00004
IEE966I SYS1.MAN3 IS BEING FORMATTED 1
IEF196I IEF285I   SYS1.PARMLIB                                  KEPT
IEF196I IEF285I   VOL SER NOS= MPRES1.
IEE949I 03.01.03 SMF DATA SETS 097
            NAME      VOLSER SIZE(BLKS) %FULL  STATUS
 2 P-SYS1.MAN1 MPCAT1      7200     9  ACTIVE
          S-SYS1.MAN2 MPCAT1      1800     0  ALTERNATE
          S-SYS1.MAN3 MPCAT1      1800     0  ALTERNATE
IEE536I SMF      VALUE 00 NOW IN EFFECT
```

Figure 44. Activating the changes to the SMF parameters

1 SMF found that SYS1.MAN3 has not been formatted. The formatting is now taking place.

2 The name of the SMF data set preceded by a *P* indicates that it is the primary SMF data set. If the name is preceded by a *S*, the data set is a secondary SMF data set.

1.14.1.1 Types of SMF records

You can control which SMF record types are recorded by using the SYS parameter in the SMFPRMxx parmlib member. You can specify SYS(TYPE(0:255)) to record all SMF records. However, this can create considerable overhead and fill up the SMF data sets very quickly. The default SYS parameter is *SYS(NOTYPE(14:19,62:69,99)*. This collects all SMF records except the following:

- Record type 14 is written for input non-VSAM direct access, tape data sets, or VIO data sets.
- Record type 15 is written for output non-VSAM direct access, or VIO data sets.
- Record type 16 is written to record information about events and operations of the sorting program.
- Record type 17 is written when a non-temporary data set or temporary data set is scratch.
- Record type 18 is written when a non-VSAM data set is renamed.
- Record type 19 is written for a list of online devices with specific IPL time frame.
- Record types 62 to 69 are written for various VSAM data sets and catalog activities.
- Record type 99 is written by the System Resource Manager (SRM) component when running in goal mode.

For more information about the rest of SMF record types, see *OS/390 V2R7.0 MVS System Management Facilities (SMF)*, GC28-1783.

Dumping SMF Data

```
//SMFCLR  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1   EXEC PGM=IFASMFDP,REGION=4M
//DUMPIN   DD  DSN=SYS1.MAN1,DISP=SHR
//DUMPOUT  DD  DISP=(NEW,CATLG),DSN=SMF.ALLRECS,
//             UNIT=SYSALLDA,VOL=SER=MPDLB2,SPACE=(CYL,(30,10),RLSE),
//             DCB=(RECFM=VBS,LRECL=32760,BLKSIZE=4096)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
     INDD(DUMPIN,OPTIONS(ALL))
/*
```

Figure 45. Dumping SMF data

1.14.2 Dumping SMF data

When the current recording data set cannot accommodate any more records, the SMF writer routine automatically switches recording from the active SMF data set to an empty SMF data set and issues the following message regarding the full data set:

```
*IEE362A SMF ENTER DUMP FOR SYS1.MAN1 ON MPCAT1
```

You can use the sample JCL shown in the visual to dump the SMF data to a user data set and clear the SMF data set for reuse.

OPTIONS(ALL) indicates that the input data set specified in the DUMPIN DD statement is to be copied, then reset and preformatted.

Note: The output data set in the DUMPOUT DD statement should be large enough to hold one SMF data set, the minimum requirement is 30 cylinders.

Alternatively, you may code an IEFU29 exit to submit the job. The IEFU29 exit is invoked (if activated in SMFPRMxx) whenever an SMF data set switch occurs. For more information about the IEFU29 dump exit, see *OS/390 V2R7.0 MVS Installation Exits*, SC28-1753.

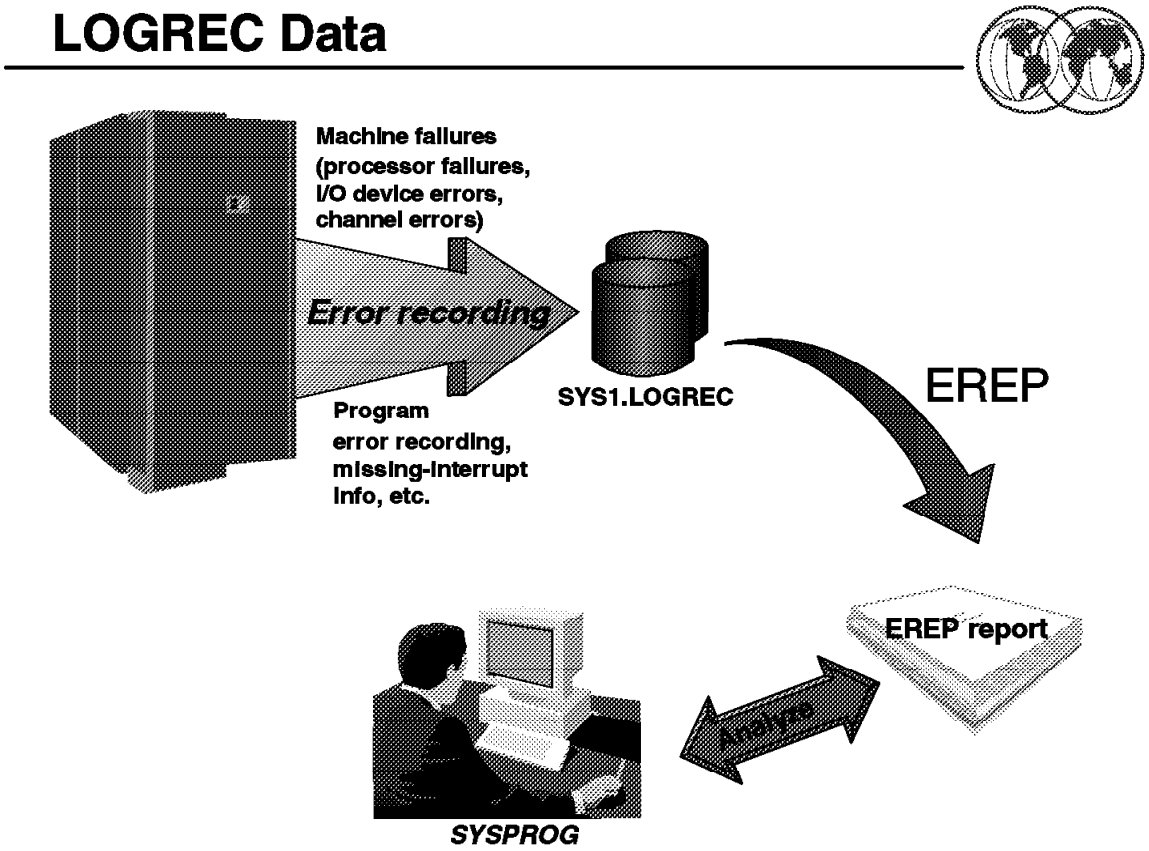


Figure 46. LOGREC data

1.15 How to create SYS1.LOGREC

When an error occurs, the system records information about the error in the LOGREC data set. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions. You can use the Environment Record, Editing, and Printing (EREP) program to:

- Print reports about the system records
- Determine the history of the system
- Learn about a particular error

Before the system can record all this information, you must first allocate the LOGREC data set and initialize it.

Note: Whenever you allocate or reallocate the LOGREC data set, the newly allocated data set will not be used until you initialize it and IPL the system on which it is to be used.

You can use the sample JCL shown in Figure 47 on page 73 to accomplish the task.

```
//LOGREC  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//*********************************************
//* CREATE A NEW LOGREC DATASET AND INITIALIZE IT
//*********************************************
//STEP1   EXEC PGM=IFCDIP00,COND=(4000,LT)
//SERERDS  DD UNIT=SYSDA,VOL=SER=MPRES1,SPACE=(CYL,3,,CONTIG),
//            DSN=SYS1.LOGREC,DISP=(,CATLG)
//SYSPRINT DD SYSOUT=*
```

Figure 47. Sample JCL to create and initialize a new LOGREC data set

Note: The IBM-supplied default name for the LOGREC data set is SYS1.LOGREC. To change the LOGREC data set name known to the system, use the LOGREC parameter of the IEASYSxx parmlib member.

If you use the default name SYS1.LOGREC in a multisystem environment take care to ensure that the LOGREC data set is uniquely specified with the volume-serial. This will prevent the initialization of another system's LOGREC data set in the event that IFCDIP00 is run on a different system.

1.15.1 How to clear SYS1.LOGREC

The LOGREC data set will eventually become full, the same as the SMF data set, and console messages appear requesting the operator to take action. Therefore, this data set needs to be cleared so that it can be reused. The process is a little different because there is only one LOGREC data set.

As a system programmer, you have to set up a process to clear the LOGREC data set when the following message appears:

```
IFB080E LOGREC DATA SET NEW FULL, DSN=SYS1.LOGREC
```

This is issued when the LOGREC data set becomes 90 percent full; if action is not taken and LOGREC fills up completely, the following message is issued which requires prompt action:

```
IFB081I LOGREC DTA SET IS FULL, hh.mm.ss, DSN=SYS1.LOGREC
```

At this stage, if you do not take any action, the system continues processing, but further error records will be lost. You can use the sample JCL shown in Figure 48 to clear your LOGREC data set.

```
//CLRLOG  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP    EXEC   PGM=IFCEREP1,PARM='CARD'
//SERLOG  DD  DSN=SYS1.LOGREC,DISP=OLD
//ACCDEV  DD  DSN=SYS1.LOGREC.TEMPSTOR,DISP=MOD
//TOURIST DD  SYSOUT=*
//EREPPT  DD  SYSOUT=*
//SYSIN   DD  DUMMY
     SYSUM
     ACC=Y
     ZERO=Y
/*
```

Figure 48. Sample JCL to clear the LOGREC data set

The job will copy the LOGREC data set to the data set specified in the DDNAME ACCDEV, print a brief summary of LOGREC data, and then clear the data set. For more information about the rest of the parameters, see *EREP V3R5 Reference*, GC35-0152.

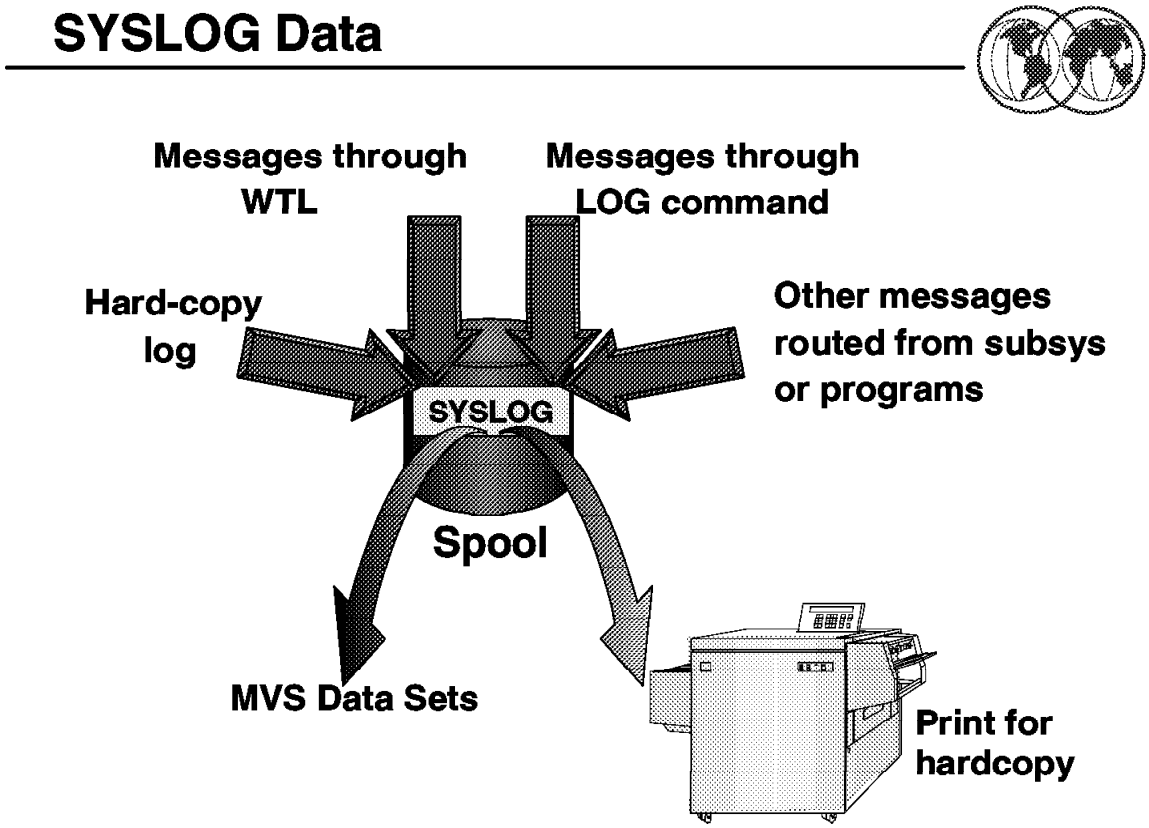


Figure 49. The SYSLOG data set

1.16 The SYSLOG

The SYSLOG on the MVS system actually consists of two separate logs: the system log and the hardcopy log.

The hardcopy log is a record of all system message traffic:

- Messages to and from all consoles
- Commands and replies entered by the operator

In a JES3 system, the hardcopy log is always written to the SYSLOG. While in a JES2 system, you can choose to have the hardcopy log written to the SYSLOG or to a console printer. You can do this by specifying the HARDCOPY statement in the CONSOLxx parmlib member as shown in Figure 50.

```
HARDCOPY DEVNUM(SYSLOG,OPERLOG) 1
         ROUTCODE(ALL)
         CMDLEVEL(CMDS)
         UD(Y)
         HCFORMAT(CENTURY)
```

Figure 50. Example of HARDCOPY statement in CONSOLxx

1 SYSLOG indicates that the system log is to be the hardcopy medium, while OPERLOG indicates that the operations log will be activated and will receive the hardcopy message set.

Note: The system defaults to SYSLOG as the hardcopy medium in any of the following cases:

- You do not code a HARDCOPY statement.
- You specify an unusable hardcopy console.
- You specify OPERLOG alone but the operations log is unusable.

For more information on the HARDCOPY statement, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

The SYSLOG is a SYSOUT data set provided by the job entry subsystem (either JES2 or JES3). SYSOUT data sets are output spool data sets on direct access storage devices (DASD). An installation should print the SYSLOG periodically to check for problems. The SYSLOG consists of the following:

- All messages issued through Write To Log (WTL) macros.
- All messages entered by the LOG operator commands.
- Usually, the hardcopy log.
- Any messages routed to the SYSLOG from any system component or program.

You can limit the maximum number of WTLs (messages) allowed for the SYSLOG at IPL time by using the LOGLMT parameter in the IEASYSxx parmlib member. The value is used by log processing to determine when the SYSLOG should be be scheduled for SYSOUT processing by JES. When this value is reached, the log processing:

- Issues a simulated WRITELOG command to close and free the current SYSLOG.
- Releases the closed SYSLOG to the printer queue whose output class is specified by the LOGCLS parameter of the IEASYSxx parmlib member.
- Allocates and opens a new SYSLOG.

Example of LOGLMT and LOGCLS in IEASYSxx

```
LOGCLS=L, LOGLMT=99999,
```

In the example, the SYSLOG is scheduled for SYSOUT processing on output class L when 99,999 WTLs have been issued. For more information on LOGLMT and LOGCLS, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

Remember that the SYSLOG provides a permanent, consolidated record of all console messages and commands, that is, the event log of the system. Thus, you might want to review the SYSLOG at a later date when you are doing problem diagnosis. Instead of having the SYSLOG written to the output queue after a certain number of WTLs, you should set up a procedure to housekeep the SYSLOG data set depending on your installation′s requirement.

You can perform your housekeeping as follows:

1. Set up a procedure to have the system operator issue the `WRITELOG.` command to close and allocate a new SYSLOG at the end of each processing day. For example, `WRITELOG H.`

   Note: Class H is not a default printer class so the SYSLOG data set is kept in the spool.
2. At the end of the week, you will have seven SYSLOG data sets in the spool. You can now decide to offload the SYSLOG to an MVS data set or route it to a printer for a hardcopy printout.
3. It′s really not necessary to keep SYSLOG data sets for more than a week, so if you keep a week′s worth of SYSLOG data sets, you should not have a problem.

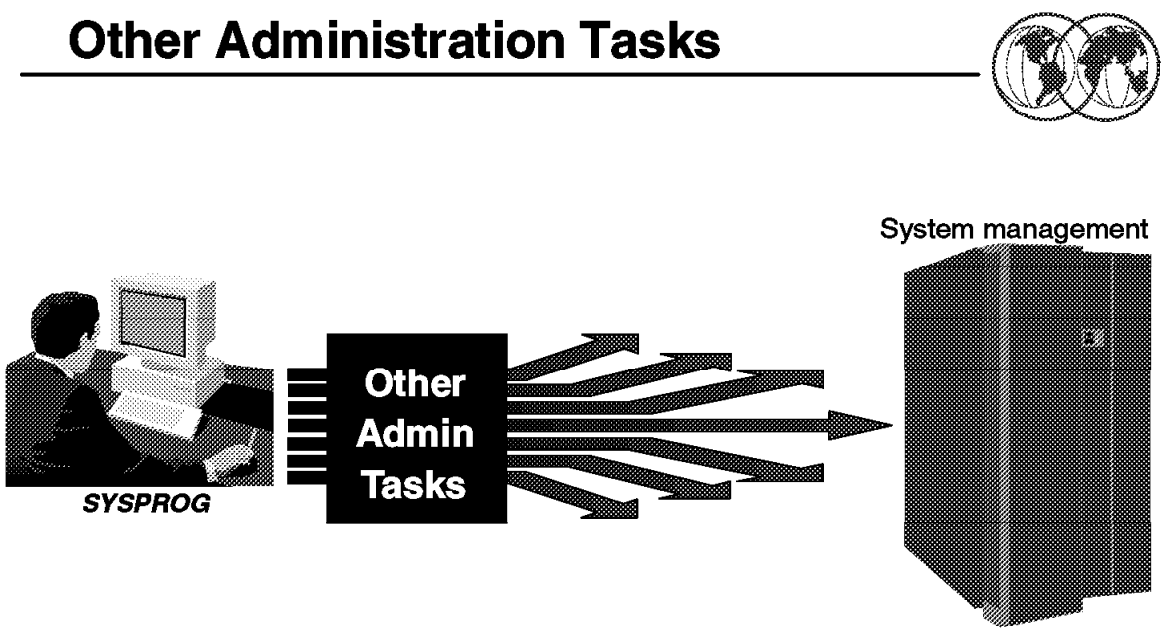


Figure 51. Other administration tasks

1.17 Other administration tasks

In this section, we discuss some other administrative tasks and techniques:

- How to work with MIH
- How to add page data sets
- How to change TSO timeout
- How to add spool volumes
- How to delete spool volumes
- How to verify system configuration
- How to view SYSOUT using ISPF
- How to change your TSO profile
- How to back up and restore OS/390

Figure 52. Working with MIH

1.17.1 How to work with Missing Interrupt Handler (MIH)

When OS/390 sends channel commands to a control unit, it waits for a response. If no response is received after a certain amount of time, a missing interrupt situation exists. The following 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

You can use the IECIOSxx parmlib member to specify the timing limits for devices in your system. 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 this limit is reached, the system abnormally ends the I/O request.

Hot I/O refers to a hardware malfunction that causes repeated, unsolicited I/O interrupts. If those I/O interrupts are not deleted, the system can loop or the System Queue Area (SQA) can be depleted. The I/O subsystem (IOS) tries to detect the hot I/O condition and performs recovery actions before the system requires an IPL. Recovery actions taken for a hot device depend on the type of device and its reserve status. You can use the HOTIO statement in IECIOSxx as shown in Figure 53 on page 79 to

modify the Hot I/O Detection Table (HIDT), and to eliminate operator intervention where recovery actions defined in HIDT (by default) require the operator to response to a message.

```
HOTIO DVTHRSH=200
MIH TIME=00:30,DEV=(2E8-2FF,7300-7370)
HOTIO DFLT111=(CHPK,CHPF),DFLT112=(CHPK,OPER)
MIH IOTIMING=00:12,DEV=(2E8-2FF,730-737)
MIH TIME=00:00,DEV=(180-187,230,B10-B17)
```

Figure 53. Example of IECIOSxx

Once you have updated the IECIOSxx parmlib member, you can activate the changes using the SET command as shown in Figure 54.

```
SET IOS=00
IEE252I MEMBER IECIOS00 FOUND IN SYS1.PARMLIB
IOS090I IECIOS00. MIH UPDATE(S) COMPLETE
IEE536I IOS      VALUE 00 NOW IN EFFECT
```

Figure 54. Using the SET command to activate the IECIOSxx changes

For more information on IECIOSxx, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

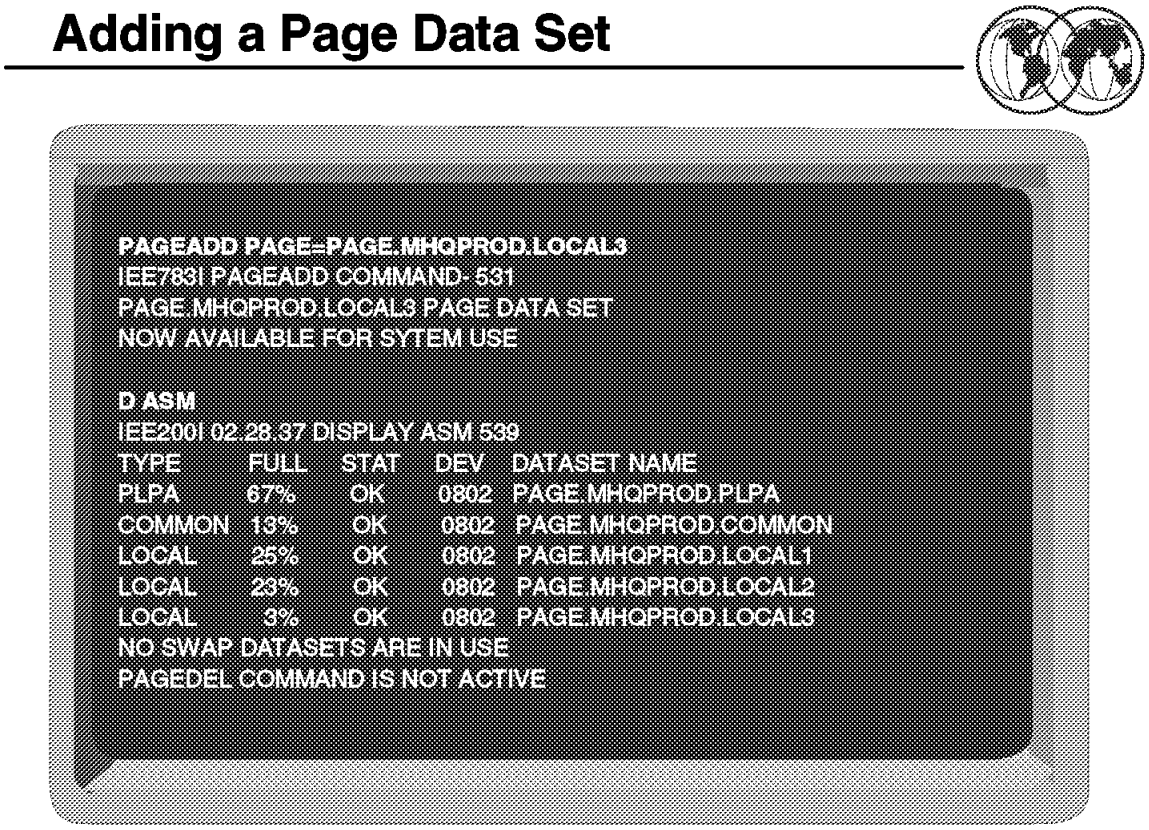


Figure 55. Adding page data sets

1.17.2 How to add additional page data sets

OS/390 can support a large amount of virtual storage by using paging data sets. Data sets named page spaces are used to contain this virtual storage. Another term often used to collectively describe these page spaces is auxiliary storage which is managed by Auxiliary Storage Manager (ASM).

As your workload increases, you may run out of available auxiliary storage, and the system issues the following message:

```
IRA200E AUXILIARY STORAGE SHORTAGE
```

At this moment, the system detected a shortage of available slots in the auxiliary storage paging space, when 70 percent of all available slots are already in use. The system rejects `LOGON`, `MOUNT`, and `START` commands until the shortage is relieved.

If no action is taken, the system issues the following message when 90 percent of all available auxiliary storage is in use:

```
IRA201E CRITICAL AUXILIARY STORAGE SHORTAGE
```

The solution to this problem is to increase the auxiliary storage available for OS/390 use by defining more page space. You can use the sample JCL shown in Figure 56 on page 81 to define a new page data set:

```
//DEFPAGE  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1    EXEC PGM=IDCAMS,COND=(4000,LT)
//SYSPRINT DD SYSOUT=*
//PAGELOC  DD UNIT=3390,VOL=SER=MPCAT1,DISP=OLD
//SYSIN    DD *
  DEFINE PAGESPACE(  -
      FILE(PAGELOC) -
      NAME(PAGE.MHQPROD.LOCAL3) -
      CYLINDERS(200) -
      VOLUME(MPCAT1) )
/*
```

Figure 56. Sample JCL to define a new page data set

After you have successfully defined a new page data set, you can now relieve the auxiliary shortage by adding the new page data set to the system as shown in Figure 57.

```
PAGEADD PAGE=PAGE.MHQPROD.LOCAL3
IEE783I PAGEADD COMMAND- 531
PAGE.MHQPROD.LOCAL3 PAGE DATA SET
NOW AVAILABLE FOR SYSTEM USE
```

Figure 57. Adding the new page data set to the system

After you have added the new page data set to the system, the system issues the following message:

```
IRA202I AUXILIARY STORAGE SHORTAGE RELIEVED
```

You can check the status of the auxiliary storage by issuing the command shown in Figure 58.

```
D ASM
IEE200I 02.28.37 DISPLAY ASM 539
TYPE     FULL STAT   DEV  DATASET NAME
PLPA      67%   OK  0802  PAGE.MHQPROD.PLPA
COMMON    13%   OK  0802  PAGE.MHQPROD.COMMON
LOCAL     25%   OK  0802  PAGE.MHQPROD.LOCAL1
LOCAL     23%   OK  0802  PAGE.MHQPROD.LOCAL2
LOCAL      3%   OK  0802  PAGE.MHQPROD.LOCAL3
NO SWAP DATASETS ARE IN USE
PAGEDEL COMMAND IS NOT ACTIVE
```

Figure 58. Checking the status of the auxiliary storage

To ensure this new page data set remains a permanent part of your OS/390 system, you should update the PAGE= parameter in your IEASYSxx parmlib member to include the new page data set as shown in Figure 59 on page 82.

```
PAGE=(PAGE.MHQPROD.PLPA,
      PAGE.MHQPROD.COMMON,
      PAGE.MHQPROD.LOCAL1,
      PAGE.MHQPROD.LOCAL2,
      PAGE.MHQPROD.LOCAL3,L),
```

Figure 59. Updating the PAGE parameter in IEASYSxx

Changing TSO Timeout

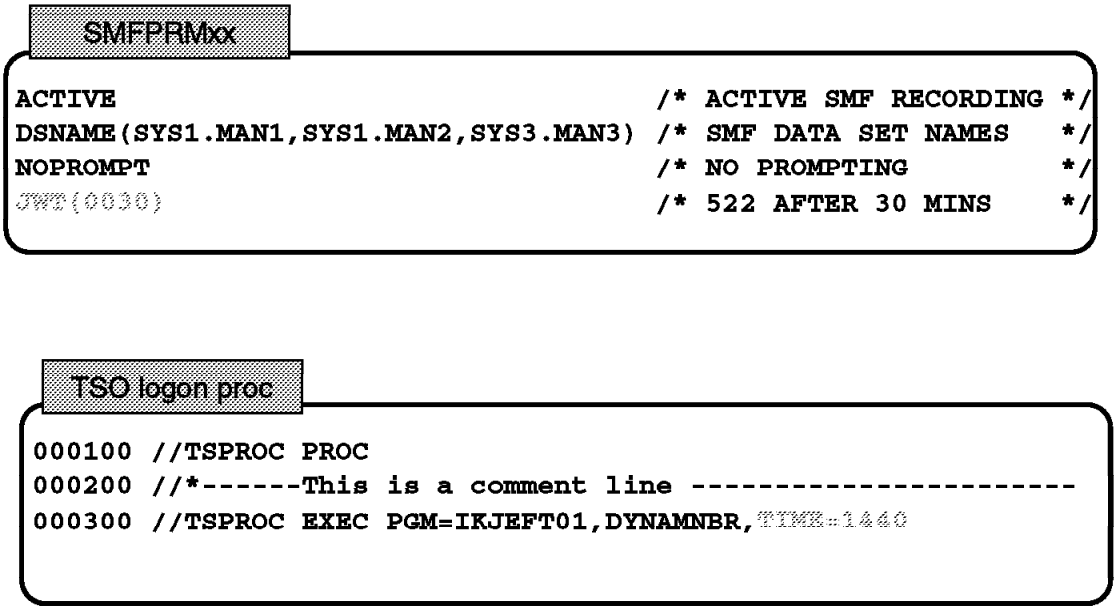


Figure 60. Changing TSO timeout

1.17.3 How to change the TSO timeout value

After a period of inactivity, TSO will automatically log off a user. This elapsed time is set in the SMFPRMxx parmlib member using the parameter JWT(hhmm) as shown in Figure 61 on page 84, where *hh* is the amount of real time in hours and *mm* is in minutes.

```
ACTIVE                          /*ACTIVE SMF RECORDING*/
DSNAME(SYS1.MAN1,              /*SMF DATA SET NAMES*/
       SYS1.MAN2,              /*SMF DATA SET NAMES*/
       SYS1.MAN3)              /*SMF DATA SET NAMES*/
NOPROMPT                       /*DON'T PROMPT THE OPERATOR*/
REC(PERM)                      /*TYPE 17 PERM RECORDS ONLY*/
INTVAL(10)                    /* INTERVAL RECORDING EVERY 10 MIN */
MAXDORM(3000)                 /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
STATUS(010000)                /* WRITE SMF STATS AFTER 1 HOUR*/
JWT(0030)                     /* 522 AFTER 24 HOURS   */
SID(&SYSNAME(1:4))            /* SYSTEM ID IS SC68  */
LISTDSN                       /* LIST DATA SET STATUS AT IPL*/
LASTDS(MSG)                   /*DEFAULT TO MESSAGE          */
NOBUFFS(MSG)                  /*DEFAULT TO MESSAGE          */
SYS(TYPE(0,6,7,15,17,21,26,30,43,45,61,64,65,66,70:79,80,81,110),
    EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,
          IEFUSI,IEFUJI,IEFUTL,IEFU29),NOINTERVAL,NODETAIL)
```

Figure 61. Example of SMFPRMxx member

In the example, a user is forced off after 30 minutes of no activity. You can edit this PARMLIB member and change this value. You can set this new value by using the SET operator command:

```
SET SMF=00
```

Note:

There is a possibility that TSO users are not being timed out when the value specified in the JWT parameter has expired. This usually happens when TIME=1440 is specified in the TSO logon procedure that the users used to logon to the system.

```
//TSPROC PROC
//TSPROC EXEC PGM=IKJEFT01,DYNAMNBR=256,TIME=1440
```

The TIME=1440 is a special value that will:

- Allow the particular job to run for an unlimited amount time
- Ignore the JWT parameter in SMFPRMxx parmlib member for the job

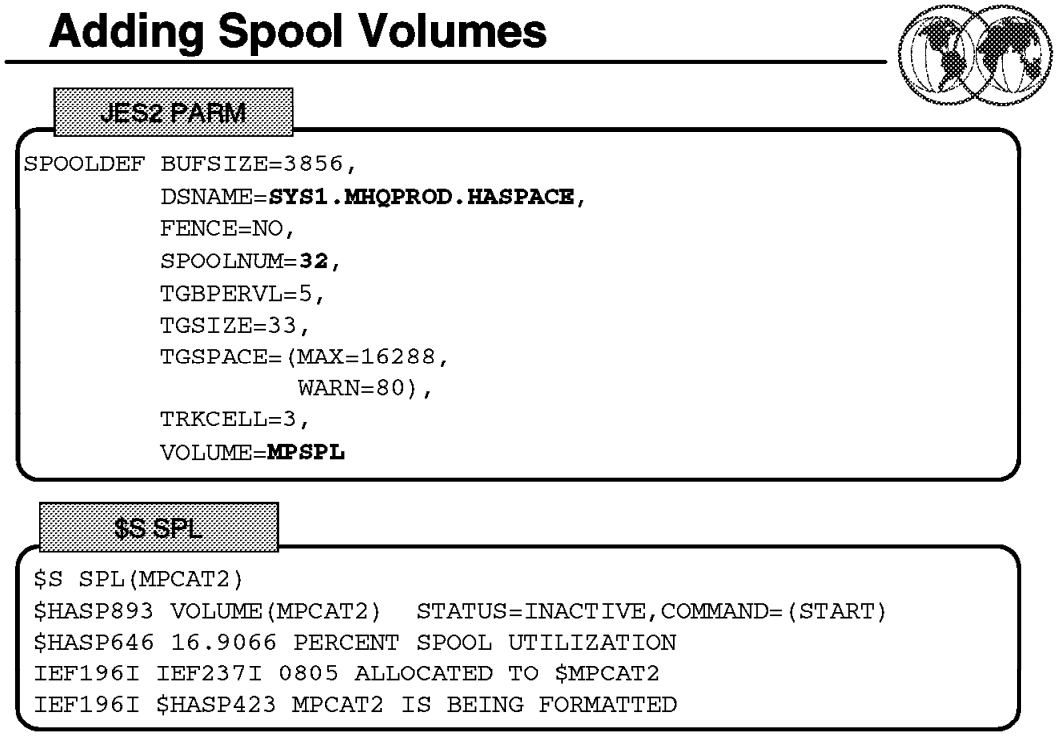


Figure 62. Adding spool volumes

1.17.4 How to add spool volumes

The spool is the repository for all input jobs and most system out (SYSOUT) that JES2 manages. Due to fluctuating workload requirements, such as I/O errors, hardware failures, or utilization needs, you might need to add spool packs.

Spool volumes are added either at cold start or dynamically through the use of the operator command $S SPL. You can define a maximum of 253 spool volumes to JES2 at any one time.

To define the spool volumes during JES2 startup, you use the SPOOLDEF statement in the JES2 PARM. Figure 63 an example of SPOOLDEF statement in the JES2 PARM:

```
SPOOLDEF BUFSIZE=3856,       /* MAXIMUM BUFFER SIZE     &BUFSIZE     c*/
         DSNAME=SYS1.MHQPROD.HASPACE 1 ,
         FENCE=NO,           /* Don't Force to Min.Vol. &FENCE      oc*/
         SPOOLNUM=32 2 ,        /* Max. Num. Spool Vols     c*/
         TGBPERVL=5,         /* Track Groups per volume in BLOB   ownc*/
         TGSIZE=33,          /* 30 BUFFERS/TRACK GROUP  &TGSIZE    wnc*/
         TGSPACE=(MAX=16288, /* Fits TGMs into 4K Page  &NUMTG=(,    c*/
                  WARN=80),  /*                        &NUMTG=(,%   onc*/
         TRKCELL=3,          /* 3 Buffers/Track-cell    &TCELSIZ     c*/
         VOLUME=MPSPL 3   /* SPOOL VOLUME SERIAL   &SPOOL     c*/
```

Figure 63. SPOOLDEF statement

1 The JES2 spool data set name.

2 This specifies the maximum number of spool volumes which can be defined at any one time.

3 Specifies the four- to five-character prefix assigned to JES2′s spool volumes. The first four to five characters of the volume serial must be identical to the character specified by this parameter. Those volumes beginning with this prefix and a data set named the same as the DSNAME specification are considered JES2 volumes.

Note: If the maximum of 253 spool volumes is exceeded during a cold start, JES2 issues a message informing the operator that more spool volumes were found than expected from the SPOOLNUM parameter on the SPOOLDEF initialization statement.

As mentioned, you can also add spool volumes dynamically by using operator command, $S SPL as shown in Figure 64.

```
$S SPL(MPCAT2)
$HASP893 VOLUME(MPCAT2)  STATUS=INACTIVE,COMMAND=(START)
$HASP646 16.9066 PERCENT SPOOL UTILIZATION
IEF196I IEF237I 0805 ALLOCATED TO $MPCAT2
IEF196I $HASP423 MPCAT2 IS BEING FORMATTED
$HASP423 MPCAT2 IS BEING FORMATTED
$HASP850 3750 TRACK GROUPS ON MPCAT1
$HASP850 586 TRACK GROUPS ON MPCAT2
$HASP851    28232 TOTAL TRACK GROUPS MAY BE ADDED
$HASP630 VOLUME MPCAT2 ACTIVE     0 PERCENT UTILIZATION
```

Figure 64. $S SPL command

Note: If the dynamic addition of another spool volume is attempted (thereby exceeding the maximum of 253 spool volumes), the command will be ignored, the volume will not be added, and the message $HASP411 will be issued to the operator.

You can now display the status and the percentage utilization of all spool volumes using the operator command, $D SPL as shown in Figure 65.

```
$DSPL
$HASP893 VOLUME(MPCAT1)  STATUS=ACTIVE,PERCENT=16
$HASP893 VOLUME(MPCAT2)  STATUS=ACTIVE,PERCENT=0
$HASP646 14.7370 PERCENT SPOOL UTILIZATION
```

Figure 65. $D SPL command

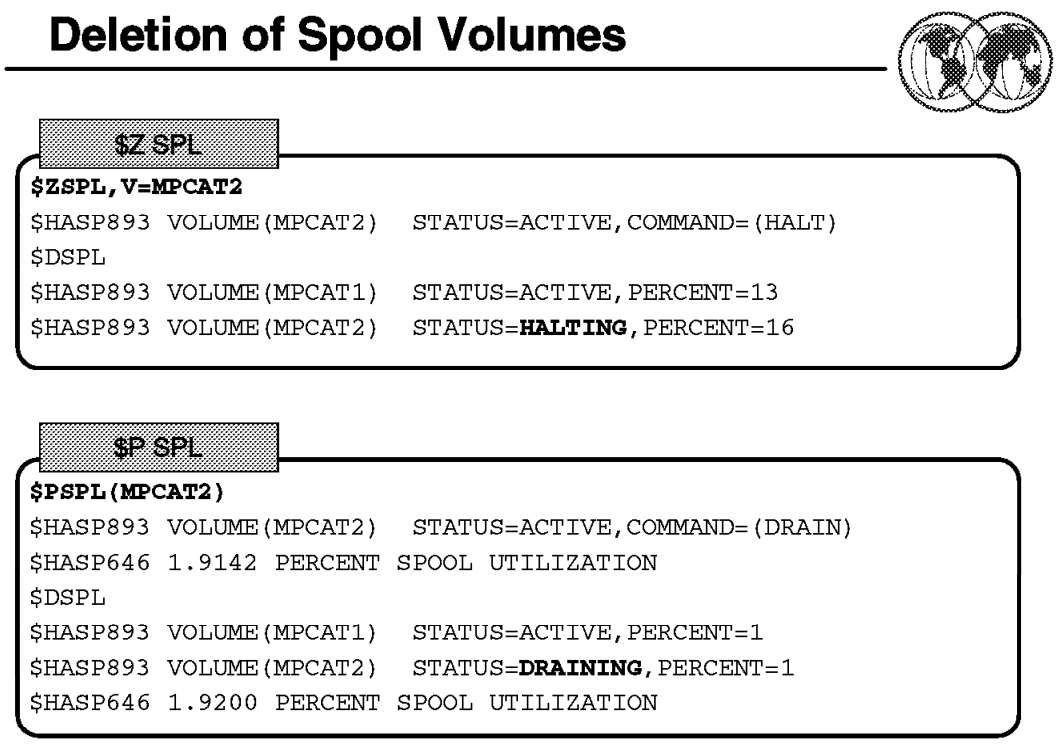


Figure 66. Deletion of spool volumes

1.17.5 How to delete spool volumes

Spool volumes can be deleted dynamically as required to meet your installation's requirements using the operator commands:

- $Z SPL
- $P SPL

Some considerations must be recognized and proper precautions must be taken to prevent improper usage of the spool volume deletion commands:

- If the volume being deleted (halted or drained) contains spooled or remote messages or the JESNEWS data set, they are automatically moved to another volume.

  Note: This process may take some time as spooled remote messages and SYSOUT are only moved from the draining volume when they become the lowest-numbered job on the volume.

- If there are no active volumes available for these data sets, the deletion process will not complete until an active volume is available to accept them.

You use the $Z SPL command to halt an active spool volume. When you issued the $Z command, no new work is allowed to start and no new space on the volume is allocated. Any currently allocated space remains intact and the volume can be removed without the loss of work.

The volume status is HALTING until all currently active jobs which have space allocated on the volume are completed. After that the status of the volume changes to INACTIVE. Figure 67 on page 88 shows the response from the system after you issued $Z command:

```
$ZSPL,V=MPCAT2
$HASP893 VOLUME(MPCAT2)  STATUS=ACTIVE 1 ,COMMAND=(HALT)
$DSPL 2
$HASP893 VOLUME(MPCAT1)  STATUS=ACTIVE,PERCENT=13
$HASP893 VOLUME(MPCAT2)  STATUS=HALTING 3 ,PERCENT=16
$HASP646 13.7066 PERCENT SPOOL UTILIZATION
        .
        .
        .
$HASP395 LSTDDF   ENDED
$HASP309 INIT B    INACTIVE ******** C=GA
IEF196I IEF285I   SYS1.MHQPROD.HASPACE                          KEPT
IEF196I IEF285I   VOL SER NOS= MPCAT2.
$HASP630 VOLUME MPCAT2 INACTIVE 4   16 PERCENT UTILIZATION
```

Figure 67. System response to the $Z command

1 The initial status of the spool volume, MPCAT2 was ACTIVE.

2 To display the status of spool volumes, you used the $DSPL command.

3 After the $Z command was issued, the status changed to HALTING.

4 When the job (LSTDDF) ends, the status of the spool volume (MPCAT2) changed to INACTIVE.

To drain and delete an entire spool volume, you can use the $P SPL operator command. The command prevents any available tracks on the draining spool volume from being selected for allocation. Until all jobs currently allocated to the volume have completed all phases of job processing the volume is considered to have a status of draining. The spool volume is drained when no allocated spool space remains on that volume, and the volume is unallocated to all systems in the multi-access spool complex.

Note: Once the spool volume is drained, JES2 does not retain any information concerning that volume. Therefore, you cannot display any information about the volume.

Figure 68 shows the system response when you issue the $P SPL command when an active job is running:

```
$PSPL(MPCAT2)
$HASP893 VOLUME(MPCAT2)  STATUS=ACTIVE 1 ,COMMAND=(DRAIN)
$HASP646 1.9142 PERCENT SPOOL UTILIZATION
$DSPL
$HASP893 VOLUME(MPCAT1)  STATUS=ACTIVE,PERCENT=1
$HASP893 VOLUME(MPCAT2)  STATUS=DRAINING 2 ,PERCENT=1
$HASP646 1.9200 PERCENT SPOOL UTILIZATION
```

Figure 68. System response to a $P SPL command when the system is running

1 The initial status of the spool volume to be drained.

2 After the $PSPL(MPCAT2) command was issued, the status of the spool volume (MPCAT2) changed to DRAINING.

If you want to cancel and deallocate all cancellable jobs on the drained spool volume, you can use the CANCEL operand together with the $P SPL command. All jobs and SYSOUT will be deleted from the

volume; any spooled remote messages and the JESNEWS data set (if present) are automatically moved to another active spool volume.

Note: The $P SPL,CANCEL command can also be used to drain an inactive volume, purging all jobs allocated to it. However, this can result in lost track groups on other volumes. You can recover any lost space by doing an all-member warm start or the JES2 automatic spool reclamation function will also recover them within one week, whichever occurs first. You can prevent the loss of spool space by using the $S SPL,P,CANCEL command only if the spool volume can be remounted. Figure 69 shows an example of $S SPL,P,CANCEL.

```
$SSPL(MPCAT2),P,CANCEL 1
$HASP893 VOLUME(MPCAT2)  STATUS=DRAINING,COMMAND=(START,DRAIN)
$HASP646 7.1200 PERCENT SPOOL UTILIZATION
        .
        .
        .
IEF196I IEF285I   SYS1.MHQPROD.HASPACE                          KEPT
IEF196I IEF285I   VOL SER NOS= MPCAT2.
$HASP806 VOLUME MPCAT2 DRAINED 2
```

Figure 69. Example of $ SPL,P,CANCEL command

1 The command will fail with the message: REQUEST INVALID DUE TO ACTIVE STATUS if the spool volume is not in the DRAINING status.

1 At the successful completion of the command, the status of the spool changed to DRAINED which indicates you can no longer display any information about the volume.

Note

If *all* the spool volumes are deleted, JES2 will not be capable of performing any work. Hence, be very careful when you delete any spool volumes.

For more information on the use and syntax of the spool configuration commands, see *OS/390 V2R7.0 JES2 Commands*, GC28-1790.

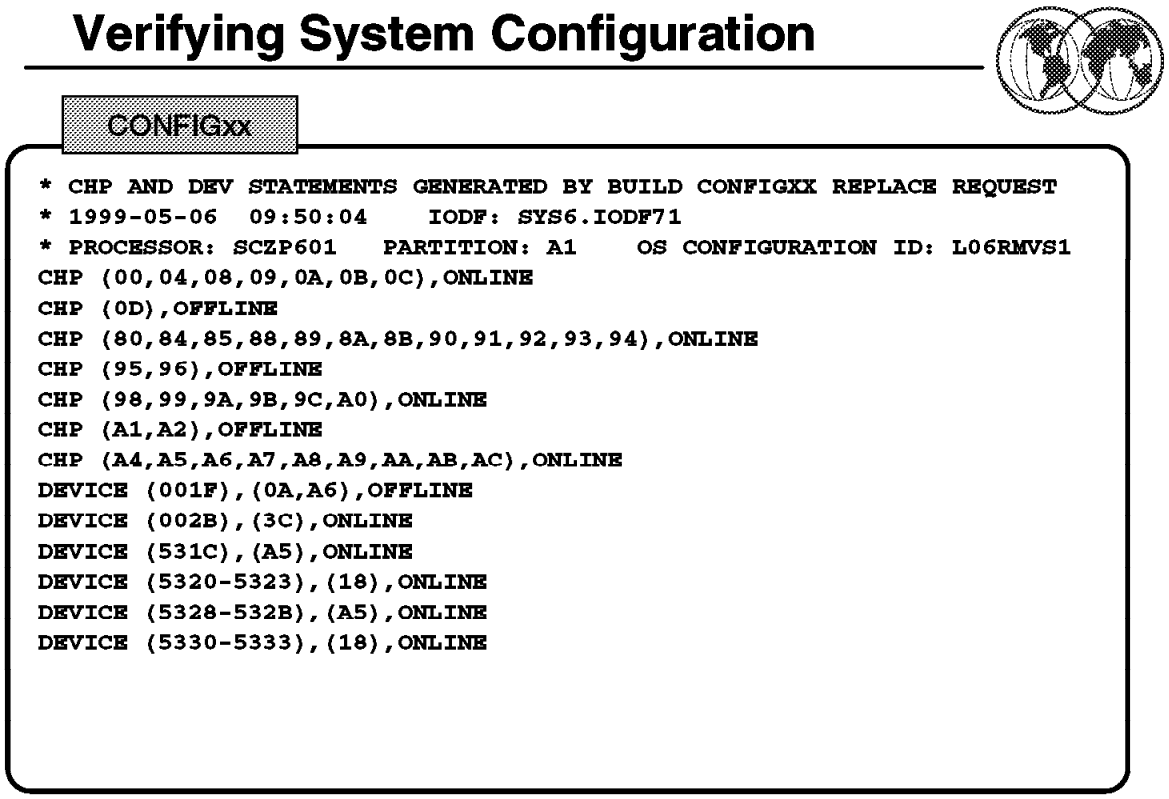


Figure 70. Verify system configuration

1.17.6 How to verify the system configuration

The CONFIGxx parmlib member is a list of records that an installation can use to define a standard configuration of system elements. The system elements include the processors, the expanded storage, vector facilities, storage, channel paths, devices, and volumes. You can use the configuration defined in CONFIGxx in two ways:

- To compare the differences between the current configuration and the standard configuration as defined in a CONFIGxx member. To do that, you use the DISPLAY M=CONFIG(xx) operator command.
- To reconfigure some of the system elements by using the CONFIG operator command with the MEMBER option.

You can create a CONFIGxx parmlib member manually as described in the *OS/390 MVS Initialization and Tuning Reference*, SC28-1752; or through the *Build CONFIGxx member* option using Hardware Configuration Definition (HCD).

To build the CONFIGxx using HCD, you have to do the following:

1. Select Option 2, Activate or process configuration data from the Hardware Configuration panel (remember to specify your production IODF):

```
                          Hardware Configuration

Select one of the following.

2   1.  Define, modify, or view configuration data
    2.  Activate or process configuration data
    3.  Print or compare configuration data
    4.  Create or view graphical configuration report
    5.  Migrate configuration data
    6.  Maintain I/O definition files
    7.  Query supported hardware and installed UIMs
    8.  Getting started with this dialog
    9.  What's new in this release

For options 1 to 5, specify the name of the IODF to be used.

I/O definition file . . . 'SYS6.IODF71'
```

2. Next select Option 6, Activate or verify configuration dynamically:

```
                      OS/390 Release 5 HCD
                  Process Configuration Data
 |
 |
 | Select one of the following tasks.
 |
 | 6_  1.  Build production I/O definition file
 |     2.  Build IOCDS
 |     3.  Build IOCP input data set
 |     4.  Create JES3 initialization stream data
 |     5.  View active configuration
 |     6.  Activate or verify configuration
 |         dynamically
 |     7.  Activate configuration sysplex-wide
 |     8.  Activate switch configuration
 |     9.  Save switch configuration
 |     10. Build OS configuration data set
 |     11. Build and manage S/390 microprocessor
 |         IOCDSs and IPL attributes
```

3. Then select Option 6, Build CONFIGxx member:

```
                  Activate or Verify Configuration
 The currently active IODF matches the hardware I/O
 configuration.  Both hardware and software definitions may be
 changed. Select one of the following tasks.

 6_  1.  Activate new hardware and software configuration.
     2.  Activate software configuration only. Validate
         hardware changes.
     3.  Activate software configuration only.
     4.  Verify active configuration against system.
     5.  Verify target configuration against system.
     6.  Build CONFIGxx member.
```

4. Next specify the system that you want your CONFIGxx member to build from:

```
                 Identify System I/O Configuration
 Specify or revise the following values. Press ENTER to
 continue.

 IODF to be used . . . . : SYS6.IODF71
    Processor ID . . . . . SCZP601   +
    Partition name . . . . A1        +
    OS configuration ID  . LO6RMVS1  +
```

5. Complete the process by specifying the partitioned data set that will contain CONFIGxx and the suffix of the CONFIGxx:

```
                      Build CONFIGxx Member
 Specify or revise the values for the CONFIGxx member. Press ENTER to
 continue.

 Partitioned data set name . 'SYS1.PARMLIB'
 Suffix of CONFIGxx member . 00

 Volume serial number  . . . ______  + (if data set not cataloged)
```

Note: It is recommended that you update the corresponding CONFIGxx member to reflect changes that you have made especially after any dynamic changes to the system elements.

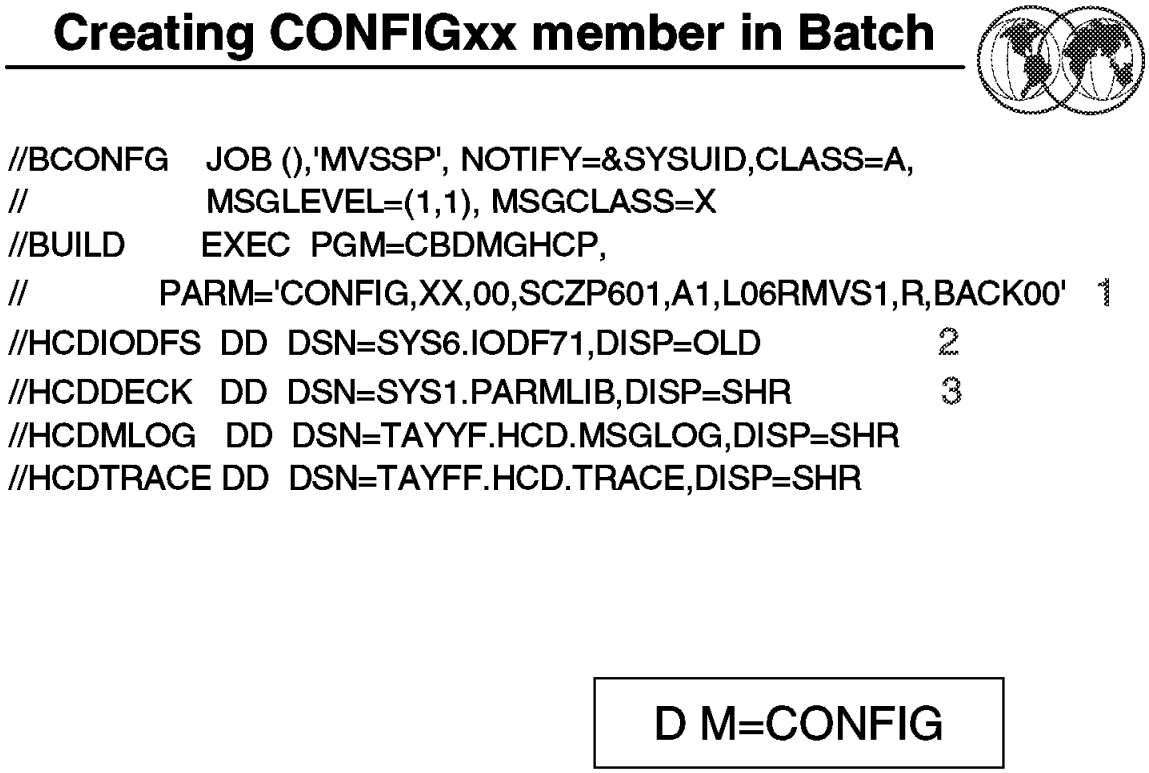


Figure 71. Creating CONFIGxx member in batch

1.17.6.1 Creating CONFIGxx member in batch

You can also use a batch job to create the CONFIGxx member. The JCL above is a sample to accomplish the task:

1 This parameter creates a CONFIG00 member with the Processor ID (SCZP601), Partition ID (A1) and OS configuration ID (L06RMVS1).

2 This specifies the Source IODF.

3 This specifies the output partitioned data set for CONFIGxx.

For more information, see *OS/390 HCD User's Guide*, SC28-1848.

When you issue the `DISPLAY M=CONFIG(xx)` operator command, the system displays the differences between the current configuration and the configuration described in the CONFIGxx parmlib member. The default suffix of the CONFIGxx is 00.

If the existing configuration matches the one specified in the CONFIGxx, the following message is shown:

```
D M=CONFIG
IEF196I IEF237I OFCO ALLOCATED TO SYS00118
IEE252I MEMBER CONFIG00 FOUND IN SYS1.PARMLIB
IEE097I 11.07.34 DEVIATION STATUS 633
               FROM MEMBER CONFIG00
        NO DEVIATION FROM REQUESTED CONFIGURATION
```

On the other hand, if there is any mismatch between the existing configuration and that defined in the CONFIGxx, the following message is shown (note that this is only an example, you might receive something different depending on what is deviated from in the configuration):

```
D M=CONFIG
IEE252I MEMBER CONFIG00 FOUND IN SYS1.PARMLIB
IEE097I 23.15.15 DEVIATION STATUS 663
               FROM MEMBER CONFIG00
CHP          DESIRED  ACTUAL
20           ONLINE   NOT AVAILABLE
DEVICE      DESIRED  ACTUAL
0790,95      ONLINE   OFFLINE
```

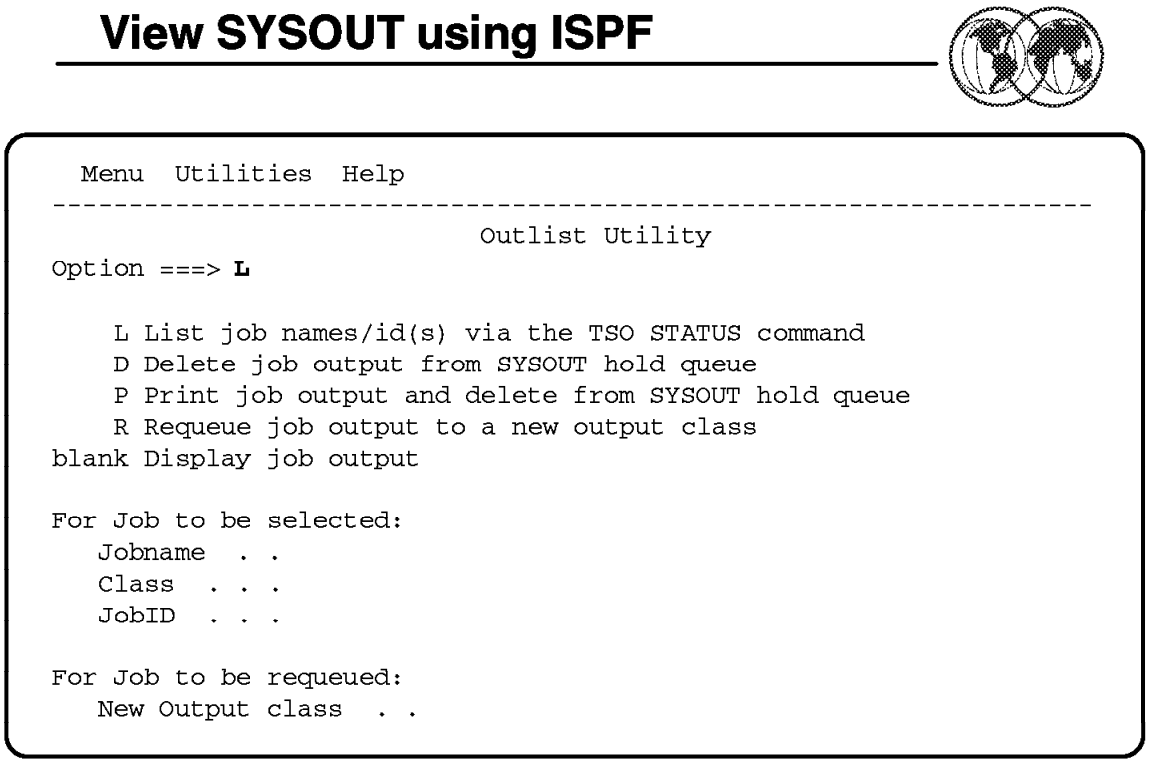


Figure 72. View SYSOUT using ISPF

1.17.7 How to view SYSOUT using ISPF

You can use Option 3.8 from the ISPF Primary Option Menu to view the SYSOUT of your jobs, instead of SDSF.

The list jobs function is used to obtain status information on jobs in your system. To list job status, fill in the following fields of the Outlist Utility panel:

- Enter *L* in the option field.
- Optionally, enter the jobname parameter as follows:
  - If jobname is blank or consists of your user ID plus one character, status is listed for all jobs having jobnames consisting of your user ID followed by one character.

    Note: The jobname must be your user ID or start with your user ID.
  - If jobname is anything else, status is listed for the specified jobname only.

The job status information will be displayed on the lower part of the screen as shown in Figure 73 on page 96. If the list is too long to fit on the screen, *** will be written as the last line. Press ENTER to display the next page of data:

```
For Job to be requeued:
   New Output class  . .

JOB TAYYFB(JOB20684) ON OUTPUT QUEUE
JOB TAYYFA(JOB20683) ON OUTPUT QUEUE
JOB TAYYFC(JOB20685) ON OUTPUT QUEUE
***
```

Figure 73. Job status information display

After you have the list of your jobs in the output queue, you can now display the output for a particular job by entering the Jobname or the Job ID in the appropriate place and pressing ENTER. The panel displayed will be similar to that shown in Figure 74. The Job ID is useful if there are several jobs with the same job name:

```
BROWSE    TAYYF.SPF103.OUTLIST                Line 00000000 Col 001 080
Command ===>                                            Scroll ===> CSR
******************************** Top of Data ***************************
                  J E S 2  J O B  L O G  --  S Y S T E M  S C 5 2  --

13.54.38 JOB20683 ---- THURSDAY,  06 MAY 1999 ----
13.54.38 JOB20683  IRR010I  USERID TAYYF    IS ASSIGNED TO THIS JOB.
13.54.38 JOB20683  ICH70001I TAYYF    LAST ACCESS AT 13:38:05 ON THURSDA
13.54.38 JOB20683  $HASP373 TAYYFA   STARTED - INIT A    - CLASS A - SYS
13.54.38 JOB20683  IEF403I TAYYFA - STARTED - TIME=13.54.38
13.54.45 JOB20683  -                                          --TIMINGS (
13.54.45 JOB20683  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SR
13.54.45 JOB20683  -TAYYFA            APPRSU      00   3047    .02    .0
13.54.45 JOB20683  IEF404I TAYYFA - ENDED - TIME=13.54.45
13.54.45 JOB20683  -TAYYFA   ENDED.  NAME-MVSSP               TOTAL CPU
13.54.45 JOB20683  $HASP395 TAYYFA   ENDED
                      .
                      .
                      .
```

Figure 74. Status display for a particular job

You can also use the requeue job output function to change output from a held SYSOUT queue to some other output class. It is typically used to print the output by directing it to a non-held output class. (Use the list job option if you wish to obtain the names and IDs of currently held jobs.) To requeue job output, fill in the following fields of the Outlist Utility panel:

- Enter R in the option field.
- Enter the job name.
- Enter the class to specify the held SYSOUT queue (which currently contains the job output).
- Enter the job ID if duplicate jobnames exist in the held SYSOUT queue (otherwise, omit).
- Enter the new output class to specify the output class into which the held output will be requeued.

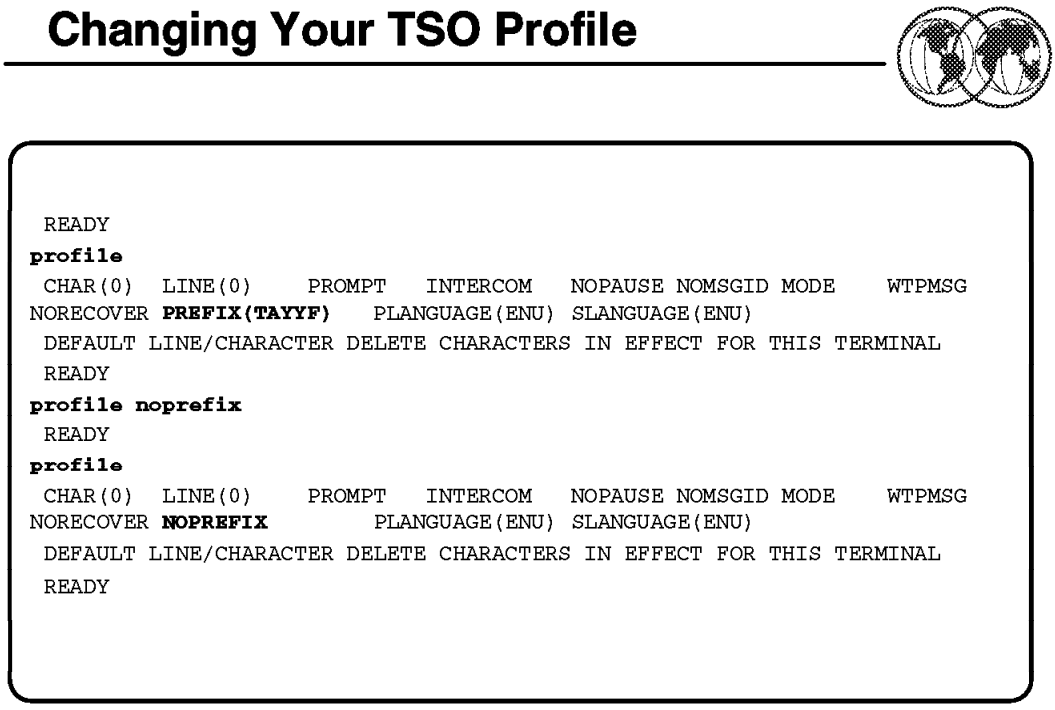


Figure 75. Changing your TSO profile

1.17.8 How to change your TSO user profile

Your user profile defines how you want the system to respond to information sent to or from your terminal. The typical user profile is defined by default values of the PROFILE command's operands. You can use the PROFILE command to display or change your user profile

You can enter the PROFILE command from a TSO prompt (either a READY prompt, or Option 6, ISPF Command Shell in ISPF). You can also enter the PROFILE command in the command line of any ISPF panel by prefixing the command with TSO:

```
TSO PROFILE
```

An example of the profile:

```
CHAR(0) LINE(0) PROMPT INTERCOM NOPAUSE NOMSGID MODE WTPMSG NORECOVER
PREFIX(TAYYF) PLANGUAGE(ENU) SLANGUAGE(ENU)
DEFAULT LINE/CHARACTER DELETE CHARACTERS IN EFFECT FOR THIS TERMINAL
```

Using the PROFILE command, you can specify:

- On terminals without delete keys, the characters that delete a single character or the remainder of a line (CHAR operand, LIST operand).
- Whether you want the system to prompt you for information (PROMPT/NOPROMPT operand).
- Whether you want to receive messages from other users (INTERCOM/NOINTERCOM operand).
- Whether you want to be able to obtain information about messages issued to your terminal while you execute a CLIST (PAUSE/NOPAUSE operand).

- Whether you want to see the message numbers of messages displayed at your terminal (MSGID/NOMSGID operand).
- Whether you want to receive mode messages at your terminal (MODE/NOMODE operand).
- Whether you want to see at your terminal messages issued to you by a program (write-to-programmer messages) (WTPMSG/NOWTPMSG operand).
- Whether you want the EDIT recovery function in effect (RECOVER/NORECOVER operand).
- A user ID or character string to be used as the first qualifier of all non-fully-qualified data set names (PREFIX operand).
- Primary and secondary languages (PLANGUAGE and SLANGUAGE operands) to be used in displaying translated messages, help information, and the TRANSMIT full-screen panel.

One of the most common settings in your profile that you will change is the TSO PREFIX. The TSO PREFIX determines the first qualifier to be added to all data sets that are not fully qualified.

Note: To specify a fully qualified name for a data set, enclose it in quotes. For example, ′SYS1.PARMLIB′.

To change the TSO PREFIX to, for example, TAYYF; enter the following command:

```
PROFILE PREFIX(TAYYF)
```

When you specify a data set without quotes, such as TEST.JCLLIB, the system recognizes it as *TAYYF.TEST.JCLLIB*. You can set the prefix to any value you want but normally it is your user ID.

If you do not want to append a prefix to non-fully-qualified data sets, enter:

```
PROFILE NOPREFIX
```

For more information on the rest of the profile settings, see *OS/390 TSO/E User′s Guide*, SC28-1968.

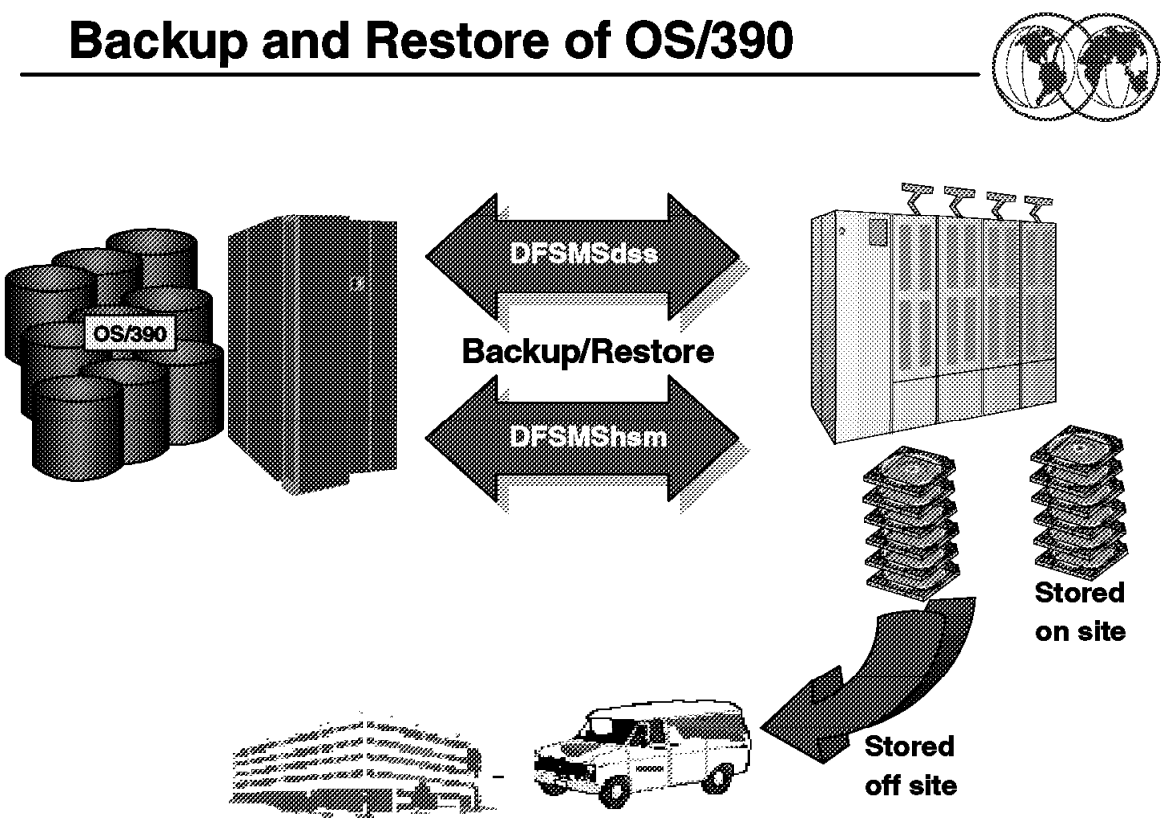


Figure 76. Backup and restore of OS/390

1.17.9 How to back up and restore OS/390

There are times when you have made a change to some system parameters and at the next scheduled IPL, the system simply refuses to start up. You realize that you have made a mistake in the change, and try to go back to the way things were before the change. That's why you need to back up the system.

There are several other situations in which you need to perform recovery actions that besides of messing up a system change:

- When a hardware error occurs; for example, an DASD crash that took out a critical DASD volume.
- An I/O error during IPL due to a change to a system library, for example SYS1.NUCLEUS, or perhaps the master catalog, or any one of the critical system data sets that you need for a successful IPL.
- A corrupted system data set.

Before you start backing up your system, you have to set up a backup/restore strategy if it is not in place in your environment. There are a number of aspects you should consider in your policy:

- *The frequency of the backup*

  Normally, a full system backup is done weekly. This is sufficient if you do not have frequent system changes. For any minor changes before the scheduled backup, you might want to consider back up by data sets instead of a full system backup.

- *The resource needed for backup*

The tape or cartridge drive (for example, IBM or compatible 3420, 3480, or 3490) is considered one of the most commonly used backup/restore devices. The disadvantage of using these devices are they are relatively slow as compared to a direct access device, like the IBM 3390.

- *How many backup sets should you keep*

  You probably want to keep two sets of backup, if your resource permits. One set is kept at your installation, and the other is kept offsite.

You can use the DFSMSdss program ADRDSSU to back up your system. Using ADRASSU, you can plan to have the following typical backup:

- *Full or complete*

  All data in the system is backed up regardless of type, content, or access pattern. Figure 77 shows sample JCL for doing a full volume dump operation.

```
//DSSDUMP JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD    SYSOUT=A
//DASD     DD    UNIT=3390,VOL=SER=MPDLB1,DISP=OLD
//TAPE     DD    UNIT=(3480,2),VOL=SER=(TAPE01,TAPE02),
//  LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=MPDLB1.BACKUP
//SYSIN    DD    *
    DUMP  INDDNAME(DASD1) -
          OUTDDNAME(BACKUP) -
          FULL OPTIMIZE(4) -
          COMPRESS
/*
```

Figure 77. Sample JCL for a full volume dump

- *Incremental*

  An incremental backup only backs up data that has been changed since a previous backup. Managing data this way requires the operating system to maintain some form of change control for individual data sets. Figure 78 shows sample JCL for doing a dump on only data sets changed since last backup.

```
//DSSDUMP JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD    SYSOUT=A
//TAPE     DD    UNIT=(3480,2),VOL=SER=(TAPE01,TAPE02),
//  LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=MPDLB1.BACKUP
//SYSIN    DD    *
    DUMP OUTDD(TAPE) -
         DS(INCL(**) -
            BY((DSCHA EQ 1)))
/*
```

Figure 78. Sample JCL for an incremental dump

- *Selective*

  Data backed up under control of the user. Specific volumes and data sets are selected for backup based on importance or access pattern. Figure 79 on page 101 shows sample JCL for backing up data sets that have a high level qualifier of ISP.

```
//DSSDUMP JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD    SYSOUT=A
//TAPE     DD    UNIT=(3480,2),VOL=SER=(TAPE01,TAPE02),
//  LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=MPRES2.BACKUP
//SYSIN    DD    *
    DUMP OUTDD(TAPE) -
         DS(INCL(ISP.**)) -
         TOL(ENQF) OPTIMIZE(4) COMPRESS
/*
```

Figure 79. Sample JCL for backing up selective data sets

For more information on the usage of ADRDSSU, see *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929.

Chapter 2. Defining subsystems

Subsystem initialization is the process of readying a subsystem for use in the system. IEFSSNxx members of SYS1.PARMLIB contain the definitions for the primary subsystems, such as JES2 or JES3, and the secondary subsystems, such as VPSS and DB2. For detailed information about the data contained in IEFSSNxx members for secondary systems, refer to the installation manual for the specific system.

During system initialization, the defined subsystems are initialized. You should define the primary subsystem (JES) first, because other subsystems, such as DB2, 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 initialized before the primary subsystem. After the primary JES is initialized, then the subsystems are initialized in the order in which the IEFSSNxx parmlib members are specified by the SSN parameter. For example, for SSN=(aa,bb) parmlib member IEFSSNaa would be processed before IEFSSNbb.

Subsystem definitions must be done in order to allow special programs to perform special tasks such as:

- JES2
- JES3
- CICS
- DB2
- IMS

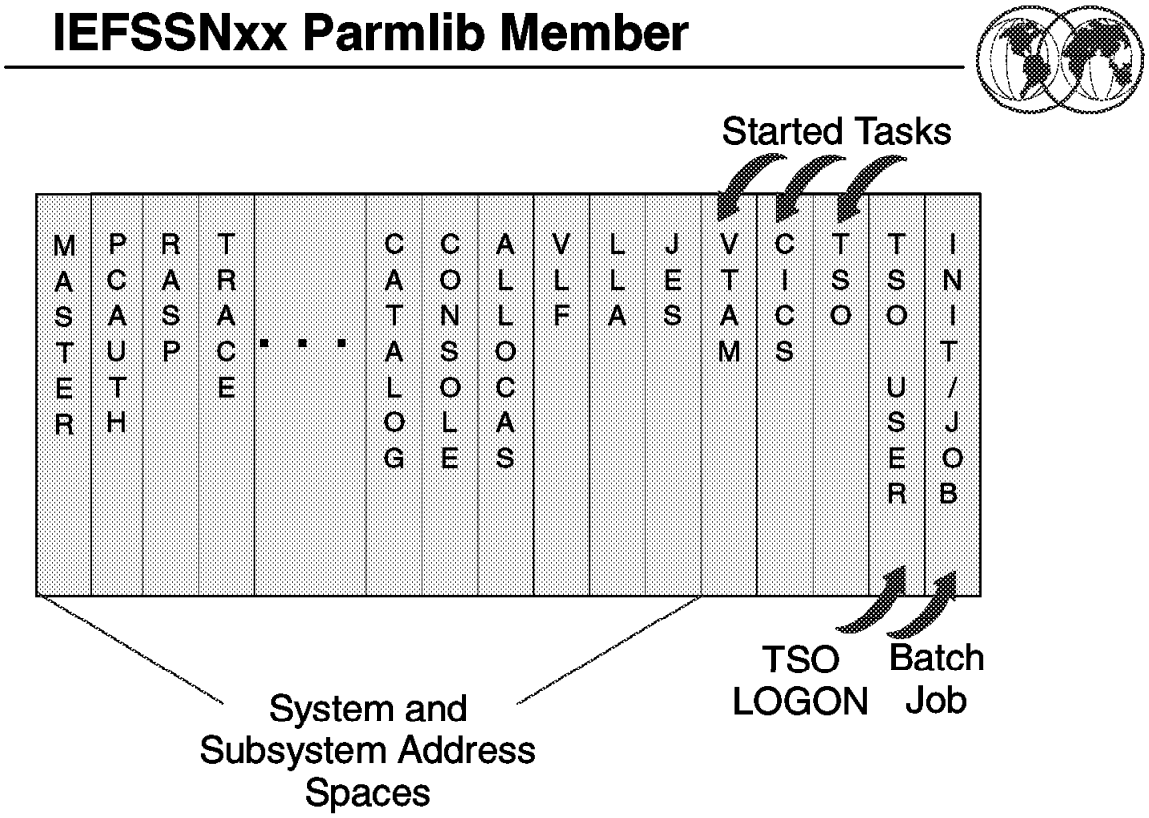


Figure 80. IEFSSNxx member

2.1 IEFSSNxx (subsystem definitions) - keyword parameter form

A subsystem is a service provider that performs one or many functions, but does nothing until it is requested. Subsystems are defined to MVS by the processing of the IEFSSNxx parmlib member during IPL. You can use either the keyword format or positional format of the IEFSSNxx parmlib member. We recommend that you use the keyword format, which allows you to define and dynamically manage your subsystems. You do this by:

- Issuing the IEFSSI macro
- Issuing the SETSSI system command

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.

For information about writing subsystems, see *OS/390 MVS Using the Subsystem Interface*, SC28-1502.

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 can 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.

Note: In general, it is a good idea to make the subsystem name the same as the name of the member of the SYS1.PROCLIB used to start the subsystem. If the name does not match, you may receive error messages when you start the subsystem.

2.1.1.1 Restrictions for IEFSSNxx

The following restrictions apply for IEFSSNxx:

- 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.

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.

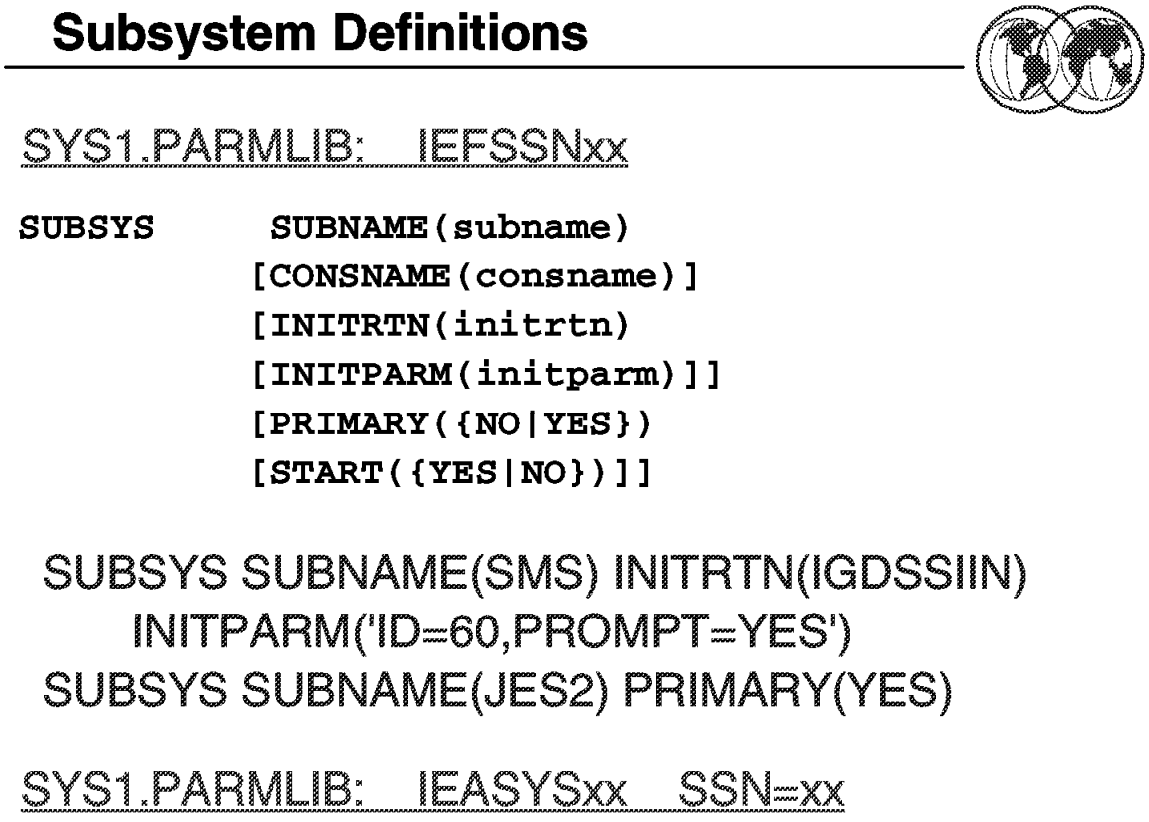


Figure 81. Statements and parameters for IEFSSNxx

2.1.2 Statements and parameters for IEFSSNxx

The storage management subsystem (SMS) is the only subsystem that can be defined before the primary subsystem. Refer to the description of parmlib member IEFSSNxx in *OS/390 MVS Initialization and Tuning Reference*, SC28-1752, for SMS considerations. The statements and parameters for IEFSSNxx are:

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 four characters long; it must begin with an alphabetic or special character (#, @, or $), and the remaining characters (if any) can be alphanumeric or special.

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 two to eight 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 master console.

INITRTN(initrtn) The name of the subsystem initialization routine. This name is optional and can be one to eight characters long. The first character can be either alphabetic or special. The remaining characters can be either alphanumeric

or special. The routine receives control in supervisor state key 0. It must be the name of a program accessible through LINKLIB.

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 quotes, or parentheses are included in the input parameters, the entire parm field must be enclosed in single quotes. If the parm field is enclosed in single quotes, a single quote within the field must be specified as two single quotes. The INITPARM keyword can only be specified if the INITRTN keyword is specified.

PRIMARY({NO|YES}) This parameter indicates 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}) This parameter indicates 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.

2.1.2.1 Example

Define 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)
```

2.1.2.2 Parameter in IEASYSxx (or specified 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 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. If you specify duplicate subsystem names in IEFSSNxx parmlib members, the system issues message IEFJ003I to the SYSLOG, the master 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 *OS/390 MVS Using the Subsystem Interface*, SC28-1502.

2.1.2.3 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 through 80, because the system ignores these columns.
- Comments may appear in columns 1 through 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.
- A statement ends with the beginning of the next valid statement type or end-of-file.
- A statement can be continued even though there is no explicit continuation character.
- 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.

2.1.2.4 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.

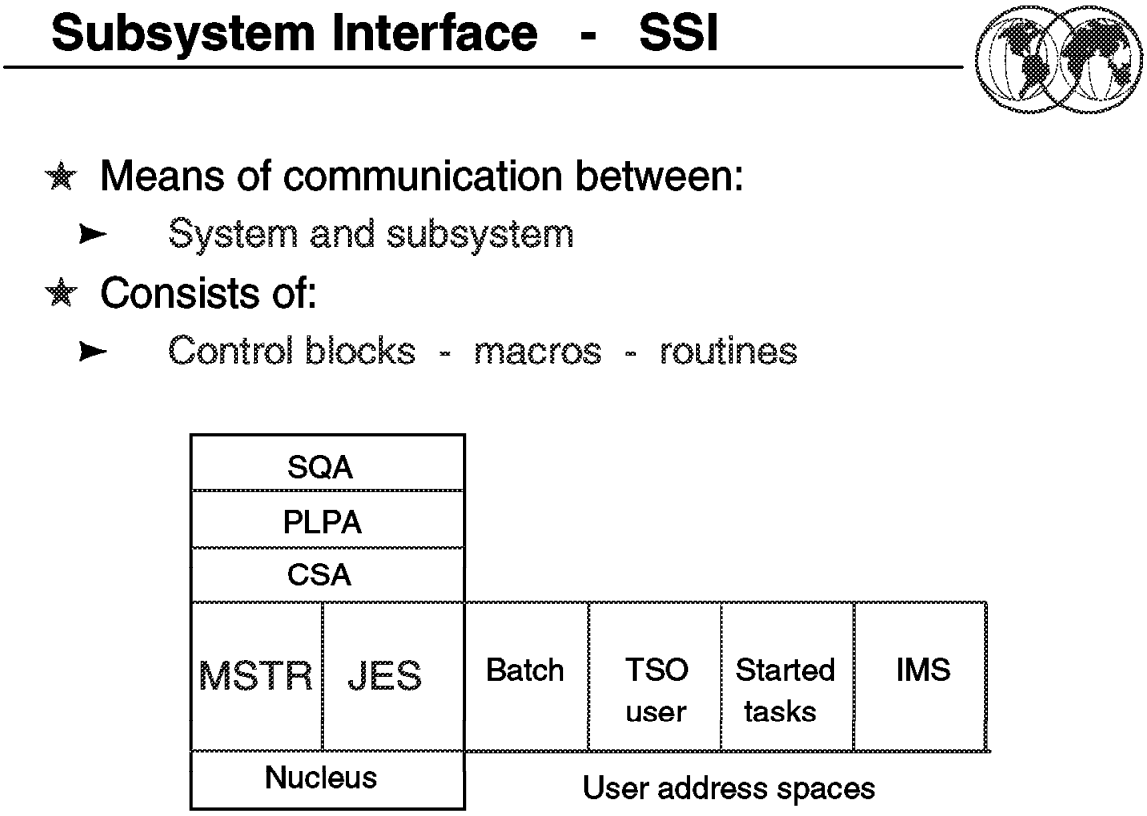


Figure 82. Subsystem interface - SSI

2.2 Subsystem interface - SSI

The SSI is an interface that provides communication between MVS and JES through the IEFSSREQ macro. The SSI is the interface used by routines (IBM, vendor, or installation-written) to request services of, or to pass information to, subsystems. An installation can design its own subsystem and use the SSI to monitor subsystem requests. An installation can also use the SSI to request services from IBM-supplied subsystems. The SSI acts only as a mechanism for transferring control and data between a requestor and the subsystem; it does not perform any subsystem functions itself.

MVS functions issue the IEFSSREQ macro to invoke JES. The calling routine uses the subsystem option block (SSOB) and subsystem identification block (SSIB) to identify the required processing to JES3. The calling routine uses the IEFSSREQ macro to pass the SSIB and SSOB to JES.

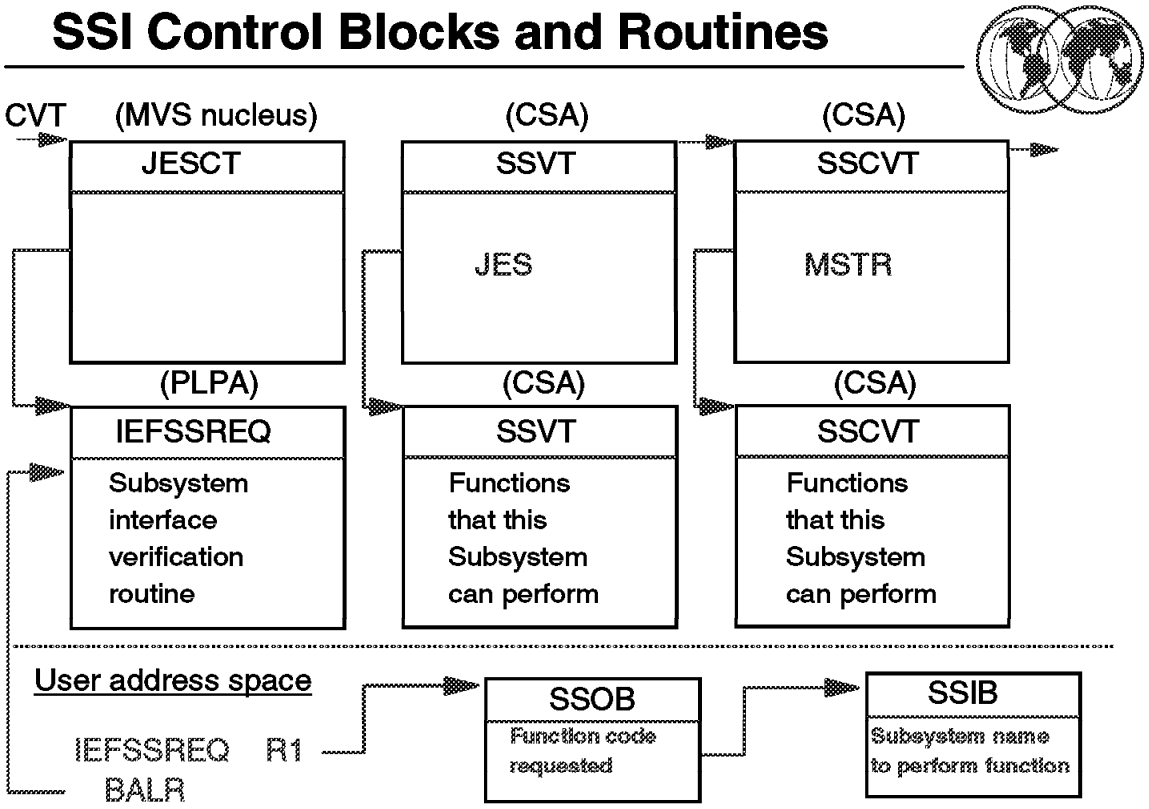


Figure 83. SSI control blocks and routines

2.2.1 SSI control blocks and routines

The MVS nucleus contains a control block named the JES control table (JESCT). This block contains a pointer to the primary subsystem (JES3) communication vector table (SSCVT), which is located in CSA. There will always be at least two SSCVTs, one of which represents the MVS master subsystem.

The macro that provides communication between MVS and JES3 is the IEFSSREQ macro. MVS functions issue this macro to invoke JES3. The calling routine uses the subsystem option block (SSOB) and subsystem identification block (SSIB) to identify the required processing to JES3. The calling routine uses the IEFSSREQ macro to pass the SSIB and SSOB to JES3. The control blocks involved are:

JESCT JES control table (JESCT). A control block in the MVS nucleus that contains information used by subsystem interface routines.

SSCVT Primary subsystem communication vector table (SSCVT). This control block is the common storage area that contains information used by the subsystem interface routines.

SSVT The subsystem vector table (SSVT). The SSVT resides in the JES3 common service area (CSA) and contains the following information:

SSOB Subsystem options block (SSOB). The control block into which MVS places a function code when communicating with JES3 over the subsystem interface. The function code identifies a requested service.

SSIB Subsystem identification block (SSIB). The control block into which MVS places the name of the subsystem to which it is directing a request over the subsystem interface.

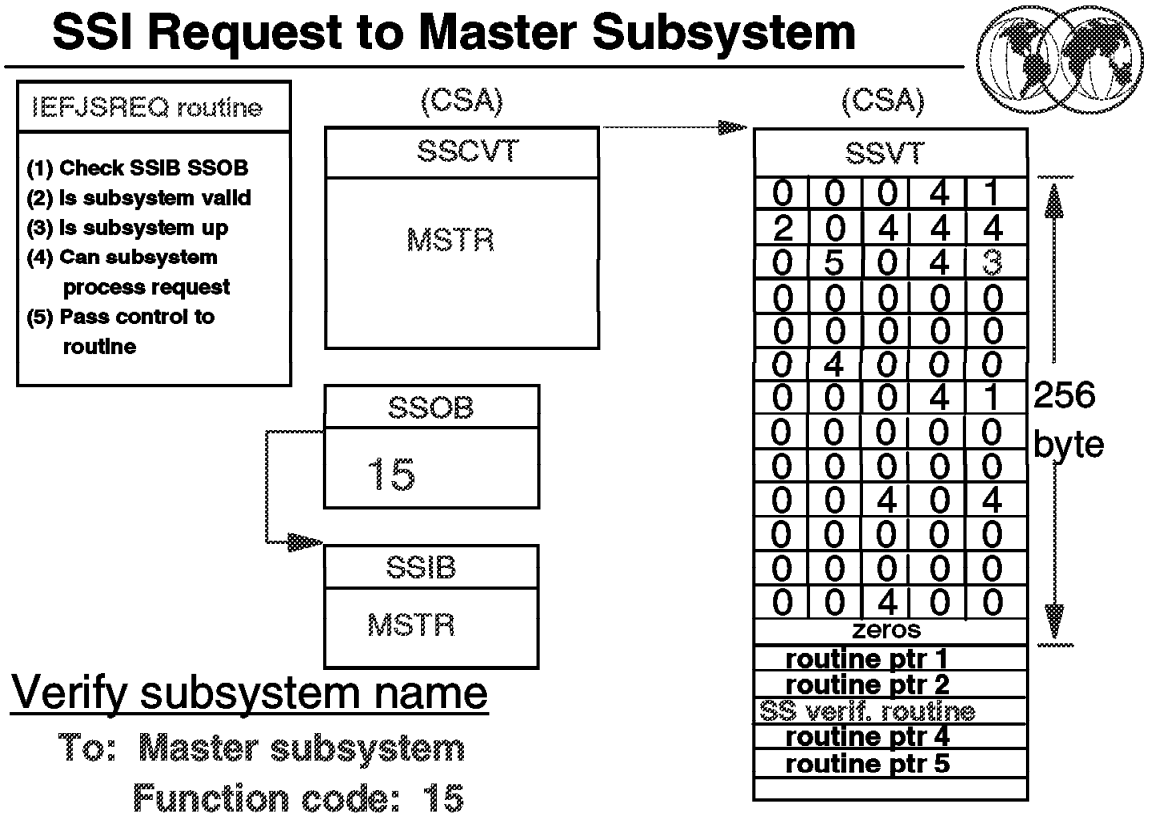


Figure 84. SSI request to master subsystem

2.2.2 SSI request to master subsystem

The IEFJSREQ routine checks the validity of the SSOB and SSIB; the request routine determines that the target subsystem exists and is started. It next uses the function code to determine if the subsystem performs the requested function and to derive the address of a routine to which the request is be passed. The SSOB contains the function code requested by the caller and the SSIB contains the name of the subsystem to perform the function requested.

There are 15 SSI functions supported by the master subsystem but several of the routines perform more than one function.

The SSVT contains a matrix of the function codes and in the visual the first function code supported by the MSTR subsystem is function code 4. The number 4 shown in the fourth position indicates that the routine that gets control is the fourth routine pointer (ptr) shown after the function code matrix in the SSVT.

Function code 15, is used by the master subsystem (MSTR) to verify the subsystem (JES) name when JES initializes and is the third routine pointer as function code 15 has a 3 in the matrix.

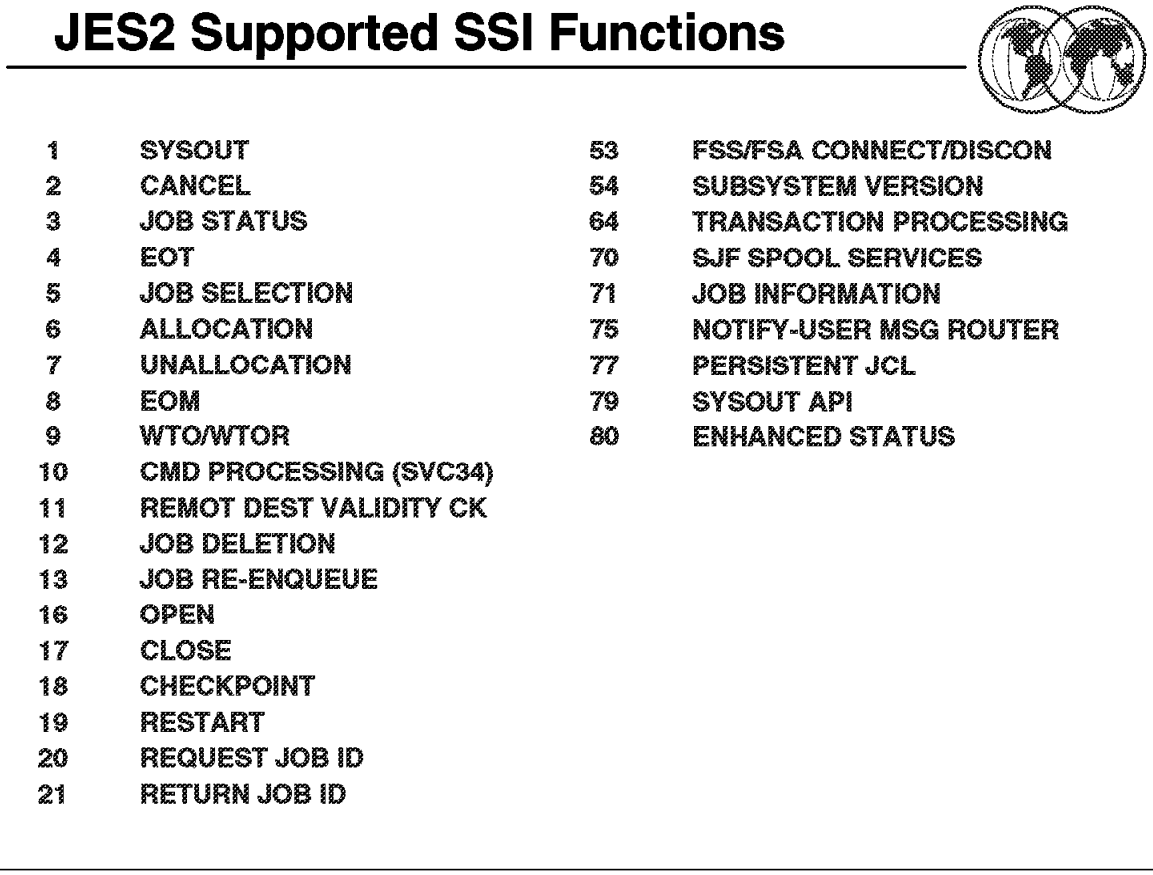
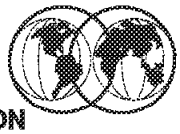


Figure 85. JES2 supported SSI funcitons

2.2.3 JES2 supported SSI functions

These are the functions codes supported by JES2 as a subsystem.

JES3 Supported SSI Functions



1	SYSOUT	24	COMMON ALLOCATION
2	CANCEL	25	COMMON UNALLOCATION
3	JOB STATUS	26	CHANGE DDNAME
4	EOT	27	CHANGE ENQ USE ATTRIBUTE
5	JOB SELECTION	28	DDR CANDIDATE SELECT
6	ALLOCATION	29	DDR CANDIDATE VERIFY
7	UNALLOCATION	30	DDR SWAP NOTIFICATION
8	EOM	31	DDR SWAP COMPLETE
9	WTO/WTOR	32	SVC34 COMMAND FAIL
10	CMD PROCESSING (SVC34)	34	WRITE TO LOG
11	REMOT DEST VALIDITY CK	40	EARLY VOLUME RELEASE
12	JOB DELETION	53	FSS/FSA CONNECT/DISCON
13	JOB RE-ENQUEUE	54	SUBSYSTEM VERSION
16	OPEN	56	SMS TO JES3 COMMUNICATION
17	CLOSE	62	BDT SUBSYSTEM
18	CHECKPOINT	64	TRANSACTION PROCESSING
19	RESTART	72	VARY PATH CALL
20	REQUEST JOB ID	75	NOTIFY-USER MSG ROUTER
21	RETURN JOB ID	77	PERSISTENT JCL
22	STEP INITIATION	79	SYSOUT API
23	DYNAMIC ALLOCATION	80	ENHANCED STATUS

Figure 86. JES3 supported SSI functions

2.2.4 JES3 supported SSI functions

These are the functions codes supported by JES3 as a subsystem.

Chapter 3. Job Management

OS/390 Job Management in an OS/390 system uses a job entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by OS/390, and to control their output processing. JES provides supplementary job management, data management, and task management functions such as: scheduling, control of job flow, and spooling. JES is designed to provide efficient spooling, scheduling, and management facilities for the OS/390 system.

For a program to execute on the computer and perform the work it is designed to do, the program must be processed by the operating system. The operating system consists of a base control program (BCP) with a job entry subsystem (JES2 or JES3) and DFSMSdfp installed with it.

For the operating system to process a program, programmers must perform certain job control tasks. These tasks are performed through the job control statements, (JCL). The job control tasks are performed by the JES and are:

- Entering jobs
- Processing jobs
- Requesting resources

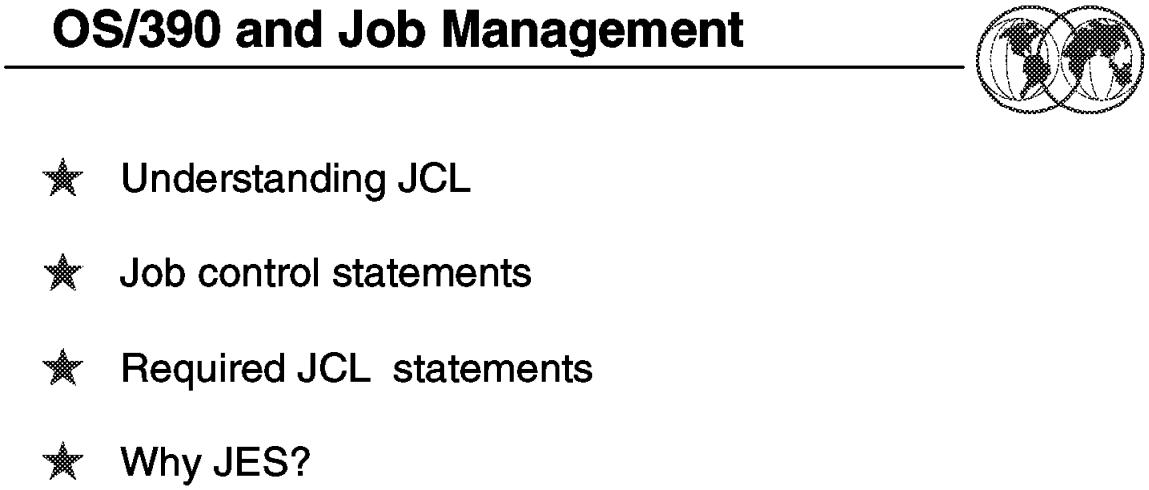


Figure 87. OS/390 and job management

3.1 OS/390 and job management

A major goal of an operating system is to process jobs while making the best use of system resources. To obtain that goal, the operating system does *resource management*, which consists of the following:

- Before job processing, reserve input and output resources for jobs
- During job processing, manage resources such as processors and storage
- After job processing, free all resources used by the completed jobs, making the resources available to other jobs

At any instant, a number of jobs can be in various stages of preparation, processing, and post-processing activity. To use resources efficiently, the operating system distributes jobs into queues, to wait for needed resources, according to their stages such as conversion queue, waiting execution queue, processing queue, output queue, and so forth. The function of keeping track of which job is in which queue is called *workflow management*.

The MVS system divides the management of jobs and resources with a Job Entry Subsystem (JES). JES does job management and resource management before and after job execution, while MVS does it during job execution. The JES receives jobs into MVS, schedules them for processing by MVS, and controls their output processing.

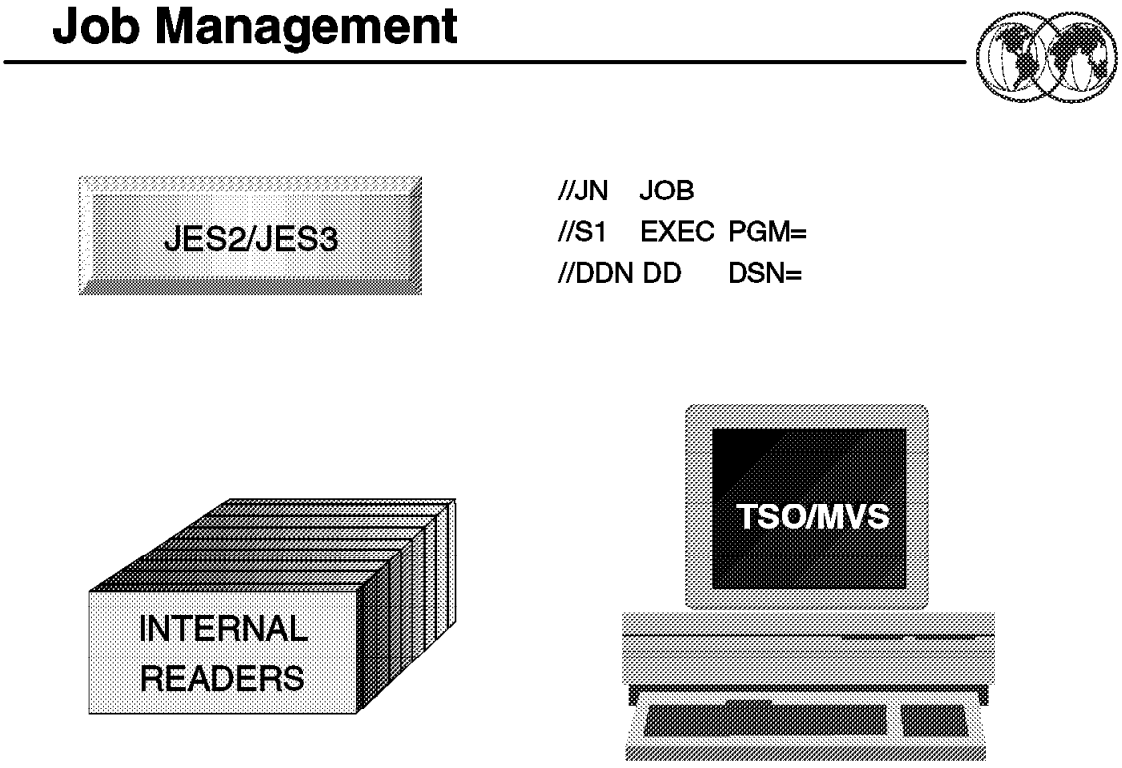


Figure 88. Job management

3.1.1 Understanding JCL

To get your MVS system to do work for you, you must describe to the system the work you want done and the resources you will need. You use Job Control Language (JCL) to provide this information to MVS.

One way of thinking about JCL is to compare it to a menu in a restaurant.

If you are a customer at a restaurant, you and the other customers don't just walk into the kitchen and start cooking your own dinners, that would defeat the very purpose of going to a restaurant. Instead, from a menu describing all the restaurant has to offer, you select items to make up an order, specifying which entrees you want, which salad dressing you prefer, and any other special requests you have. You then ask the waiter to take your order to the kitchen.

In the kitchen, a team of chefs divides up the work and the appropriate ingredients in order to prepare each dish as quickly and efficiently as possible. While the meals are being prepared, you and your friends can ignore what's going on in the kitchen, engaging instead in dinner conversation, catching up on the latest news. When the waiter brings your meal out, you concentrate on your enjoyment of the meal.

How does this relate to JCL?

Now imagine yourself back at the office using your OS/390 system, and think of JCL as the menu. In the same way that you and the other diners select items from the menu and place orders for the waiter

to take to the team of chefs, you and other OS/390 users use JCL to define work requests (called jobs), and use a job entry subsystem (JES) to submit those jobs to OS/390.

Using the information that you and the other users provide with JCL statements, OS/390 allocates the resources needed to complete all of your jobs, just as the kitchen chefs divided up the work to prepare the orders of all the customers.

And just as the chefs worked in the kitchen while you and the other diners devoted your attention to what was going on at your tables, OS/390 completes the submitted jobs in the background of the system, enabling you and the other users to continue working on other activities in the foreground.

And just as the waiter conveys the results of the chef's work to you, JES presents the output of the jobs to you.

Figure 89 on page 119 shows an overview of the job-submission process. The user performs the parts on the left side of the figure, and the system performs the parts on the right. In this figure, OS/390 and JES comprise the "system."

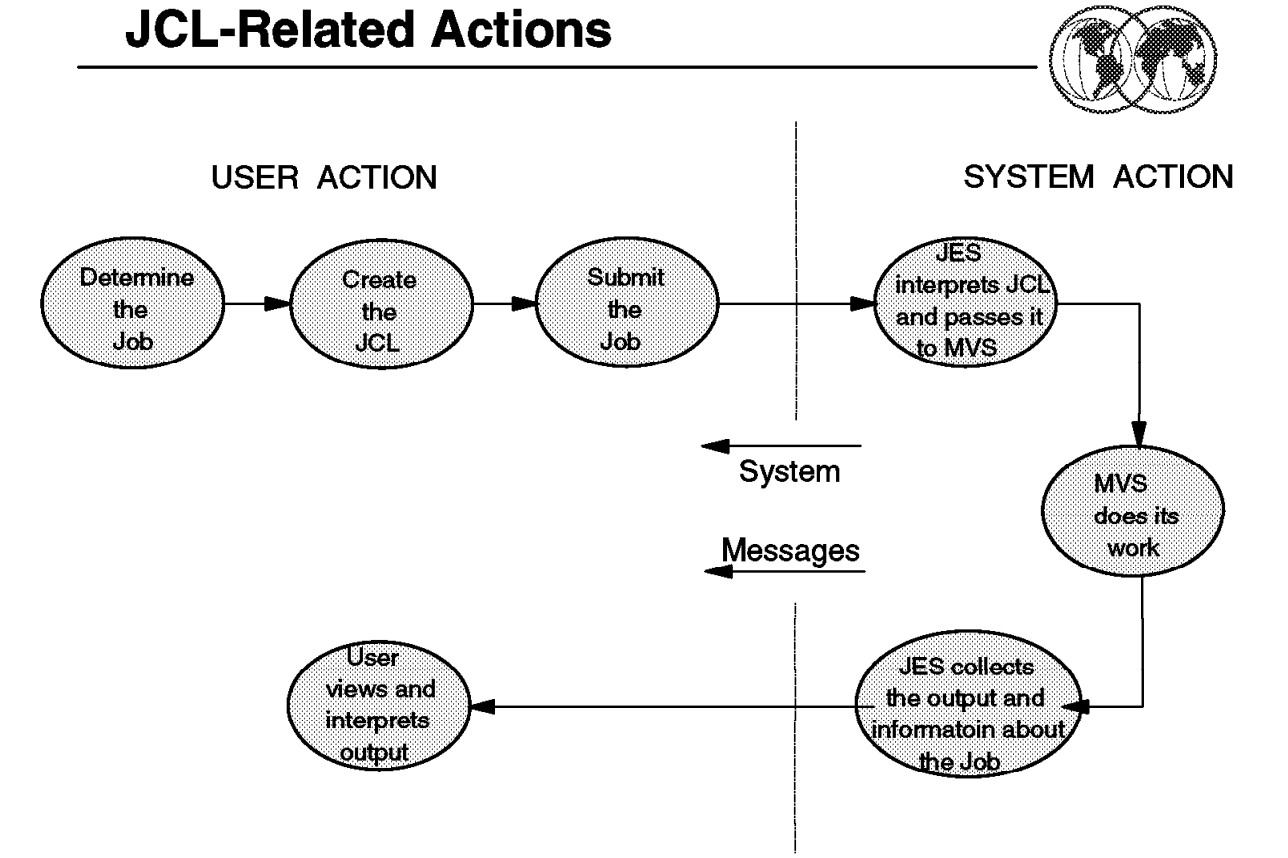


Figure 89. JCL-related actions (user and MVS)

3.1.2 Job control statements

For every job that you submit, you need to tell to MVS where to find the appropriate input, how to process that input (that is, what program or programs to run), and what to do with the resulting output.

You use JCL to convey this information to MVS through a set of statements known as job control statements. The set of job control statements is quite large, enabling you to provide a great deal of information to MVS.

Most jobs, however, can be run using a very small subset of these control statements. Once you become familiar with the characteristics of the jobs you typically run, you may find that you need to know the details of only some of the control statements.

Within each job, the control statements are grouped into job steps. A job step consists of all the control statements needed to run one program. If a job needs to run more than one program, the job would contain a different job step for each of those programs. The job control statements consist of:

- JCL statements
- JES2 control statements
- JES3 control statements

A JCL statement:

- Consists of one or more records 80 bytes length.
- Is coded from column 1 to column 71, in uppercase.
- Starts with a *//* at the beginning of the line.

- A commentary line begins with a ***//****.
- A line with only *//* and blanks indicates end of job.
- A comma indicates that the statement has continuation.
- A continuation of a statement from a previous line must start from column 4 to 16.
- Comments must be separated from operators/parameters by at least one blank.

3.1.2.1 Required control statements

Every job must contain a minimum of the following two types of control statements:

- A job statement, to mark the beginning of a job and assign a name to the job.

  ```
  //jobname JOB .............
  ```

  The job statement is also used to provide certain administrative information, including security, accounting, and identification information. The required job statement parameters depends on the JES customization. Every job has one (and only one) job statement.

- An EXEC (execute) statement, to mark the beginning of a job step, to assign a name to the step, and to identify the program or procedure to be executed in the step.

  ```
  //stepname EXEC .............
  ```

  You can add various parameters to the EXEC statement to customize the way the program executes. Every job has at least one EXEC statement.

In addition to the JOB and EXEC statements, jobs usually contain:

- One or more DD (data definition) statements, to identify and describe the input and output data to be used in the step.

  ```
  //ddname DD .............
  ```

  The DD statement may be used to request a existing data set, to define a new data set, to define a temporary data set, or to define and specify the characteristics of the output.

You can create your own JCL by using the TSO/ISPF **Edit** option. When you have finished entering the JCL into the data set, submit the job by entering the SUBMIT command from the ISPF EDIT command line, as shown in the following example.

```
EDIT ---- userid.SORT.JCL -------------------------- LINE 00000000 COL 001 080
COMMAND ===> SUBMIT                                      SCROLL ===> CSR
******************************** TOP OF DATA  ********************************
//jobname  JOB 'accounting data'
//stepname EXEC 'parameter'
//ddname   DD   'data parameter'
//                    .
//                    .
```

Figure 90. Submit the JCL to the System as a Job

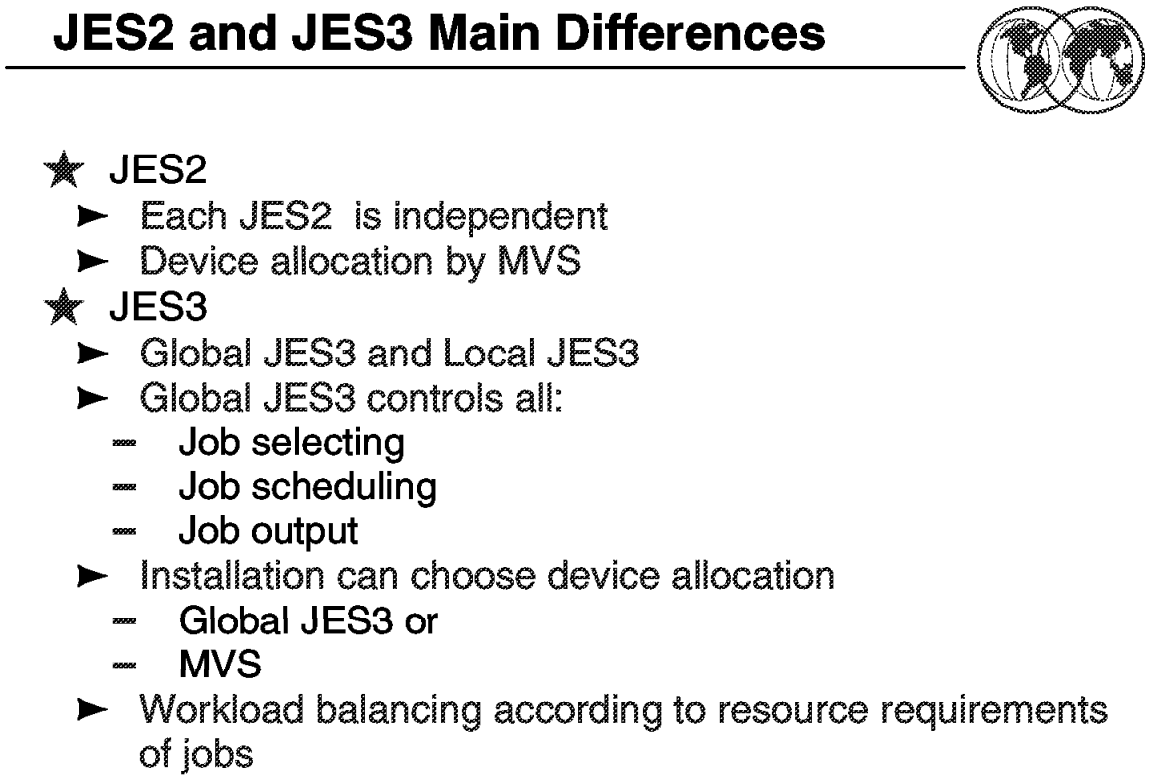


Figure 91. JES2 and JES3 main differences

3.2 JES2 and JES3 main differences

IBM provides two JESs from which to choose: JES2 and JES3. In an installation that has only one MVS image, JES2 and JES3 perform similar functions. That is, they read jobs into the system, convert them to internal machine-readable form, select them for processing, process their output, and purge them from the system. However, for an installation that has more than one MVS image in a configuration, there are noticeable differences in how JES2 and JES2 work.

Each JES2 in each MVS image exercises independent control over its job processing functions. That is, within the configuration, each JES2 processor controls its own job input, job scheduling, and job output processing.

In contrast, JES3 exercises centralized control over its processing functions through a single global JES3 processor. The global processor provides all job selection, scheduling, and device allocation functions for all the other JES3 systems. The centralized control that JES3 exercises provides increased job scheduling control, deadline scheduling capabilities, and increased control by providing its own device allocation.

To better understand the functional differences between JES2 and JES3, refer to *OS/390 JES2 Initialization and Tuning Guide*, SC28-1791, and *OS/390 JES3 Initialization and Tuning Guide*, SC28-1802.

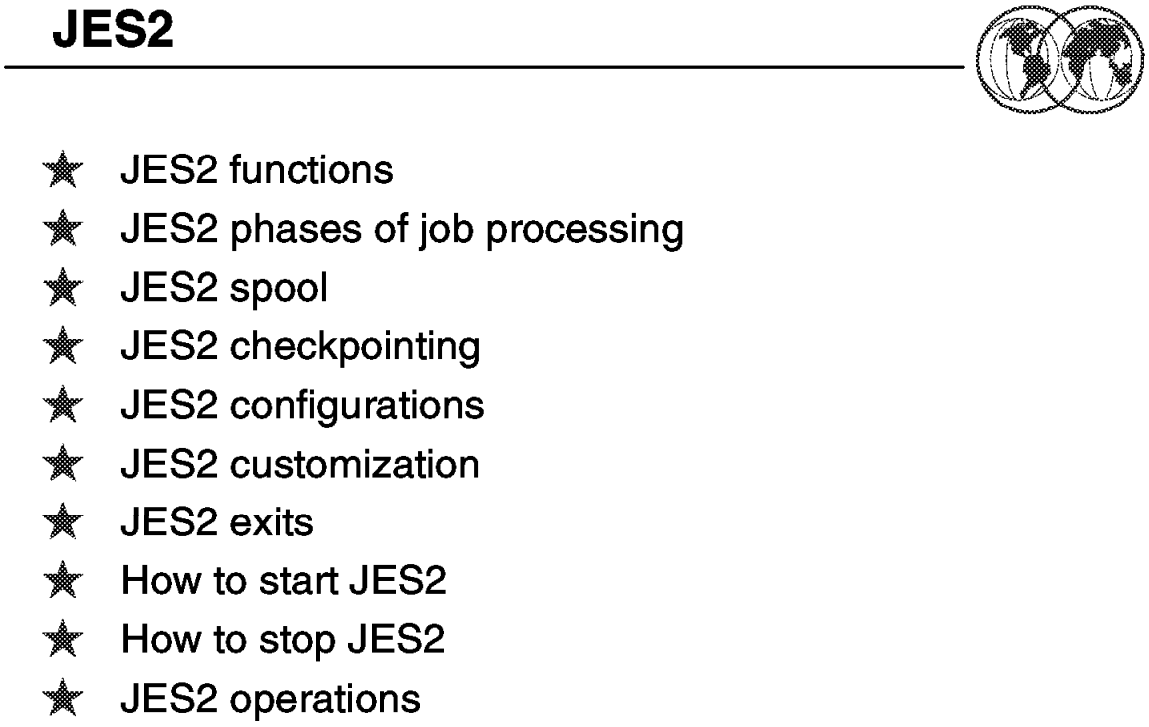


Figure 92. JES2

3.3 JES2

JES2 is descended from Houston automatic spooling priority (HASP). HASP is defined as a computer program that provides supplementary job management, data management, and task management functions such as scheduling, control of job flow, and spooling. HASP remains within JES2 as the prefix of most module names and the prefix of messages sent by JES2 to the operator.

JES2, or Job Entry Subsystem 2, is a functional extension of the HASP II program that receives jobs into the system and processes all output data produced by the job. So what does all that mean? Simply stated, JES2 is the component of OS/390 that provides the necessary functions to get jobs into, and output out of, the OS/390 system. It is designed to provide efficient spooling, scheduling, and management facilities for the OS/390 operating system.

You can run OS/390 in your installation in many processing configurations that range from a single MVS image with a single JES2 that is completely isolated from other processing systems to one that is a part of a worldwide processing network. The choice of configuration complexity is yours and generally is a dynamic one that grows as your business needs grow.

The visual shows the topics discussed in this chapter about JES2.

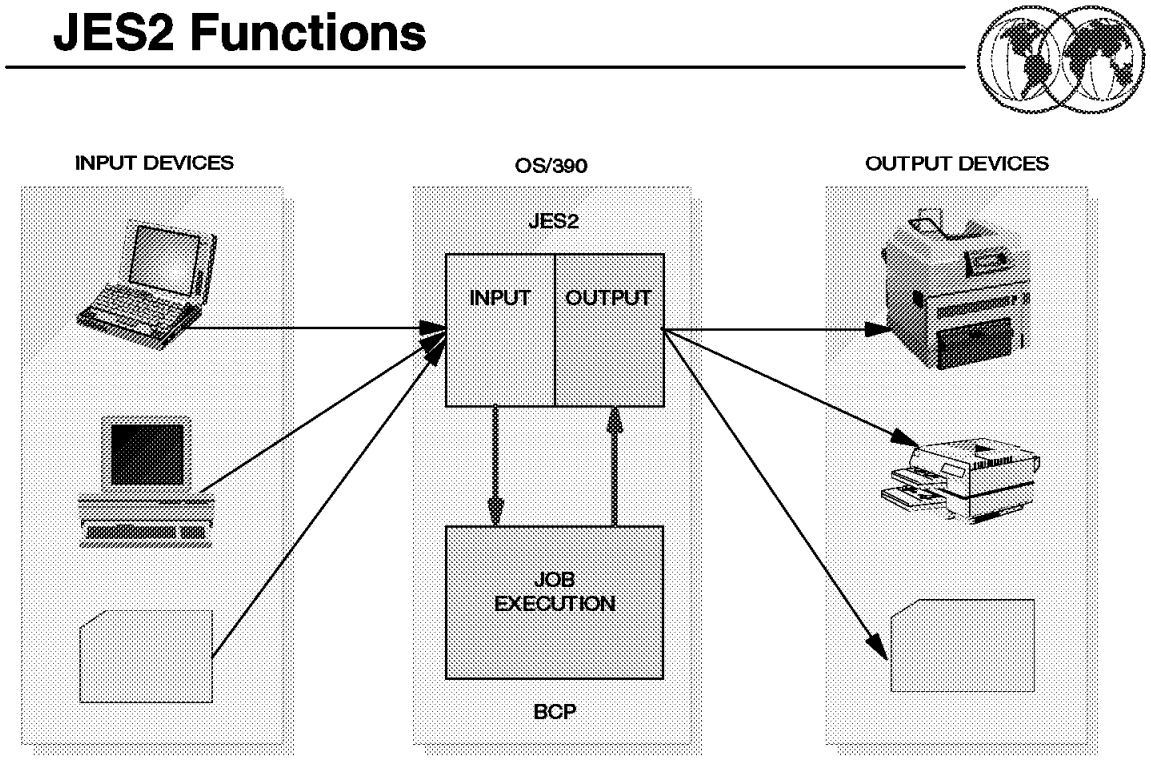


Figure 93. Relationship of JES2 to BCP

3.3.1 JES2 functions

To manage job input/output, JES2 controls a number of functional areas, all of which you can customize to your installation′s need. Some of the major functional areas and processing capabilities are:

- Getting work into and out of MVS (input/output control):

  JES2 controls output devices such as local and remote printers, punches, and card readers. You define each device to JES2 using JES2 initialization statements. All are directly under JES2′s control with the exception of those printers that operates under the Print Services Facility (PSF). Printed and punched output can be routed to a variety of devices in multiple locations. The control JES2 exercises over its printers ranges from the job output classes and job names from which the printer can select work to such specifications such as the forms on which the output is printed. This control allows the system programmer to establish the job output environment most efficiently without causing unnecessary printer backlog or operator intervention. Through JES2, the installation defines the job input classes, reader specifications, and output device specifications. As a result, JES2 is the central point of control over both the job entry and job exit phases of data processing.

- Maximizing efficiency through job selection and scheduling:

  JES2 allows the installation to define work selection criteria. It can be specific for each output device (local and remote printers and punches and offload devices). The work selection criteria are defined in the device initialization statements and can be changed dynamically using JES2 commands. You can define:

- Specification of job and output characteristics that JES2 considers when selecting work for an output device
- Priority of the selection characteristics
- Characteristics of the printer and job that must match exactly

A job's output is grouped based on the data sets output requirements. The requirements may be defined in the job's JCL or by JES2-supplied defaults.

When selecting work to be processed on a device, JES2 compares the device characteristics with the output requirements. If they match, JES2 sends the output to the device. If they do not match, JES2 does not select the work for outpt until the operator changes the device characteristics or the output requirements.

- Offloading work and backing up system workload:

  JES2 gives your installation the capability to offload data from the spool and later reload data to the spool. This is useful if you need to:

  - Preserve jobs and SYSOUT across a cold start
  - Migrate your installation to another release of JES2
  - Convert to another DASD type for spool
  - Archive system jobs and SYSOUT
  - Relieve a full-spool condition during high-use periods
  - Provide a backup for spool data sets
  - Back up network connections

  JES2 offers many selection criteria to limit the spool offload operation. These selection criteria can be changed by operator command, according to initial specification in the JES2 initialization statements.

- System security:

  JES2 provides a basic level of security for resources through initialization statements. That control can be broadened by implementing several JES2 exits available for this purpose. A more complete security policy can be implemented with System Security Facility (SAF) and a security product such as RACF.

- Supporting advanced function printers (AFPs)

JES2 is responsible for all phases of job processing and monitors the processing phase. The base control program (BCP) and JES2 share responsibility in the OS/390 system. JES2 is responsible for job entry (input), the base control program for device allocation and actual job running, and finally JES2 for job exit (output).

Figure 94 on page 125 presents a view of the job phases.

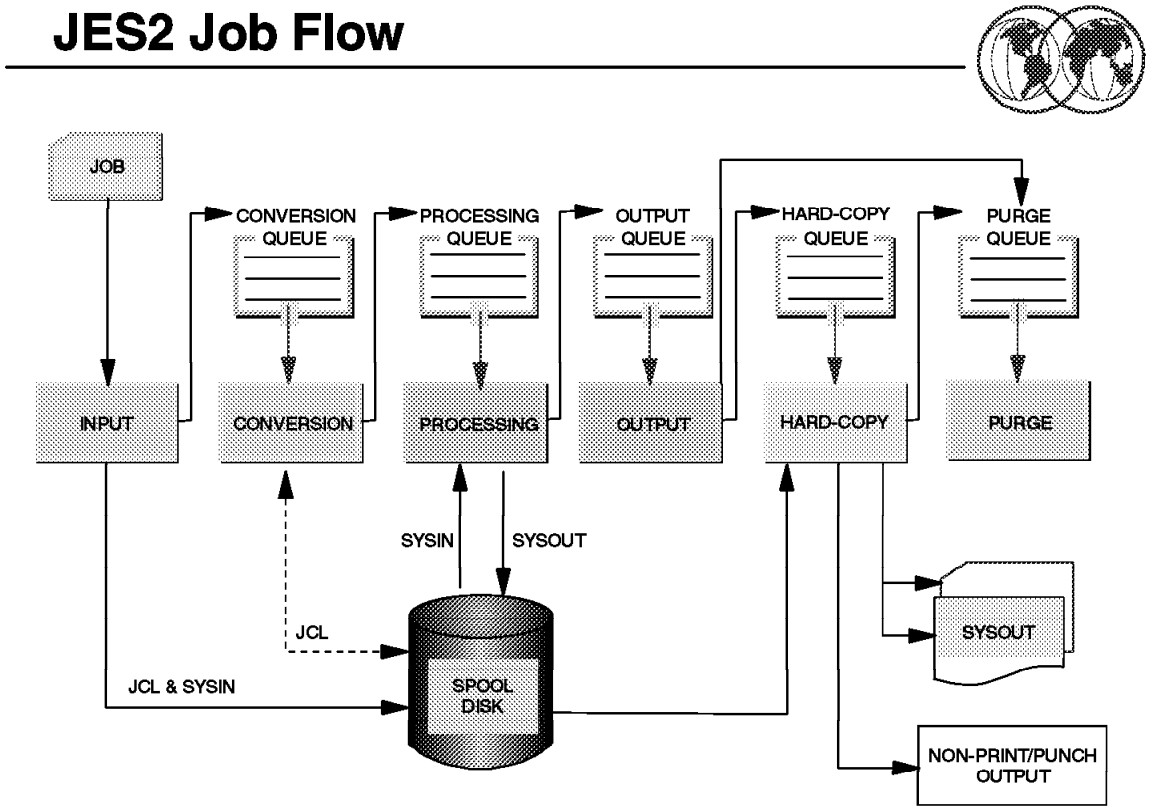


Figure 94. JES2 job fLow

3.3.2 JES2 - Phases of job processing

The job queues contain jobs:

- Waiting to run - conversion queue
- Currently running - execution queue
- Waiting for their output to be produced - output queue
- Having their output produced - hard-copy (print/punch) queue
- Waiting to be purged from the system - purge queue

The six phases of job processing are as follows:

1. **Input phase**

   JES2 accepts jobs, in the form of an input stream, from input devices, internal readers, other nodes in a job entry network, and from other programs.

   The internal reader is a program that other programs can use to submit jobs, control statements, and commands to JES2. Any job running in MVS can use an internal reader to pass an input stream to JES2. JES2 can receive multiple jobs simultaneously through multiple internal readers. MVS uses internal readers, allocated during system initialization, to pass to JES2 the job control language (JCL) for started tasks, START and MOUNT commands, and TSO LOGON requests.

   The system programmer defines internal readers to be used to process all batch jobs other than STCs and TSO requests. These internal readers are defined in JES2 initialization statements and JES2 allocates them during JES2 initialization processing. The internal readers for batch jobs can be used for STCs and TSO requests, if not processing jobs.

JES2 reads the input stream and assigns a job identifier to each JOB JCL statement. JES2 places the job′s JCL, optional JES2 control statements, and SYSIN data onto DASD data sets called spool data sets. JES2 then selects jobs from the spool data sets for processing and subsequent running.

2. **Conversion phase**

   JES2 uses a converter program to analyze each job′s JCL statements. The converter takes the job′s JCL and merges it with JCL from a procedure library. The procedure library can be defined in the JCLLIB JCL statement, or system/user procedure libraries can be defined in the PROCxx DD statement of the JES2 startup procedure. Then, JES2 converts the composite JCL into converter/interpreter text that both JES2 and the job scheduler functions of MVS can recognize. Next, JES2 stores the converter/interpreter text on the spool data set. If JES2 detects any JCL errors, it issues messages, and the job is queued for output processing rather than run. If there are no errors, JES2 queues the job for execution.

3. **Processing phase**

   In the processing phase, JES2 responds to requests for jobs from the MVS initiators. An initiator is a system program that is controlled by JES or by WLM (in goal mode, with WLM Batch Initiator Management).

   JES2 initiators are defined to JES2 through JES2 initialization statements. JES2 initiators are started by the operator or by JES2 automatically when the system initializes. The installation associates each initiator with one or more job classes in order to obtain an efficient use of available system resources. Initiators select jobs whose classes match the initiator assigned class, obeying the priority of the queued jobs.

   WLM initiators are started by the system automatically based on the performance goals, relative importance of the batch workload, and the capacity of the system to do more work. The initiators select jobs based on their service class and the order they were made available for execution. Jobs are routed to WLM initiators via a JOBCLASS JES2 initialization statement.

   In goal mode, with WLM Batch Management, a system can have WLM initiators and/or JES2 initiators.

   After a job is selected, the initiator invokes the interpreter to build control blocks from the converter/interpreter text that the converter created for the job. The initiator then allocates the resources specified in the JCL for the first step of the job. This allocation ensures that the devices are available before the job step starts running. The initiator then starts the program requested in the JCL EXEC statement.

   JES2 and the MVS Basic Control Program (BCP) communicate constantly to control system processing. The communication mechanism, known as the subsystem interface, allows MVS to request services of JES2. For example, a requestor can ask JES2 to find a job, message or command processing, or open (access) a SYSIN or SYSOUT data set. Further, the base control program notifies JES2 of events such as messages, operator commands, the end of a job, or the end of a task.

   By recognizing the current processing phase of all jobs on the job queue, JES2 can manage the flow of jobs through the system.

4. **Output phase**

   JES2 controls all SYSOUT processing. SYSOUT is a data set in a printer device format, that is, it is ready to be printed. Intermediately, a sysout file is stored in the spool data set. There are many advantages in spooling a sysout to JES, such as faster processing (DASD is faster than a printer), optimization of the printer devices, and spool backup and archive. MVS directs to sysout system messages related to job execution.

   After a job finishes, JES2 analyzes the characteristics of the job′s output in terms of its output class and device setup requirements; then JES2 groups data sets with similar characteristics. JES2 queues the output for print processing.

5. **Hardcopy phase**

   JES2 selects output for processing from the output queues by output class, route code, priority, and other criteria. The output queue can have outputs to be processed locally or output to be processed at a remote location (either an RJE workstation or another node). After processing all the output for a particular job, JES2 puts the job on the purge queue.

6. **Purge phase**

   When all processing for a job completes, JES2 releases the spool space assigned to the job, making the space available for allocation to subsequent jobs. JES2 then issues a message to the operator indicating that the job has been purged from the system.

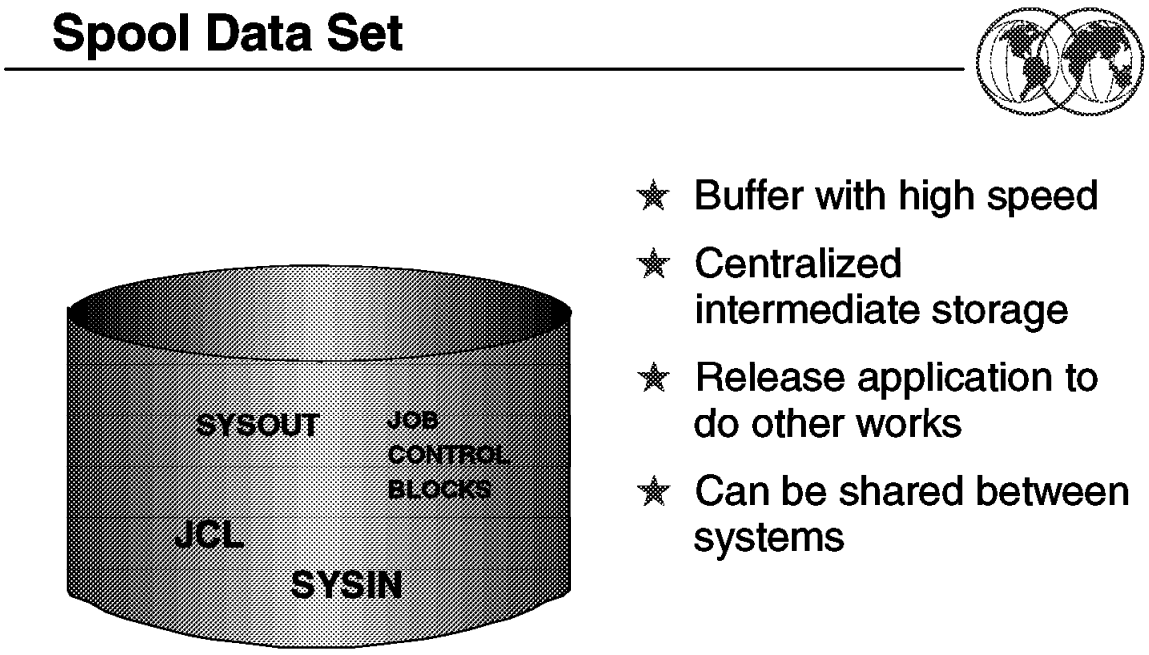


Figure 95. Spool data set

3.3.3 JES2 spool

The Simultaneous Peripheral Operations Online (spool) data set is the direct access device or devices that contains the spool data.

Spooling is a process by which the system manipulates its work.

This includes:

- Using storage on direct access storage devices (DASD) as buffer storage to reduce processing delays when transferring data between peripheral equipment and a program to be run.
- Reading and writing input and output streams on an intermediate device for later processing or output.
- Performing an operation such as printing while the computer is busy with other work.

For these reasons, spooling is critical to maintain performance in the MVS-JES2 environment.

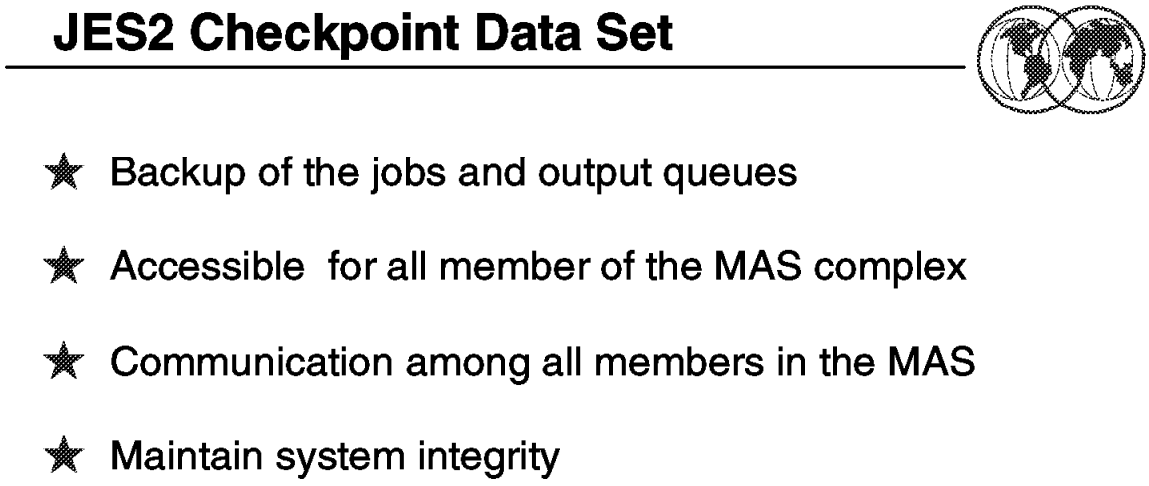


Figure 96. JES2 checkpoint data set

3.3.4 JES2 checkpointing

Errors can occur while processing jobs and data. Some of these errors can result in the halting of all system activity. Other errors can result in the loss of jobs or the invalidation of jobs and data. If such errors occur, it is preferable to stop JES2 in such a way that allows processing to be restarted with minimal loss of jobs and data. The checkpoint data set, checkpointing, and the checkpoint reconfiguration dialog all help to make this possible.

Checkpoint is the data set that JES2 maintains on either DASD or a Coupling Facility that permits JES2 to perform two separate functions:

1. Job and output queue backup to ensure ease of JES2 restart, since it contains information about what work is yet to be processed and how far along that work has progressed.
2. In a multi-access spool (MAS) environment, member-to-member workload communication to ensure efficient independent JES2 operation.

JES2 periodically updates the checkpoint data set by copying the changed information from the in-storage copy to the checkpoint data set copy. Information in the checkpoint data set is critical to JES2 for normal processing as well as in the event that a JES2 or system failure occurs.

Checkpointing is the periodic copying of a member′s in-storage job and output queues to the checkpoint data set. In a MAS environment, checkpointing ensures that information about a member′s in-storage job and output queues is not lost if the member loses access to these queues as the result of either hardware or software errors. Because all members in a JES2 MAS configuration operate in a loosely-coupled manner, each capable of selecting work from the same job and output queues,

checkpointing allows all members to be aware of current queue status. Within the single-member environment, the checkpoint function operates solely as a backup to the “in-storage” job and output (SYSOUT) queues maintained by JES2.

There are three ways of specifying the checkpoint configuration mode. The type of mode you select depends upon your JES2 configuration:

1. DUPLEX-mode processing with backup:

   JES2 can operate its checkpoint configuration in DUPLEX mode if you have defined two checkpoint data sets, a primary and a duplex. They can be either in DASD or a Coupling Facility. The duplex data set is updated once for every write to the primary. If the primary data set suffers an error, the duplex can provide either an exact duplicate of the primary or a very similar, almost current (depending on when it was last updated) copy of the checkpoint data.

   IBM recommends that all members in the JES2 configuration operate, if possible, with DUPLEX=ON in the CKPTDEF. This provides the greatest protection from checkpoint error.

2. DUPLEX-mode processing without backup:

   Processing with a single checkpoint data set is not recommended in most cases. If that data set becomes damaged, lost, or suffers an I/O error, and your member experiences a concurrent failure, you will not have a checkpoint data set available to restart JES2. But you have the benefit that JES2 will not need to read/write to two data sets. However, depending on your specific volume or structure use and configuration, it might not be practical for particular members of your JES2 configuration to maintain a backup checkpoint (for example, in systems that manage basically online transactions).

3. DUAL-mode processing:

   This configuration also uses two data sets, but in a “flip-flop” scheme; that is, individual members do not always read and write to the same CKPTn data set. However, the member performing a read always uses the checkpoint data set last updated. This is not required in a single-member environment. However, it is recommended in a MAS environment, if the installation is suffering from degraded system performance due to high checkpoint data set I/O. The use of the change provides reduced I/O to the checkpoint data set during most, if not all, update accesses by a member of the multi-access spool configuration.

   Note: This mode cannot be used if any checkpoint data set resides on a Coupling Facility structure.

JES2 provides a dynamic means by which the current checkpoint configuration can be changed. It is the checkpoint reconfiguration dialog and can be initiated by the operator system or by JES2. JES2 will enter a checkpoint reconfiguration for any of these reasons:

- To complete initialization processing:
  - Either JES2 could not determine the checkpoint data set specifications, or JES2 requires operator verification of the data set specifications
  - JES2 startup option
  - Checkpoint statement definition
- To correct an I/O error to the checkpoint data set
- To move the checkpoint off a volatile Coupling Facility structure

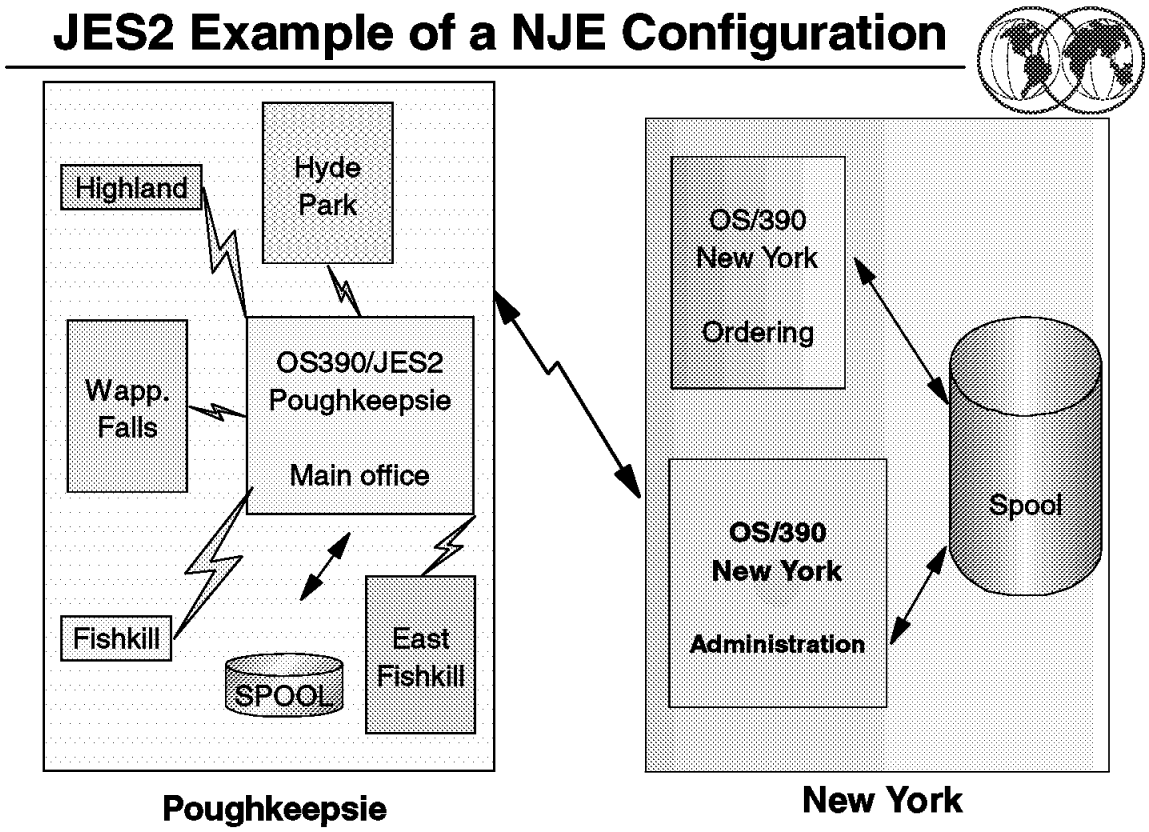


Figure 97. JES2 example of a NJE configuration

3.3.5 JES2 configurations

You can run MVS in your installation in many processing configurations that range from a single MVS image with a single JES2 that is completely isolated from other processing systems, to one that is a part of a worldwide processing network. The choice of configuration complexity is yours and generally is a dynamic one that grows as your business needs grow. Basic configurations are:

- **Remote job entry (RJE)**

  Permits extension of your local processing configuration by defining remote locations that can consist of job input terminals and output devices at a different physical site, connected to the MVS/JES2 image through telecommunication links such as telephone lines and satellites. In this manner, an installation can provide input and output support throughout a large office building, to locations across town, or even in another city. But all the remote sites and their attached devices are defined to a single MVS/JES2 configuration. JES2 can communicate with the remote workstations through two protocols: synchronous data link control (SDLC) and and binary synchronous communication (BSC).

- **Network job entry (NJE)**:

  Allows individual MVS/JES2 to be connected in a network of JES2 and non-JES2 systems that can communicate, pass jobs, and route output to any attached output devices. Each system and its local and remote device make up a *node*. Two or more attached nodes make up an NJE network.

  NJE network nodes communicate using various teleprocessing facilities. Nodes on the same physical processor use ACF/VTAM. Nodes located in the same room or building can utilize

channel-to-channel adapters (CTCA) or telecommunications links. Nodes that are geographically dispersed utilize SNA or BSC telecommunication links.

- **Single system configuration**

  The spool and checkpoint data set are not shared between MVS systems. This configuration is suitable for an installation that has a relatively small workload, or possibly an installation that requires an image to be isolated from the remainder of the data processing complex, possibly to maintain a high degree of security.

- **Multi-access spool (MAS)**

  Also referred to as a multi-system configuration, MAS consists of two or more JES2/MVS images sharing the same spool and checkpoint data sets. There is no direct connection between the JES2 processors. The shared direct access data sets provide the communication link. Each JES2 in a multiple system complex operates independently of the other JES2 processors in the configuration.

  The JES2 share a common job queue and a common output queue, which reside on the checkpoint data sets. These common queues enable each JES2 to share in processing the installation′s workload. Jobs can run on whatever MVS image is available and print or punch output on whatever MVS image has an available device with the proper requirements. Users can also request jobs to run on a particular processor and output to print or punch on a specific device. If one MVS image in the configuration fails, the others can continue processing by selecting work from the shared queues and optionally take over for the failed MVS image. Only work in process on the failed MVS image is interrupted; the other JES2 systems continue processing.

- **Poly-JES**

  MVS allows more than one JES2 subsystem to operate concurrently. One subsystem must be designed as the primary subsystem and the others are defined as secondary subsystems. The secondary JES2 does not have the same capabilities, and some restrictions apply to its use. For example, TSO/E users can only access the primary subsystem. However, secondary JES2 can be useful in testing user modifications while the the primary JES2 is being used for production.

Figure 97 on page 131 shows an example of a JES2 configuration. Poughkeepsie is the MVS site, with remote stations at Fishkill, East Fishkill, and so forth. They comprise the Poughkeepsie node. They are connected with the New York node (MAS configuration).

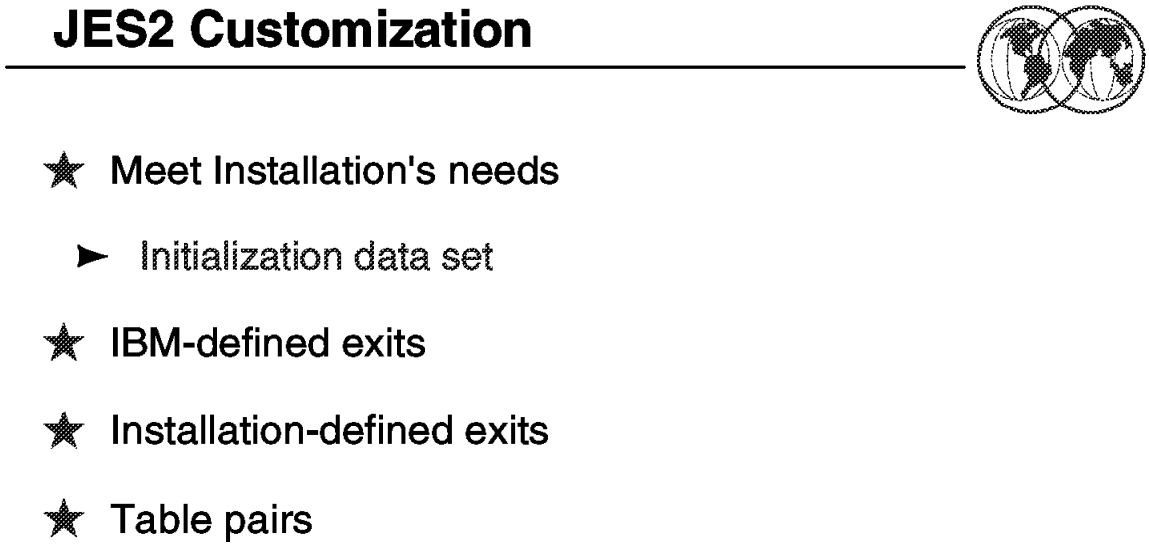


Figure 98. JES2 customization

3.3.6 JES2 customization

JES2 is designed to be tailored to meet an installation′s particular processing needs. You can address your basic customization needs when you create the JES2 initialization data set. If you find this control over JES2 processing to be insufficient, JES2 also provides exits and table pairs to change many JES2 functions or processes.

3.3.6.1 JES2 initialization data set

The JES2 initialization data set provides all the specifications that are required to define:

- Output devices (printers and punches)
- Job classes
- The JES2 spool environment
- The checkpoint data sets
- The trace facility
- And every other JES2 facility and function

A sample JES2 initialization data set is shown in Appendix B, “Job Management” on page 295.

With a set of approximately 70 initialization statements, you can control all JES2 functions. Each initialization statement groups initialization parameters that define a function. The use of most JES2

initialization statements is optional. Many of the parameters provide defaults. You need define only the required parameters and those to fit your installation′s needs.

In the IBM-provided sample data set SYS1.VnRnMn.SHASSAMP data set, the HASI* members are samples you can tailor to meet your installation′s needs.

JES2 initialization statements give the system programmer a single point of control to define an installation′s policies regarding the degree of control users have over their jobs. For example, consider the confusion an installation would experience if users were allowed to define their own job classes and priorities. Very likely, all users would define all their jobs as top priority, resulting in no effective priority classifications at all.

JES2 provides a sort of commands you can use to can dynamically alter JES2 customization whenever your processing needs change. Here are some of 71 available JES2 initialization statements:

```
BUFDEF - Local JES2 Buffer Definition
CKPTDEF - JES2 Checkpoint Definition
CONDEF - JES2 Console Communication Definition
DEBUG - JES2 Debug Option
DESTDEF - Defining How Destinations Are Processed
EXIT(nnn) - Exit and Exit Routine Association
INITDEF - Initiator Definition
INTRDR - Internal Reader
JOBCLASS - Job, Started Task, and Time Sharing User Class
JOBDEF - Job Definition
JOBPRTY(n) - Job Priority
LOGON(nnn) - Identification of JES2 to VTAM
MASDEF - Multi-Access Spool Definition
MEMBER(nn)  - Define Members of a Multi-Access Spool
NETACCT - Define Network/JES2 Account Number Correspo
NODE (xxxxxxxx) - Define a Network Node to JES2
OPTSDEF - Start Options Processing Definitions
OUTCLASS(v) - SYSOUT Class Characteristics
OUTDEF - Job Output Definition
OUTPRTY(n) - Job Output Priority
PRINTDEF - Local Print Environment Definition
RMT(nnnnn) - SNA RJE Workstation
SMFDEF - JES2 SMF Buffer Definition
SPOOLDEF - Spool Volume Definition
TRACEDEF - Trace Facility Definition
```

For more information, refer to *OS/390 JES2 Initialization and Tuning Reference*, SC28-1792.

When you are first getting started, you need not define or even be concerned about some of the more sophisticated processing environments such as a multi-access spool complex, nodes, or remote workstations. You are simply building a base on which your installation can grow. There is no need to be overwhelmed with the potential complexity of your JES2 system.

As your installation grows, you will likely use more and more of JES2′s available functions. The sample data set shipped in SYS1.PARMLIB contains all default values and requires only the addition of installation-defined devices and installation-specific values. It contains all the JES2 initialization statements and the defaults for all parameters for which they are provided.

3.3.6.2 JES2 table pairs

Table pairs provide a facility to change, delete, or add to JES2 processing and/or function. It is a structured mechanism to change JES 2 processing that imposes fewer constraints and less complexity than using exit points.

A number of JES2 functions (such as initialization statement processing, command processing, and message building) use tables. You can customize these JES2 functions, and others, by extending their associated tables. JES2 examines two tables, known as table pairs. The first table is the JES2 table. It provides the default processing specifications. The second table is the user table. The user table is used to extend, change, or delete the default processing specifications. For example, you can add your own JES2 commands and messages, add a new initialization statement or parameter, abbreviate the length of a JES2 command, or delete an unnecessary command to prevent its accidental misuse. For more details about table pairs, refer to *OS/390 JES2 Macros*, SC28-1795.

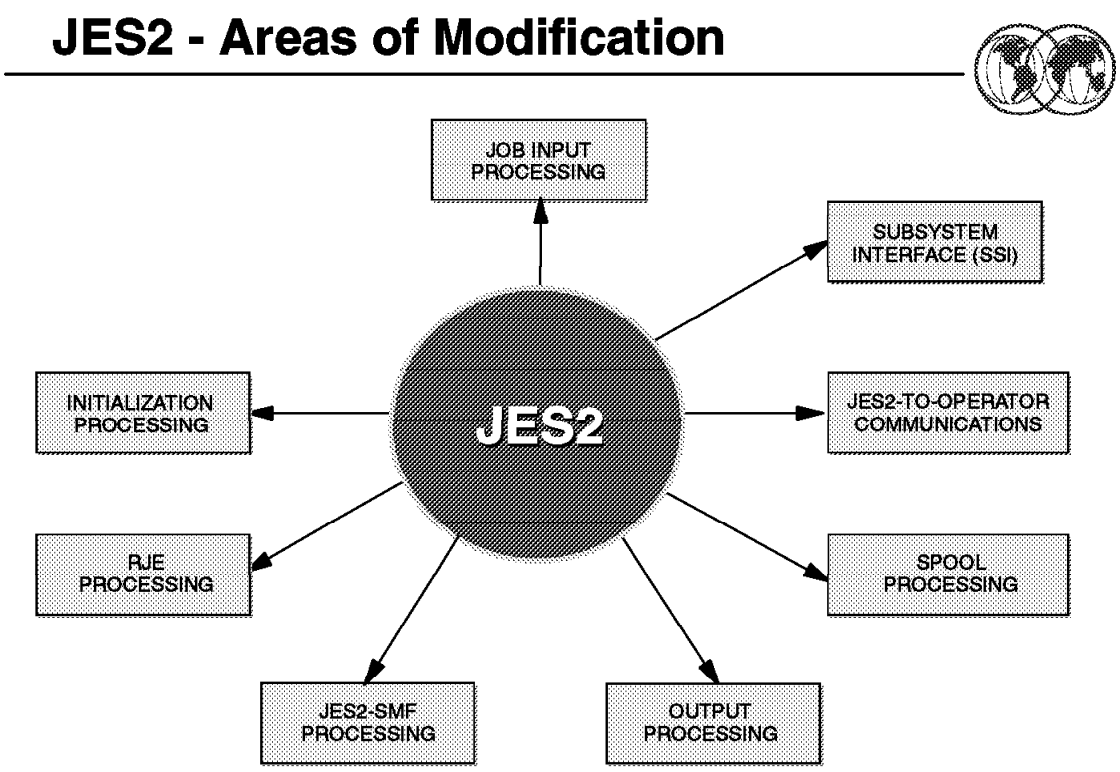


Figure 99. Areas of JES2 modification

3.4 JES2 exits

JES2 may not satisfy all installation-specific needs at a given installation. When you modify JES2 code to accomplish your specific functions, you are susceptible to the migration and maintenance implications that result from installing new versions of JES2. JES2 exits allow you to modify JES2 processing without directly affecting JES2 code. In this way, you keep your modifications independent of JES2 code, making migration to new JES2 versions easier and making maintenance less troublesome.

JES2 provides various strategic locations, called exit points, from where an installation-written exit routine can be invoked. JES2 can have up to 256 exits, each identified by a number from 0 to 255. JES2 code includes a number of *IBM-defined exits*, which have already been strategically placed in JES2 code. For these IBM-defined exits you need only write your own exit routines and incorporate them via the EXIT(nnn) and LOAD(xxxxxx) initialization statements, where *nnn* is the exit point number and *xxxxxx* is the load module name.

If the IBM-defined exits are not sufficient, you can define your own exit points. However, exits established by you are modifications to JES2 code, and you must remember that you run a greater risk of disruption when installing a new version of JES2 code. The new JES2 code into which you have placed your exits may have significantly changed since your insertion of your exit point.

The IBM-defined exits can be classified into two categories:

- Not job-related exits:

These are exits taken during functions not necessarily related to individual jobs (for example, JES2 initialization, JES2 termination, RJE, and JES2 command processing).

- Job-related exits:

  These exits are described in further detail in the following section.

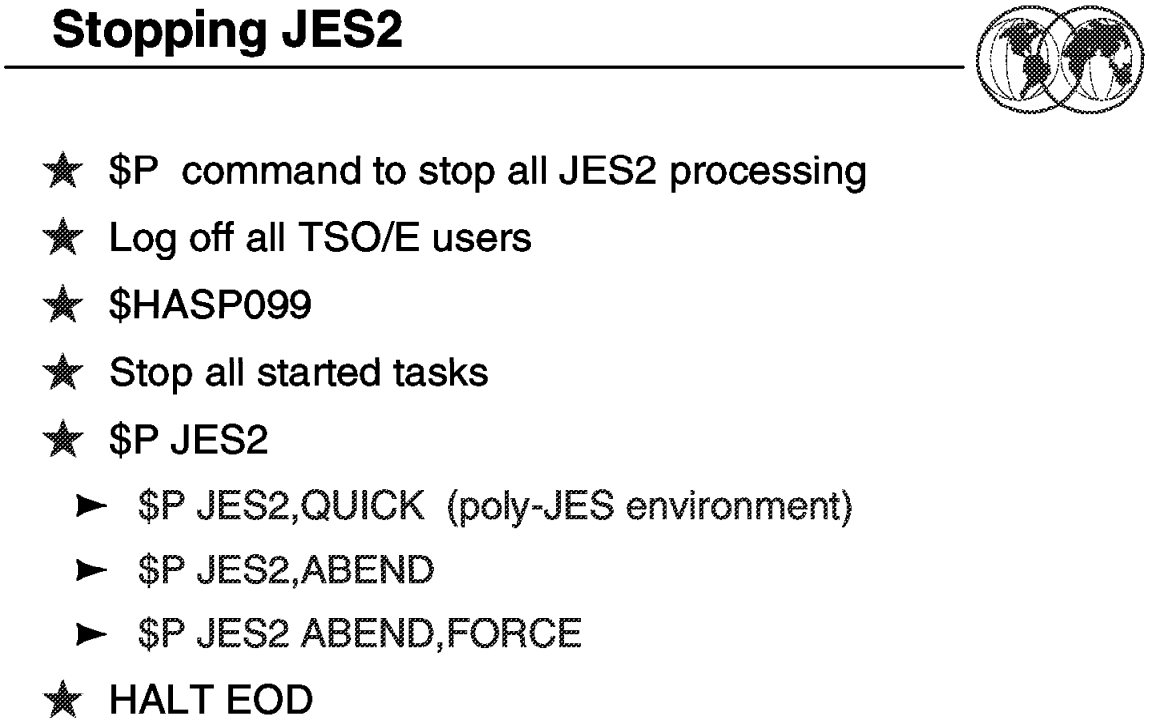


Figure 100. JES2 - Exits on input phase

3.4.1 JES2 job-related exits

Job-related exits can be classified as:

- Specific purpose exits

  They provides a specific function. These exits do not occur at predictable intervals during the life of a job. For that reason, they are not appropriate for general-purpose use. Examples of specific purpose exits are job output overflow (Exit 9) and spool partitioning exits (Exits 11 and 12).

- General purpose exits

  These exits are usually considered when there is a user requirement to control installation standards, job resources, security, output processing, and other job-related functions.

There are two major considerations when selecting an exit to satisfy user requirements:

1. The environment of the exit

   This determines the addressable data areas, the facilities available when the exit is taken, etc.

2. The sequence of the exits

   Which exits precede and which exits follow each exit? What processing has preceded and followed the exit? It is very important to avoid some exit processing in order to avoid being overridden by another exit's processing.

Often the use of more than one exit is required, and sometimes a combination of JES2 and other exits (such as Systems Management Facilities (SMF) exits) must be used. Table 1 on page 139 and Table 2

on page 141 list the job-related exits in order to help you decide which exits to choose to control certain processes or functions during the life of a job.

Figure 100 on page 138 shows the available exit points and their sequence in the job input phase. Many installations use input service exits to control installation standards, tailor accounting information, and provide additional controls. When choosing a exit to act in this phase, it is important to consider all sources of jobs, especially if you want to select jobs from some sources to follow standards. For more details, refer to *JES2 Installations Exits*, SC28-1793.

Table 1 (Page 1 of 3). JES2 job-related exits

Exit	Exit title	Comments and some specific uses
1	Print/punch job separator	Taken when a job's data sets have been selected for printing or punching, prior to the check for the standard separator page. Uses: • Selectively produce separators for particular users or particular job classes. • Place the company's logo on header page. • Suppress production of standard separator.
2	JOB statement scan	The first exit taken for a job and before the statement is processed. Uses: • Enforce security and standards. • Alter/supply JOB statement parameters. • Selectively cancel or purge jobs.
3	JOB statement accounting field scan	Taken after JOB statement has been processed. Uses: • Process nonstandard accounting fields. • Pass information to subsequent exits through the JCT user fields.
4	JCL and JES2 control statement scan	Taken for each JCL and JECL statement submitted but not including JOB and PROCLIB JCL statements. Uses: • Process your own installation-defined JES2 control statement sub-parameters. • Alter/supply JCL parameters.
6	Converter/Interpreter Text scan	A good exit for scanning JCL because of structured text and single record for each statement (no continuation).
7	$JCT Read/Write (JES2 environment)	Receive control whenever control block I/O is performed by the JES2 main task. Uses: • Read or write your own installation-defined job-related control blocks to spool along with the reading and writing of JES2 control.
8	$JCT Read/Write (user/subtask environment)	As 7, but from a user or subtask environment.
15	Output Data Set/Copy	Receive control to handle the creation of separator pages on a data set or copy basis. Uses: • Selectively generate separator pages for each data set to be printed. • Change default print translation tables.
20	End of Job Input	Taken at the end of input processing and before $JCT is written. This is usually a good place to make final alterations to the job before conversion.

Table 1 (Page 2 of 3). JES2 job-related exits		
Exit	**Exit title**	**Comments and some specific uses**
28	SSI Job Termination	Receive control prior to the freeing of job-related control blocks. Uses: • Free resources obtained by Exit 32. • Replace JES2 job termination messages by installation-defined messages
30	SSI Data Set Open/Restart	Taken for SYSIN, SYSOUT, or internal reader Open or Restart processing. Uses: • Examine data set characteristics for validity checking, authorization, and alteration.
31	SSI Data set Allocation	Taken for SYSIN, SYSOUT, or internal reader Allocation processing. Uses: • Fail an allocation. • Affect how JES2 processes data set characteristics.
32	SSI Job Selection	Taken after all job selection processing is complete. Uses: • Suppress job selection-related messages. • Perform job-related processing such as allocation of resources and I/O for installation-defined control blocks.
33	SSI Data Set Close	Taken for SYSIN, SYSOUT, or internal reader Close processing. Uses: • Free resources obtained at OPEN.
34	SSI Data Set Unallocation - Early	Taken for SYSIN, SYSOUT, or internal reader early in allocate processing. Uses: • Free resources obtained by Exit 30.
35	SSI End-of-Task	Taken at end of each task during job execution. Uses: • Free task-related resources.
36	Pre-SAF	Taken just prior to JES2 call to SAF. Uses: • Provide/change additional information to SAF. • Eliminate call to SAF.
37	Post-SAF	Taken just after the return from the JES call to SAF. Uses: • Change the result of SAF verification. • Perform additional security authorization checking above what SAF provides.
40	Modifying SYSOUT	Taken during OUTPUT processing for each SYSOUT data set before JES2 gathers data sets with like attributes into a $JOE. Uses: • Change the destination of a SYSOUT data set. • Change the class of a SYSOUT data set to affect grouping.
44	Post Conversion (JES2 environment)	Taken after job conversion processing and before the $JCT and $JQE are checkpointed. . Uses: • Change fields in the $JQE and $JCT. .

Table 1 (Page 3 of 3). JES2 job-related exits

Exit	Exit title	Comments and some specific uses
46	NJE Transmission	Taken for NJE header, trailer, and data set header during NJE job transmission. Uses: • Remove/add/change installation-defined sections to an NJE data area before transmission.
47	NJE Reception	Taken for NJE header, trailer, and data set header during NJE job reception. Uses: • Remove/change installation-defined sections that were previously added to an NJE data area.
48	Sysout unallocation - Late	More suitable then exit 34 when modifying SYSOUT characteristics or affecting SPIN processing.
49	Job Queue Work Select	Taken whenever JES2 has located a pre-execution job for a device. Uses: • Provide an algorithm to accept or not accept a JES2-selected job • Control WLM initiator job selection.

Table 2. Some SMF job-related exits

Exit	Exit title	Comments and some specific uses
IEFUJV	SMF Job Validation	Receives control: • Before each JCL statement is interpreted and • After all the JCL is converted and again • After all the JCL is interpreted.
IEFUJI	SMF Job Initiation	Receives control before a job on the input queue is selected for initiation. Uses: • Selectively cancel the job.
IEFUSI	SMF Step Initiation	Receives control before each job step is started (before allocation). Uses: • Limit the user region size.
IEFACTRT	SMF Job Termination	Receives control on the termination of each step or job. Uses: • Decide whether the system is to continue the job (for step job) • Decide whether SMF termination records are to be written to SMF data set.
IEFUJP	SMF Purge	Receives control when a job is ready to be purged from the system, after the job has terminated and all its sysouts have been written. Uses: • Selectively decide whether the SMF job purge record (type 26) is to be written to the SMF data set.

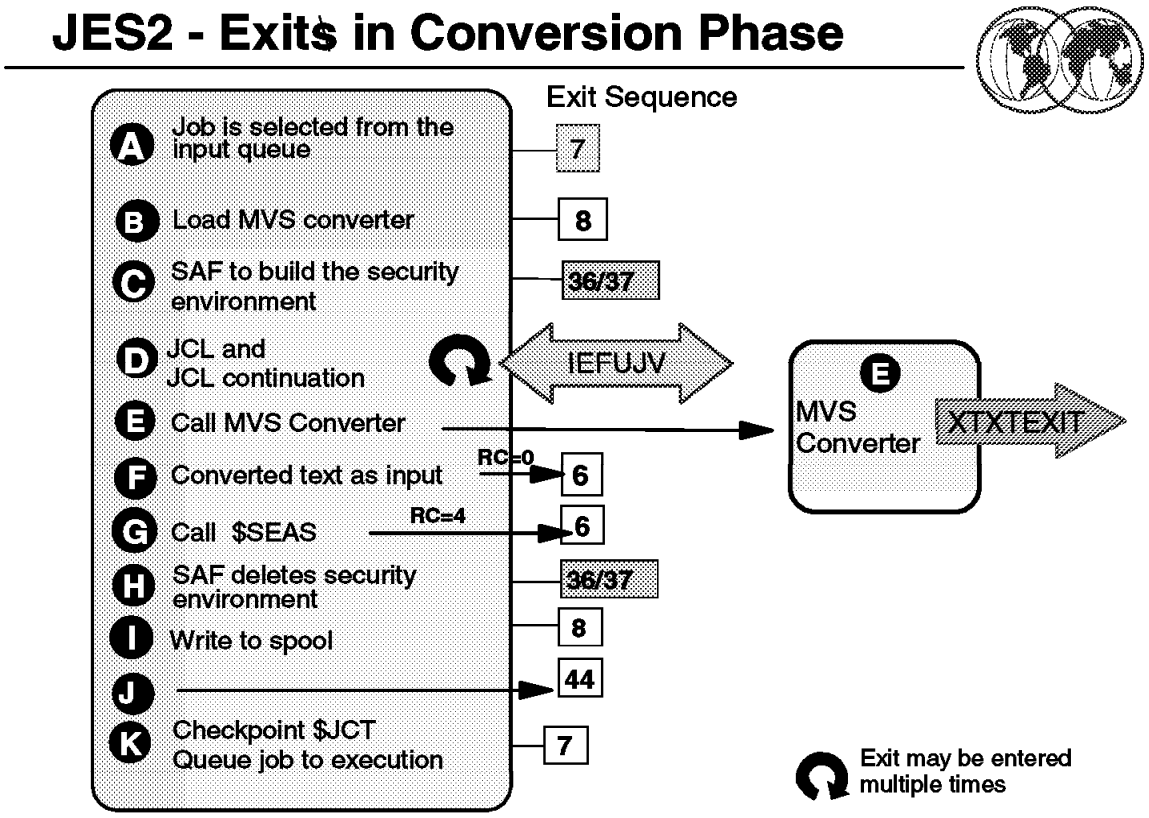


Figure 101. JES2 - Exits in conversion phase

3.4.2 JES2 - Exits in conversion phase

The visual shows the exit points in the conversion phase. The steps **A** and **K** are processed in the JES2 environment. The other steps are processed in the JES2 subtask environment. The reason for the subtask environment is that the conversion process requires the reading of the JCL data set from spool, reading JCL from PROCLIB, writing JCL images to spool, and the writing of C/I text to spool. These I/O operations cannot be accomplished in the maintask environment.

The conversion phase offers the only chance to have exit control over all of a job′s JCL. Although SMF exit IEFUJV is taken for each JCL and continuation statement, JES2 Exit 6 offers some advantages as follows:

- The format of the C/I text is more structured.
- There is no continuation statement, only a single C/I text statement.
- All major syntax errors have been removed.

An SAF security environment can be used with the RACF FACILITY class to control the specification of options within the JCL.

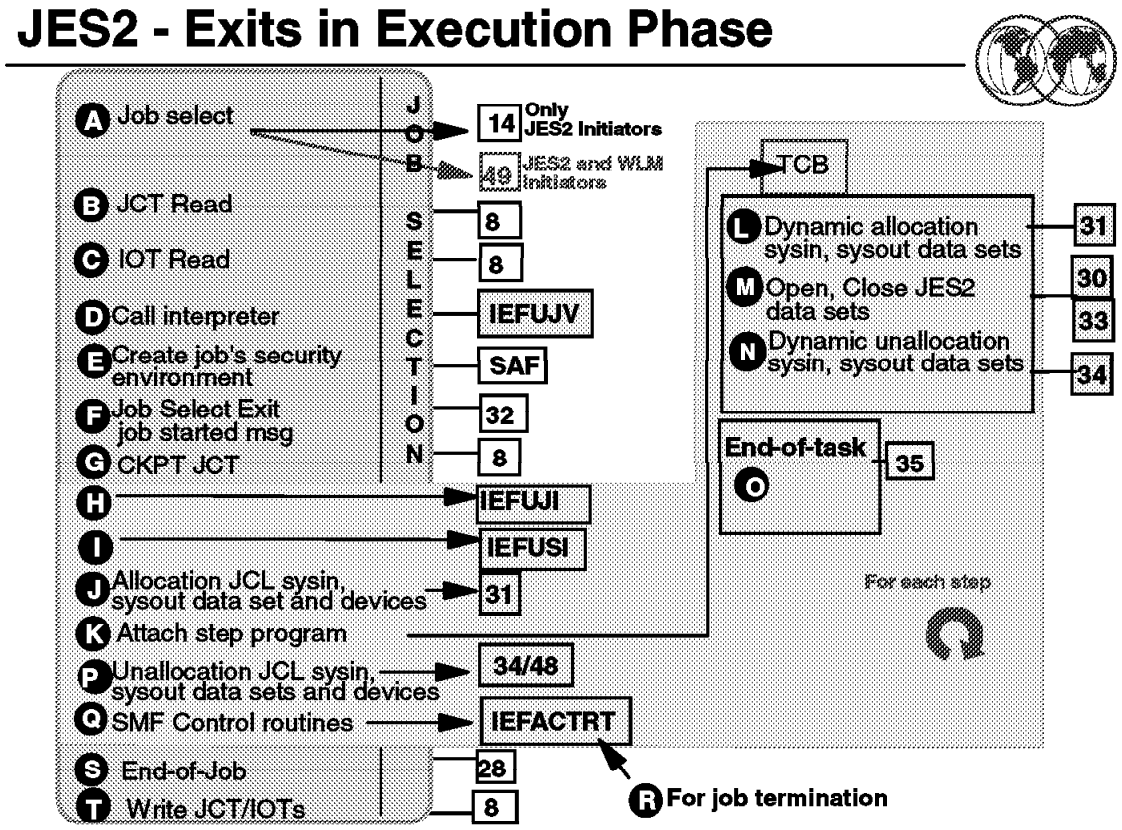


Figure 102. JES2 - Exits on execution Phase

3.4.3 JES2 - Exits on execution Phase

The visual shows exits and their sequences for the execution phase. **Job Selection** is defined as the period starting with the initiator′s Subsystem Interface (SSI) call for job selection by class and ends with the JES2 message, $HASP373 JOB STARTED.

The visual shows services provided by the JES2 job selection (includes end-of-job functions) and the MVS Initiator.

Refer to *OS/390 JES2 Installation Exits*, SC28-1793, for all JES2 job-related exit scenarios.

JES2 Start Procedure

```
//JES2      PROC M=JES2PARM
//IEFPROC   EXEC PGM=HASJES20,TIME=1440,
//           PARM=(WARM,NOREQ)
//HASPLIST DD DDNAME=IEFRDER
//HASPPARM DD DISP=SHR,DSN=SYS1.PARMLIB(&M)
//PROC00    DD DISP=SHR,DSN=SYS1.PROCLIB
//PROC01    DD DISP=SHR,DSN=SYS1.PROCLIB
//STEPLIB   DD DISP=SHR,DSN=SYS1.VnRnMn.SHASLINK
```

Figure 103. JES2 start procedure

3.5 How to start JES2

JES2 can be started after the OS/390 system has been initialized. The OS/390 system automatically starts JES2 if your installation provides this capability. Otherwise, you must issue the START command to invoke a JCL procedure in SYS1.PROCLIB that starts JES2. JES2 initialization is performed after JES2 has been started. Initiators will not accept work (process jobs) until JES2 initialization is complete.

3.5.1.1 JES2 start procedure

To start the JES2 component, an installation must provide a subsystem cataloged JCL procedure (PROC). The PROC name can be a maximum of four characters. The PROC must be defined in the IEFSSNxx member of SYS1.PARMLIB.

The basic JCL procedure is shown in this visual. It contains an EXEC statement and five DD statements. The EXEC statement parameters specify as follows:

PGM= Specify the name of the JES2 load module that resides in an authorized library.

TIME= Specify 'NOLIMIT' or 1440 to prevent system ABEND code 322.

PARM= Specify JES2 start options. IBM suggests you specify *NOREQ* for the start parameters. *WARM* is the default. Table 3 on page 147 lists the JES2 start options available.

IBM recommends that you do not specify the REGION parameter to prevent any virtual storage limitations in the JES2 address space. The DD statement parameters specify as follows:

PROC00 Defines a default procedure library to be used for converting the JCL of batch jobs, time-sharing logons, and system tasks.

PROCxx (01-99) Can be used to define other user-cataloged procedure libraries that are associated with job classes by the JOBCLASS initialization statement or by the /*JOBPARM JES2 control statement.

HASPLIST Defines what is normally a dummy output data set, but you can modify this statement to permanently point to a specific device or data set.

HASPPARM Specifies the data set containing the initialization statements that will be used for JES2 initialization. With this statement, you can control all JES2 functions. The majority of them can be changed dynamically using JES2 commands.

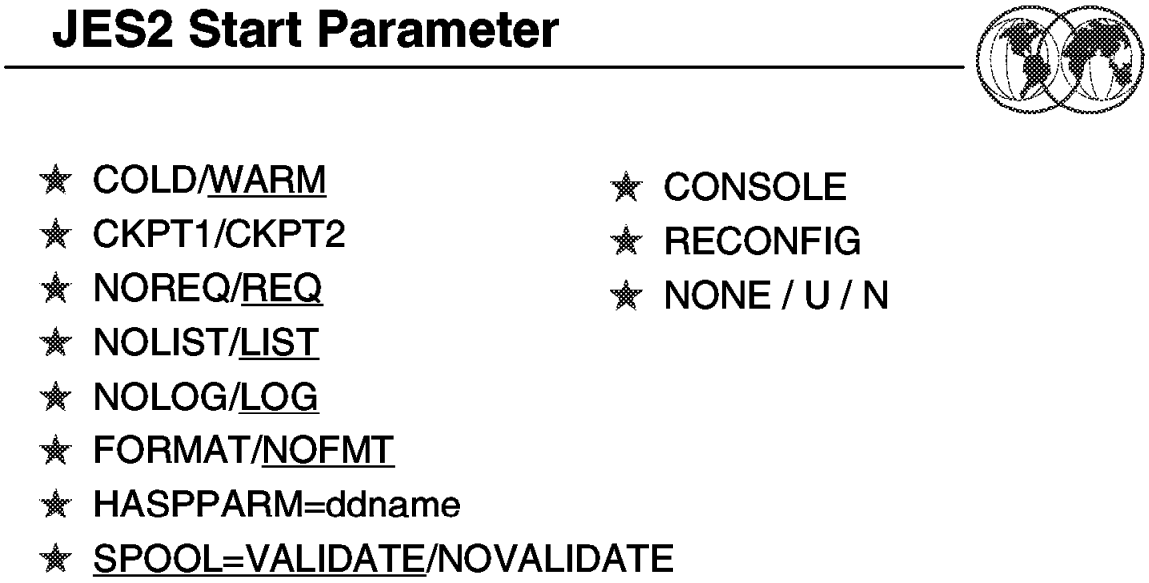


Figure 104. JES2 start options

3.5.2 JES2 start options

JES2 uses start options to determine how it will perform the current initialization. You can specify start options in either of two ways:

- As parameters on the EXEC statement in the JES2 procedure
- As options specified at the console.

If you do not specify the options on the EXEC statement, JES2 requests them from the operator by issuing the $HASP426 (SPECIFY OPTIONS) message.

The operator then enters the options using the standard reply format as described in OS/390 JES2 Commands. The operator can enter options in upper or lower case; they must be separated by commas. If the operator enters conflicting options (for example, WARM,COLD), the last option specified is the one JES2 uses.

If the options are specified on the EXEC statement, JES2 suppresses the $HASP426 (SPECIFY OPTIONS) message and completes initialization without operator intervention unless CONSOLE control statements have been added to the JES2 initialization data set or an initialization statement is in error.

If you let JES2 prompt for the options, JES2 issues message $HASP426:

```
*id $HASP426 SPECIFY OPTIONS - jesname
```

You should respond using the OS/390 REPLY command to specify the JES2 options determined by your installation procedures.

```
REPLY id,options
```

Table 3 explains the JES2 start options. Read the rules following the table before attempting to specify start options from the console.

If you respond to message $HASP426 with the $PJES2 command, JES2 will terminate.

Note: If you specify the NOREQ option, JES2 will automatically start processing following initialization. Otherwise, you must enter the $S command in response to the $HASP400 ENTER REQUESTS message to start JES2 processing.

Table 3 (Page 1 of 2). JES2 Start Parameter

Option	Explanation
FORMAT **NOFMT**	FORMAT specifies that JES2 is to format all existing spool volumes. If you add unformatted spool volumes, JES2 automatically formats them whether FORMAT is specified or not. With FORMAT, JES2 automatically performs a cold start. Default: NOFMT specifies that JES2 is not to format existing spool volumes unless JES2 determines that formatting is required.
COLD **WARM**	COLD specifies that JES2 is to be cold-started. All jobs in the system will be purged and all job data on the spool volumes will be scratched. Default: WARM specifies that JES2 is to continue processing jobs from where they were stopped. If the FORMAT option was also coded, then JES2 will ignore the WARM specification and perform a cold start.
SPOOL=VALIDATE **NOVALIDATE**	VALIDATE specifies that the track group map is validated on a JES2 all-member warm start. Default: SPOOL=NOVALIDATE specifies that the track group map is not validated when JES2 restarts.
NOREQ **REQ**	NOREQ specifies that the $HASP400 (ENTER REQUESTS) message is to be suppressed and JES2 is to automatically start processing when initialization is complete. Default: REQ specifies that the $HASP400 (ENTER REQUESTS) message is to be written at the console. This message allows you to start JES2 processing with the $S command.
NOLIST **LIST**	NOLIST specifies that JES2 is not to print the contents of the initialization data set or any error flags that occur during initialization. If you specify NOLIST, JES2 ignores any LIST control statements in the initialization data set. *OS/390 JES2 Initialization and Tuning Reference* presents an example of an initialization data set listing produced by using the list option. Default: LIST specifies that JES2 is to print all the statements in the initialization data set and any error flags that occur during initialization. (JES2 prints these statements if a printer is defined for that purpose when JES2 is started.) LIST will not print any statements that follow a NOLIST control statement in the initialization data set.
NOLOG **LOG**	NOLOG specifies that JES2 is not to copy initialization statements or initialization errors to the HARDCPY console. If you specify NOLOG, JES2 ignores LOG control statements in the initialization data set. Default: LOG specifies that JES2 is to honor any LOG statements in the initialization data set.
CKPT1 **CKPT2**	Specifies what checkpoint data set JES2 must use for building the JES2 work queues. Default: If you do not specify, JES2 automatically determines which checkpoint data set to use.

Table 3 (Page 2 of 2). JES2 Start Parameter	
Option	**Explanation**
RECONFIG	RECONFIG specifies that JES2 will use the checkpoint data set definitions as specified on the CKPTDEF statement in the initialization data set. JES2 overrides any modifications to the checkpoint data set definitions previously made either by the $T CKPTDEF command or through the use of the checkpoint reconfiguration dialog. Specifying RECONFIG will also cause JES2 to enter the reconfiguration dialog during initialization and issue message $HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED. If you previously reconfigured your checkpoint configuration through the checkpoint reconfiguration dialog, the CKPTDEF statement definition may not contain the most current checkpoint definition. Changes made through the checkpoint reconfiguration dialog are not saved in the input stream data set.
HASPPARM=ddname	HASPPARM=ddname specifies the name of the data definition (DD) statement that defines the data set containing the initialization statements that JES2 is to use for this initialization. Default: HASPPARM=HASPPARM specifies that JES2 is to be initialized using the initialization statements in the data set defined by the HASPPARM DD statement in the JES2 procedure.
CONSOLE	Causes JES2 to simulate receiving a CONSOLE initialization statement after all initialization statements are processed. That is, if CONSOLE is specified, JES2 will divert to the operator console for further parameter information after the input stream data set has been exhausted.
NONE **U** **N**	NONE, U, or N character specifies that JES2 is to use all of the default start options. There is no difference between these three options. These options are equivalent. When NONE, U, or N is specified, JES2 uses the default start options which are: NOFMT WARM REQ LIST LOG

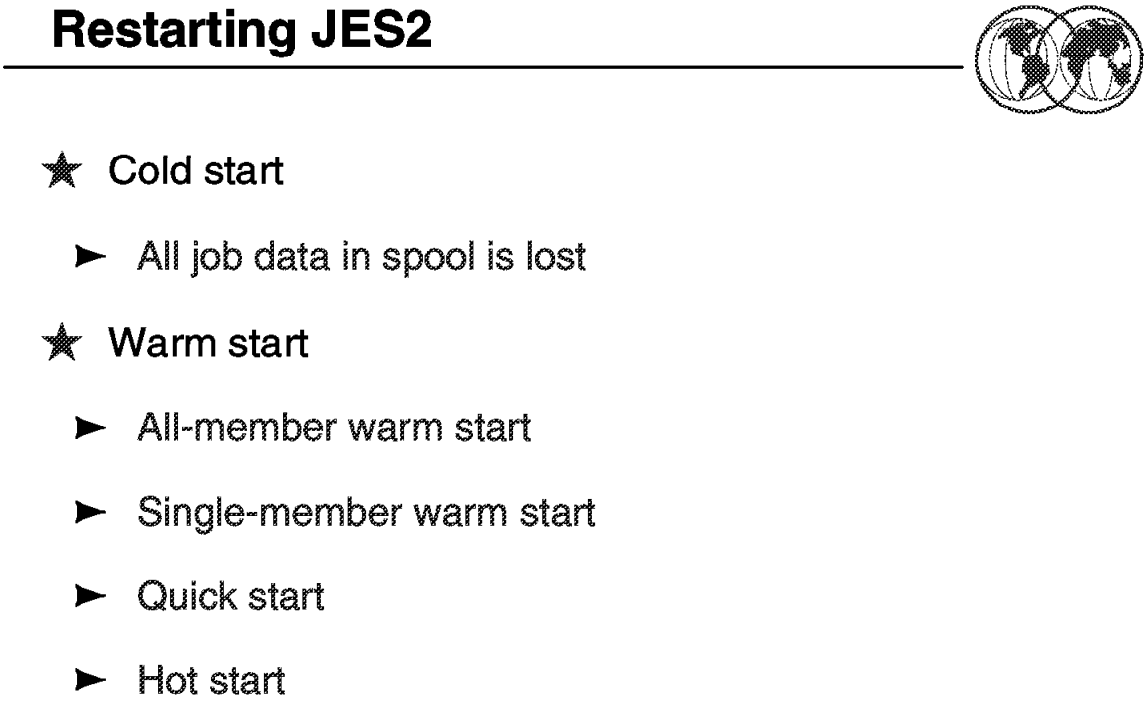


Figure 105. Restarting JES2

3.5.3 Restarting JES2 with cold start option

Very few definitions, or redefinitions, of some JES2 facilities and resources require the JES2 system be totally shut down. JES2 must be restarted with a cold start to allow all component systems to be aware of the changed facilities and resources. The time to restart JES2 in this manner is based on the work in the system and, if not scheduled, causes a disruption in data processing services. All job data previously on the spool volumes is lost with cold start. You can avoid data loss by scheduling a spool offload of the JES2 queues. In a MAS configuration, no other member can be active during a cold start. Use of the FORMAT option causes a cold start.

An IPL must precede a JES2 cold start, unless JES2 was stopped with a $P JES2 operator command.

3.5.3.1 Warm start

During a warm-start initialization, JES2 reads through its job queues and handles each job according to its status:

- Jobs in input readers are lost and must be reentered.
- Jobs in output (print/punch) are requeued and later restarted. The checkpoint for jobs on the 3800 printer points to the end of the page being stacked at the time of the checkpoint. Jobs that were sent to the 3800 printer but that did not reach the stacker are reprinted from the checkpoint. If no checkpoint exists, then it is reprinted from the beginning.
- Jobs in execution are either requeued for execution processing or are queued for output processing.

- All other jobs remain on their current queues.

There are four ways in which a warm start can be performed in a multi-access spool configuration, as follows:

1. **All-member warm start**

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

2. **Single-member warm start**

   This is performed when WARM is specified and others members of the configuration are active. The warm-starting member joins the active configuration and recovers only work in process on that member when it failed or was stopped.

3. **Quick start**

   This is performed when you specified a warm start and JES2 determined that the job queue and job output table do not need to be updated. In this case, the member being started is not the first member being started in the MAS. This occurs:

   - After $P JES2 has been issued to quiesce the member. Because all work is quiesced, there is no need to update the job queue or job output table before restarting.
   - After an all-member warm start has been performed and no work is waiting to be processed; therefore, the job and output queues are empty.
   - When a $E MEM command was entered at a processor within the MAS configuration other than the member being started.

4. **Hot start**

   This is a warm start of an abnormally terminated JES2 member without an intervening IPL. When it happens, all address spaces continue to execute as if JES2 had never terminated. JES 2 validates (and rebuilds, when necessary) the job and output queues and the job queue index. Damaged or corrupted job output elements (JOEs) and job queue elements (JQEs) are placed on the rebuild queue.

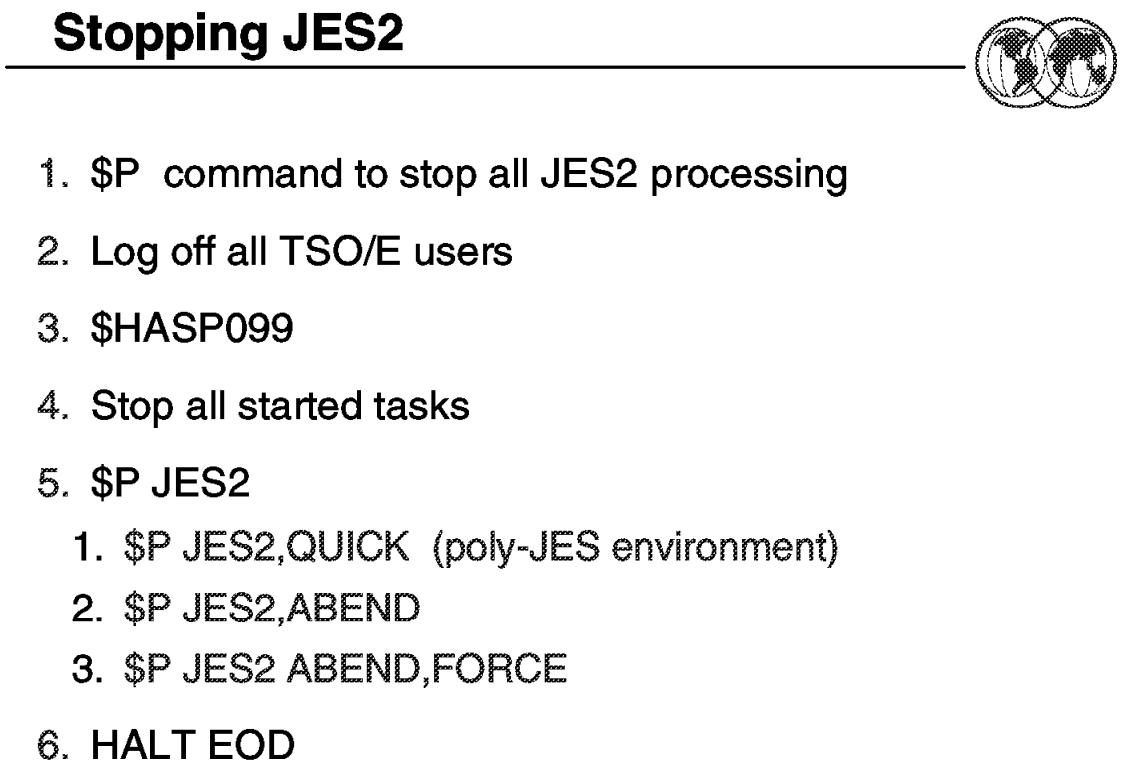


Figure 106. Stopping JES2

3.6 Stopping JES2

There are instances when JES2 must be stopped and restarted either by a warm or cold start. For example, redefining the number of systems in a network job environment requires a warm start. You can stop and restart JES2 in a system at any time by using operator commands. This allows you to:

- Quiesce job processing in preparation for an orderly system shutdown.
- Restart JES2 to perform an initialization with different initialization parameter specifications.

You can dynamically change JES2 initialization statements by using $T operator for most parameters. Before stopping JES2 for the purpose of changing initialization statement parameters, refer to *OS/390 JES2 Initialization and Tuning Reference*, SC28-1791. ,

To stop JES2, do the following:

1. Issue the $P command to stop all JES2 processing. System initiators, printers, punches, job transmitters, and SYSOUT transmitters will not accept any new work and will become inactive after completing their current activity. However, new jobs will be accepted through input devices. When all TSO users log off, and all JES2 started tasks, logical initiators, printers, and punches complete their current activities and become inactive, JES2 notifies you with the following message:

   ```
   $HASP099 ALL AVAILABLE FUNCTIONS COMPLETE
   ```

2. Stop all started tasks.

3. Enter the $P JES2 command to withdraw JES2 from the system. If any jobs are being processed or any devices are active, the $P JES2 command is processed as a $P command and drains JES2 work from the system.

   If it is not possible or reasonable to drain the JES2 member (for example, due to large numbers of lines, jobs, and remotes; or, if you plan to restart JES2 using a hot start) you can specify:

   ```
   $P JES2,ABEND
   ```

   The ABEND parameter forces JES2 termination regardless of any JES2 or system activity. If the checkpoint resides on a Coupling Facility structure and the member is processing a write request, JES2 issues the $HASP552 message and delays the $P command until the checkpoint write has completed.

   If the $P JES2,ABEND command does not successfully terminate JES2, you can also specify the FORCE parameter. The $PJES2,ABEND,FORCE command results in a call to the recovery termination manager (RTM) to terminate the JES2 address space. Because the FORCE parameter can cause unpredictable results, always attempt to enter the $P JES2,ABEND command first.

   To withdraw JES2 from a system involved in cross-system activity, you can issue the $P JES2,QUICK command. Cross-system activity occurs when a user on one JES2 subsystem requests a cross-system function from another JES2 subsystem within the same poly-JES2 environment. This option deletes the control blocks for the request submitted by the user who requested cross-system function. Before using the QUICK keyword on the $P JES2 command, you should send a message to the user asking them to end their cross-system activity.

4. Issue the HALT EOD command. It ensures that important statistics and data records in storage are not permanently lost.

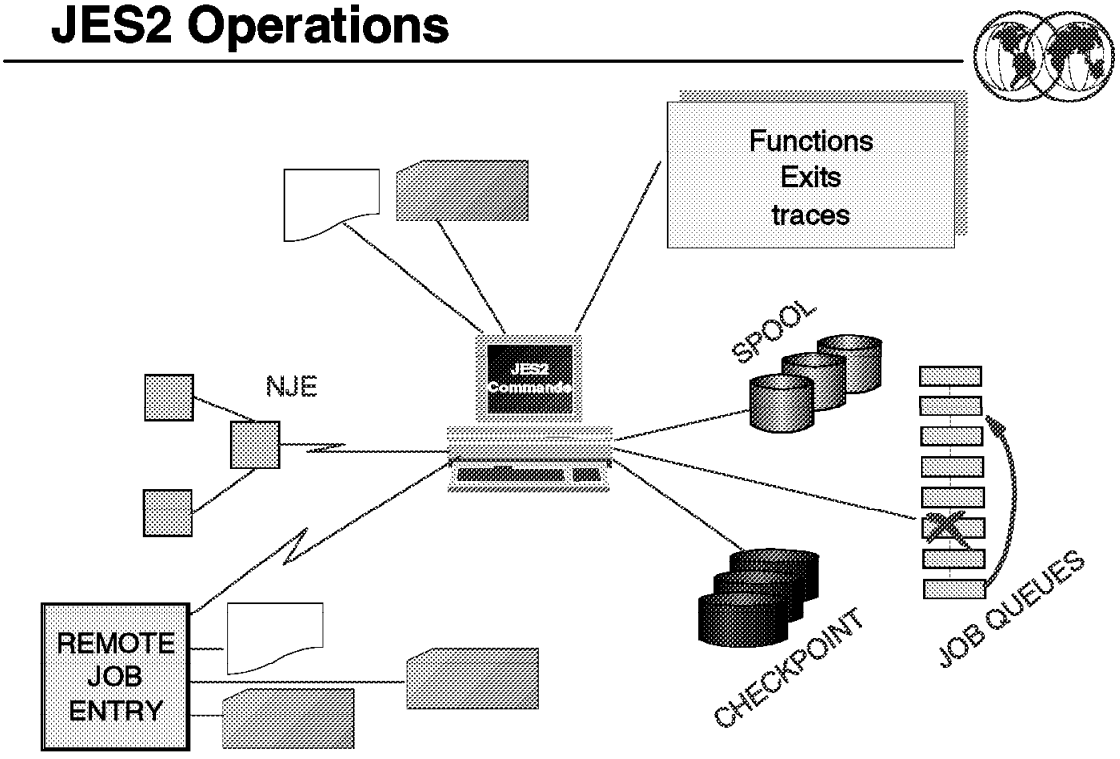


Figure 107. JES2 operations

3.7 JES2 operations

No large data processing system or subsystem can continuously operate independently of system programmer or operator intervention. To help you maintain your overall work environment, JES2 provides a set of commands that permits you to control the most of JES2 functions and devices under its control.

As your JES2 complex becomes more sophisticated, you might connect your system to others to form a network of systems. You can use operator commands to control the lines that connect the separate systems, as well as to define the separate systems to yours. This is typically a very dynamic environment, as different systems are added or deleted from the network due to maintenance, hardware reconfiguration requirements, workload balancing, or the need to access a database at a different location. JES2 commands permit you to alter most of your original network definition, as required. Almost all JES2 initialization statements can be changed dynamically by JES2 operator commands.

Operator commands can be used to:

- Display status information and device definition.
- Start, stop, and halt devices under JES2′s control.
- Assign units to local printers, punches, card readers, and lines, or reassign units to these devices.

- Modify processing, such as: output definition, the dynamic alteration of the checkpoint definition, enabling installation-defined exits, offload devices, printer and punch characteristics, and job characteristics.
- Add function and functional subsystems.
- Delete function, network systems, exits, and diagnostic traces.

Your installation can require that only certain operators can issue certain JES2 commands and/or restrict commands coming in from RJE workstations, NJE nodes, or any other device. You can use RACF and customization techniques to limit the degree of control an individual or group can have over JES2 resources.

All JES2 commands are entered using standard OS/390 command interfaces such as an OS/390 console or within the JES2 initialization data set. As the default, JES2 prefixes its commands and messages with a dollar sign ($). For commands, the prefix character defines the scope of the command as being JES2 only; for messages, the prefix character is informational in that it designates that the message was issued by the JES2 component. You can change this symbol through the CONDEF initialization statement.

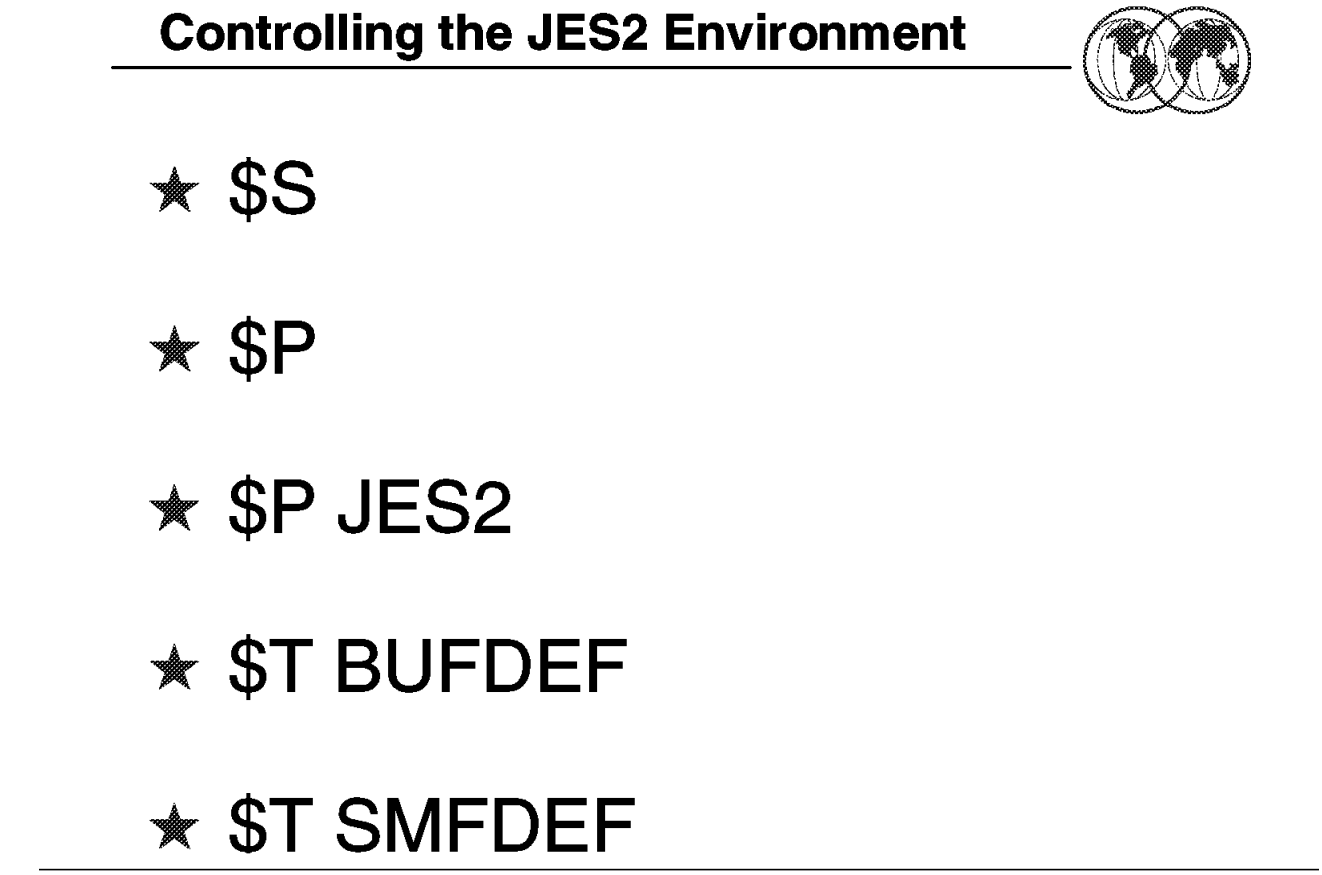


Figure 108. Controlling the JES2 environment

3.7.1 Controlling the JES2 environment

The following commands are useful to control the JES2 environment:

- Start JES2 processing, $S
- Stop JES2 processing, $P
- Withdraw JES2 from the system, $P JES2
- Monitor local buffers, $T BUFDEF
- Monitor SMF buffers, $T SMFDEF

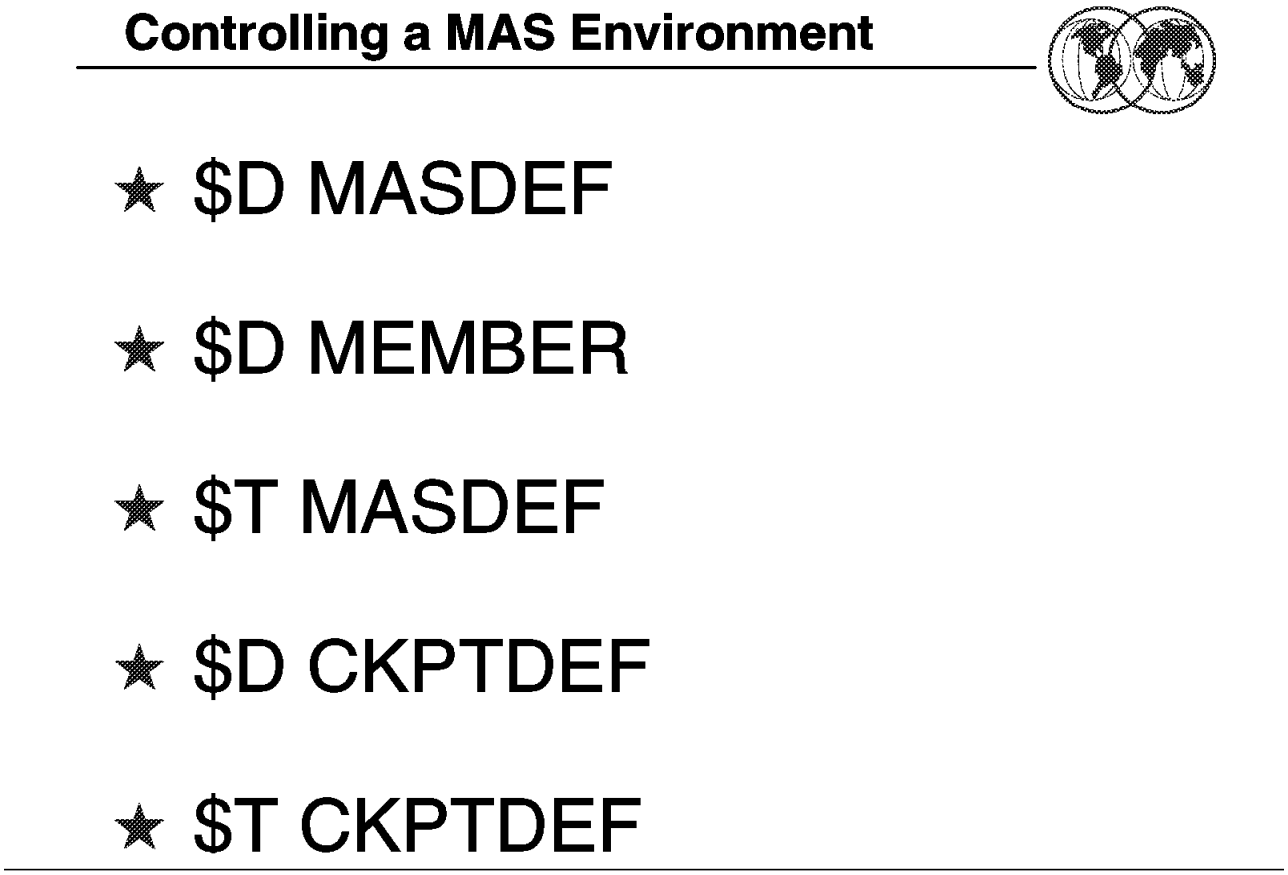


Figure 109. Controlling a JES2 MAS environment

3.7.2 Controlling a JES2 MAS environment

The following commands are useful to control your multi-access spool (MAS) environment:

- Display characteristics of the MAS, $D MASDEF
- Display the status of MAS members, $D MEMBER
- Control the MAS environment, $T MASDEF
- Display the checkpoint definition, $D CKPTDEF
- Control the checkpoint definition, $T CKPTDEF

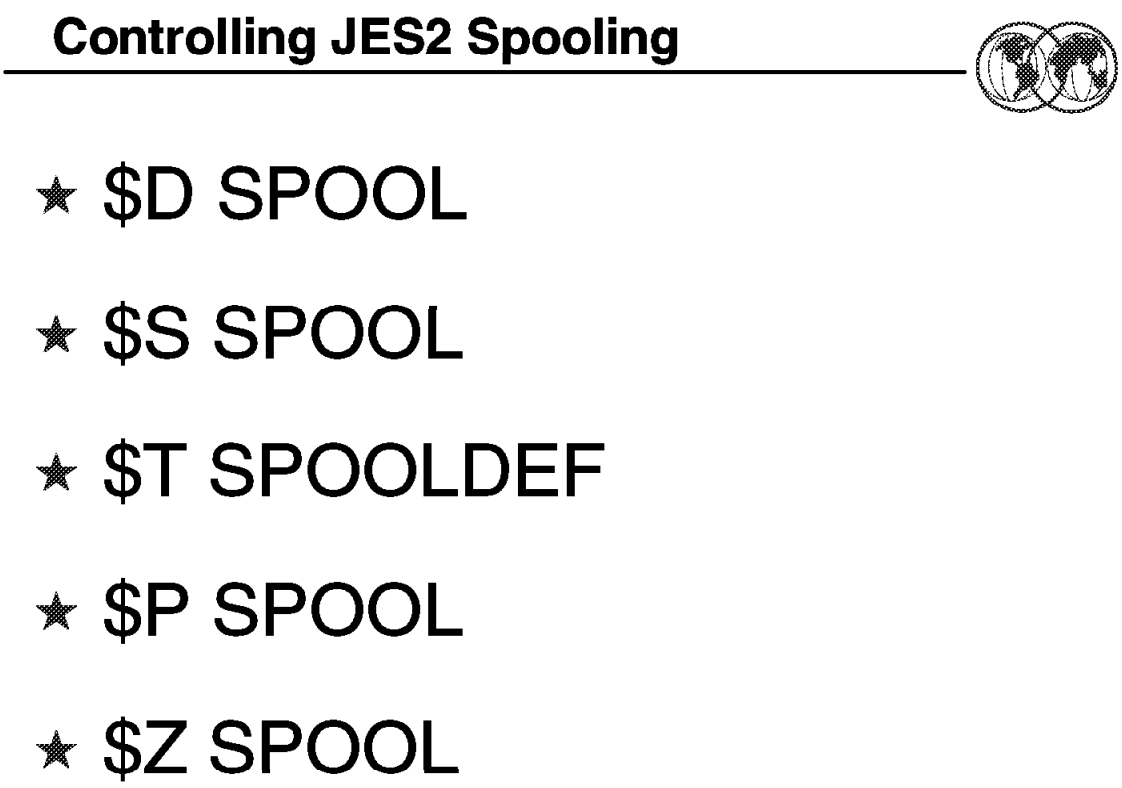


Figure 110. Controlling JES2 spooling

3.7.3 Controlling JES2 spooling

The following commands are useful to control the JES2 spool:

- Display spool volume usage, $D SPOOL
- Start a spool volume, $S SPOOL
- Control the JES2 spooling environment, $T SPOOLDEF
- Drain a spool volume, $P SPOOL
- Halt a spool volume, $Z SPOOL

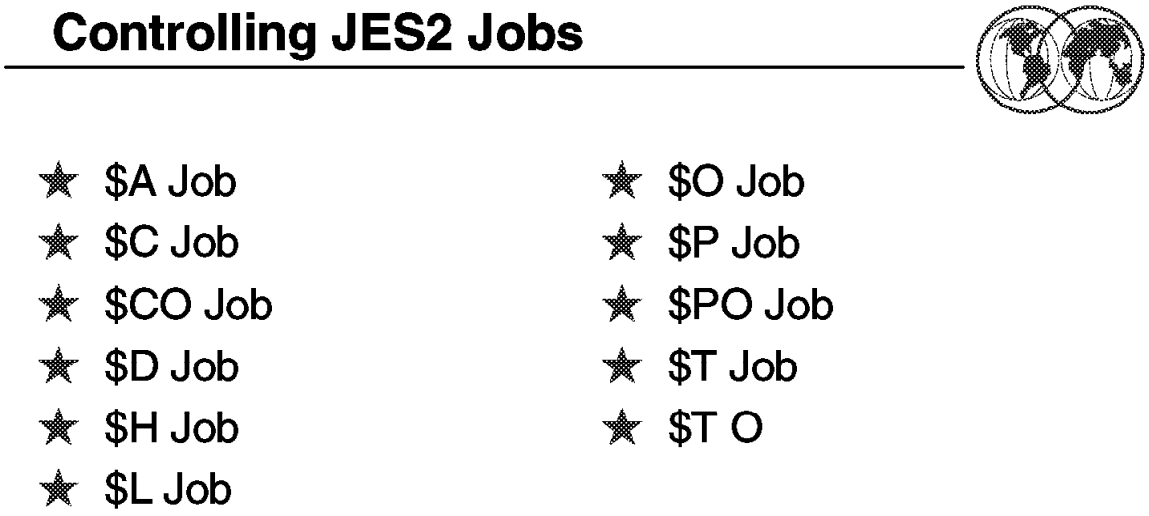


Figure 111. Controlling JES2 jobs

3.7.4 Controlling JES2 jobs

The following commands are useful to control JES2 jobs:

- Release specified jobs, $A Job
- Cancel a job, $C Job
- Cancel output groups, $CO Job
- Display information about a specified job, $D Job
- Hold a specified job, $H Job
- List job output information, $L Job
- Release or cancel held output groups, $O Job
- Purge a job, $P Job
- Purge an output job, $PO Job
- Change a job′s class, scheduling priority, or member affinity, $T Job
- Set output characteristics, $T O

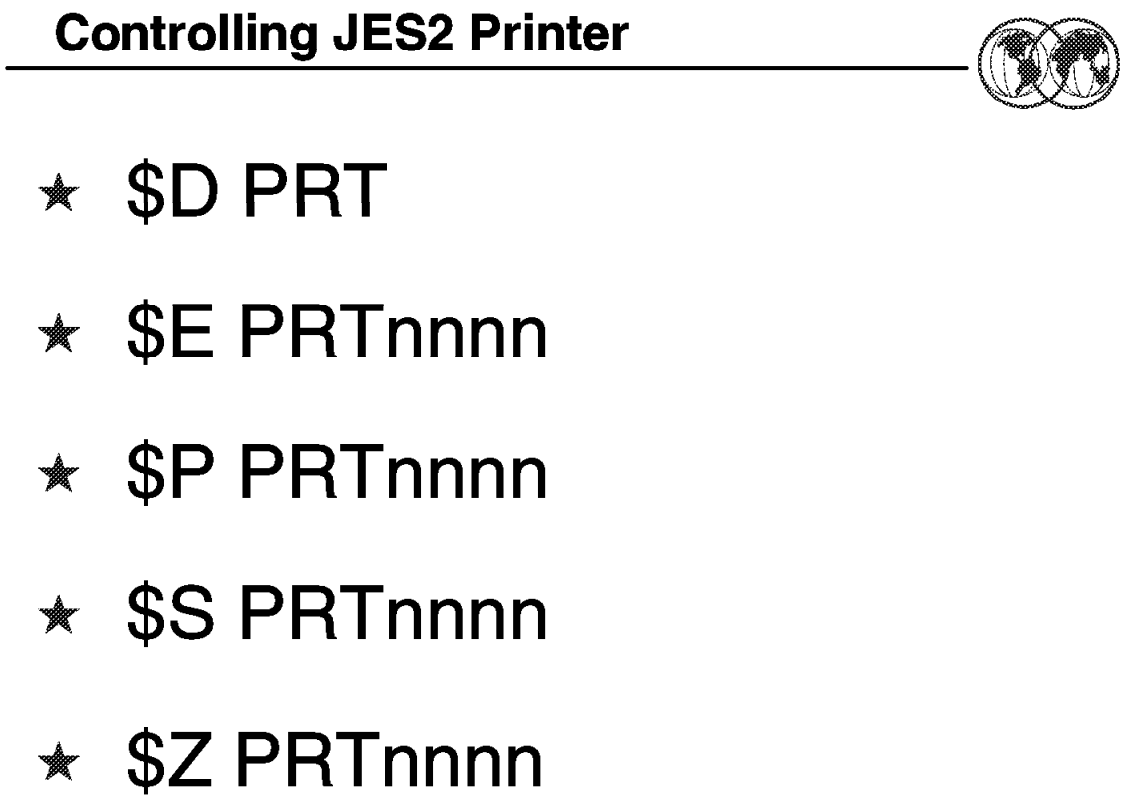


Figure 112. Controlling JES2 printers

3.7.5 Controlling JES2 printers

The following commands are useful to control JES2 printers:

- Display printers, $D PRT
- Restart printer activity, $E PRTnnnn
- Stop a printer, $P PRTnnnn
- Start a printer, $S PRTnnnn
- Halt printer activity, $Z PRTnnnn

3.7.5.1 JES2 diagnosis communication mechanisms

The following diagnostic tools are available for detecting and diagnosing problems occurring in the JES2 environment.

- Messages

  JES2 provides a set of messages to alert initially the JES2 operator and system programmer of processing errors.

- Traces

  Optionally, your installation can use the JES2 tracing facility, a function that records events associated with specific functions, such as each time JES2 is initialized or terminated or each time an exit routine is taken.

- IPCS

  JES2 exploits the interactive problem control system (IPCS) facility to allow you to view the formatted contents of JES2 control blocks and dumps of system data necessary when diagnosing and recovering from processing errors.

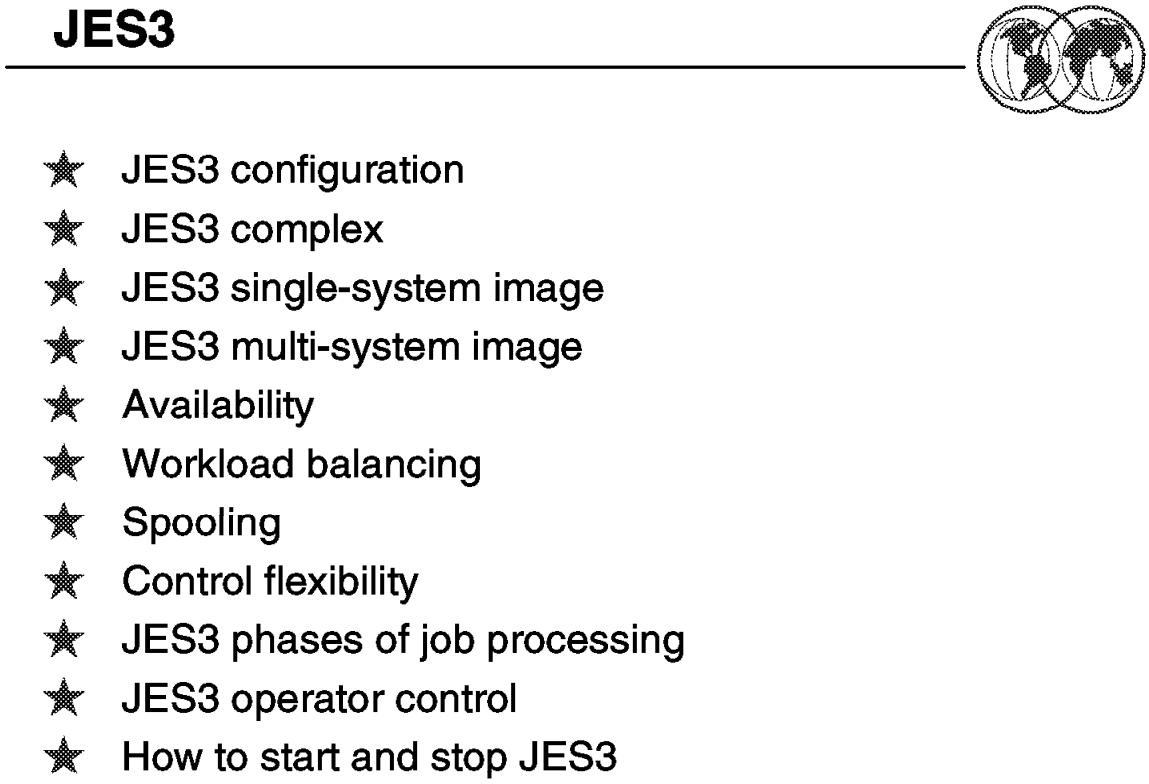


Figure 113. JES3

3.8 JES3

With the OS/390 MVS JES3 system, resource management and workflow management are shared between MVS and its Job Entry Subsystem 3 (JES3) component. Generally speaking, JES3 does resource management and workflow management *before* and *after* job execution, while MVS does resource and workflow management during job execution.

JES3 considers job priorities, device and processor alternatives, and installation-specified preferences in preparing jobs for processing job output. The features of the JES3 design include:

- Single-system image
- Workload balancing
- Availability
- Control flexibility
- Physical planning flexibility

This visual shows the JES3 topics discussee in this section.

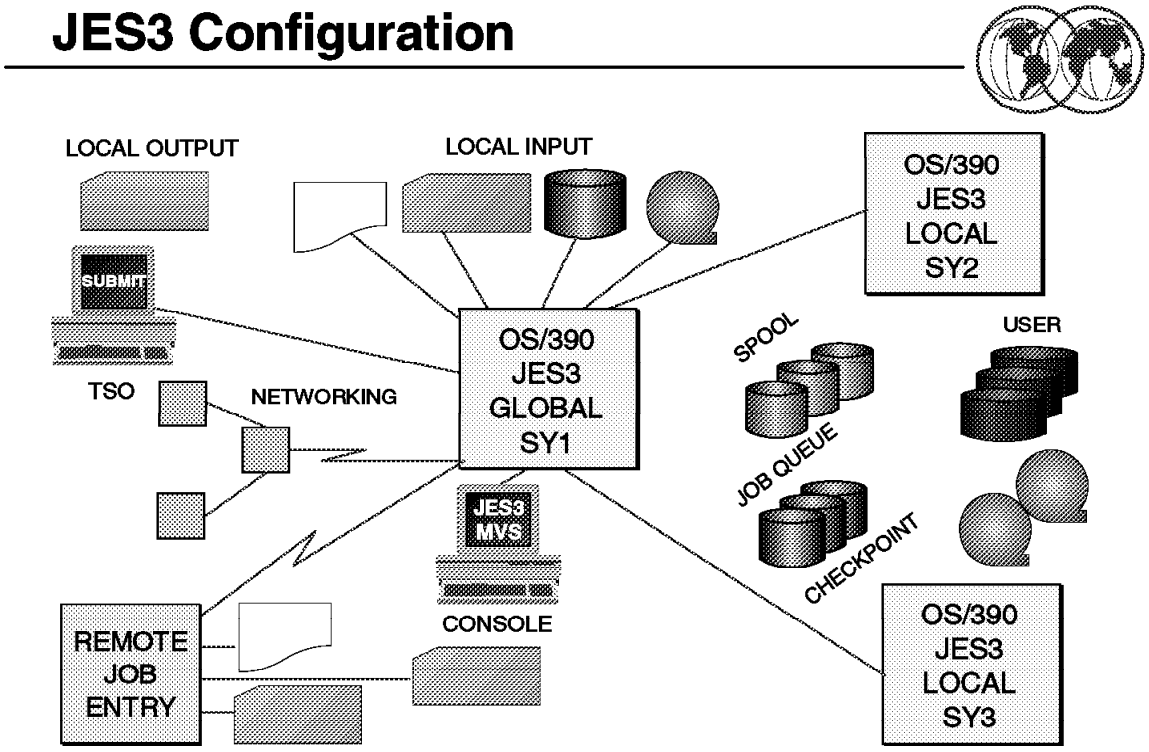


Figure 114. JES3 configuration

3.8.1 JES3 configuration

JES3 runs on either one processor or on up to thirty-two processors in a sysplex. A *sysplex* is a set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads.

In a sysplex, your installation must designate one processor as the focal point for the entry and distribution of jobs and for the control of resources needed by the jobs. That processor, called the *global processor*, distributes work to the processors, called local processors.

It is from the global processor that JES3 manages jobs and resources for the entire complex, matching jobs with available resources. JES3 manages processors, I/O devices, volumes, and data. To avoid delays that result when these resources are not available, JES3 ensures that they are available before selecting the job for processing.

JES3 keeps track of I/O resources, and manages workflow in conjunction with the workload management component of MVS by scheduling jobs for processing on the processors where the jobs can run most efficiently. At the same time, JES3 maintains data integrity. JES3 will not schedule two jobs to run simultaneously anywhere in the complex if they are going to update the same data.

JES3 may be operated from any console that is attached to any system in the sysplex. From any console, an operator can direct a command to any system and receive the response to that command. In addition, any console can be set up to receive messages from all systems, or a subset of the

systems in the sysplex. Thus, there is no need to station operators at consoles attached to each processor in the sysplex.

If you want to share input/output (I/O) devices among processors, JES3 manages the sharing. Operators do not have to manually switch devices to keep up with changing processor needs for the devices.

The JES3 architecture of a global processor, centralized resource and workflow management, and centralized operator control is meant to convey a single-system image, rather than one of separate and independently-operated computers.

JES3 runs in the following environments:

- Single-processor environment
- Multiprocessor environment
- Remote job processing environment
- JES3 networking environment
- APPC environment

A JES3 complex can involve any combination of these environments.

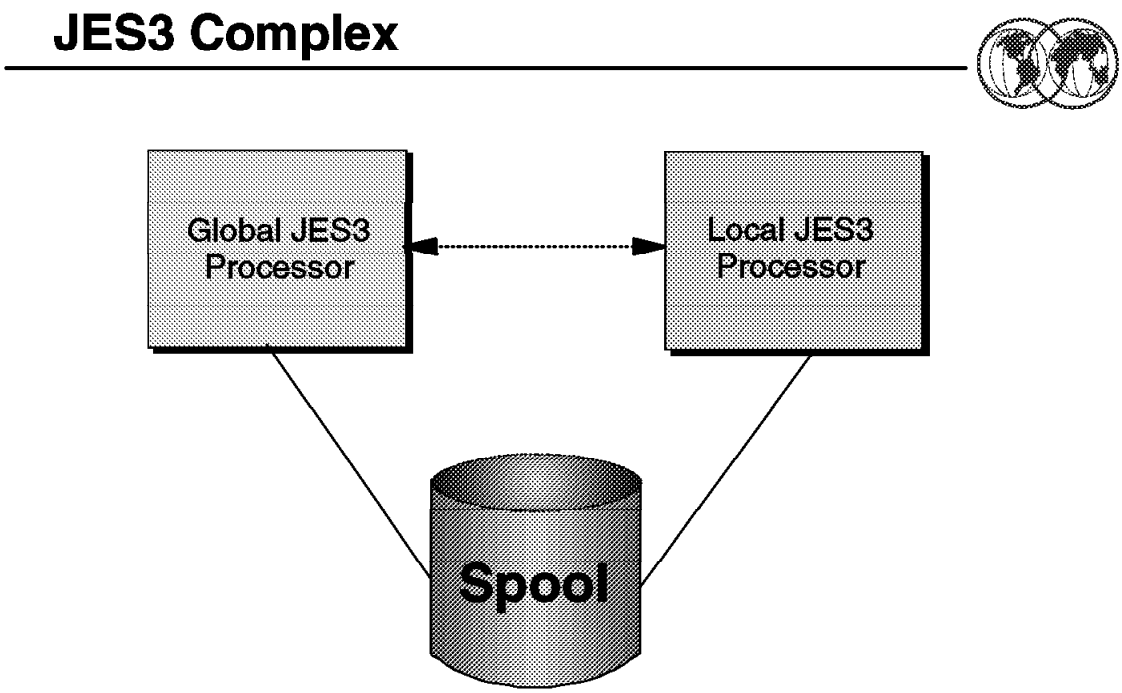


Figure 115. JES3 complex

3.8.2 JES3 complex

OS/390 uses the Job Entry Subsystem 3 (JES3) portion of the OS/390 System Product - JES3 to control the input, processing, and output of jobs. JES3 services the job processing requirements of one to 32 physically connected OS/390 processors called *mains*. Viewed as a whole, the one- to 32-main environment serviced by JES3 is called a *complex*.

JES3 has its own private address space in each of the mains in the complex. One main, the JES3 global main, is in control of the entire complex. There must be a *global main*; if there is only one main in the complex, that main is the global. In a complex with more than one main, the other mains in which JES3 resides are called *local mains*. There can be as many as 31 local mains.

JES3 is designed so that if the global fails, any properly-configured local within the complex can assume the function of the global through a process called dynamic system interchange (DSI). The visual illustrates the basic structure of a JES3 complex.

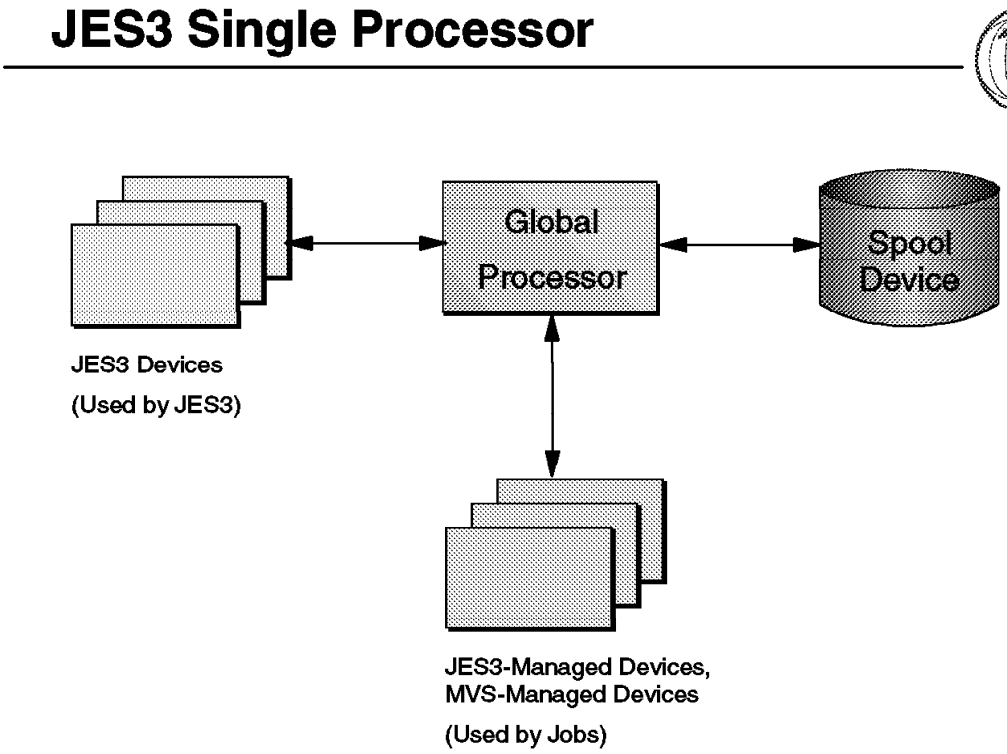


Figure 116. JES3 in a single-processor environment

3.8.3 JES3 single processor

The visual shows JES3 in a single-processor environment, also known as a single-system sysplex. Besides the processor, two categories of I/O devices are shown:

- JES3 devices (those used by JES3)
- JES3- and MVS-managed devices (those used by jobs).

The spool device is a direct-access storage device (DASD) that is treated in a special way by JES3, so it is shown and explained separately.

In installations with one processor, the global processor, JES3 coexists in that processor with MVS and with running jobs. In many respects, JES3 is treated like a job by MVS. (MVS handles JES3 in much the same way it handles the jobs it receives from JES3 for processing). JES3 becomes a processable ″job″ during initialization, while jobs submitted to the system become processable by the actions of JES3.

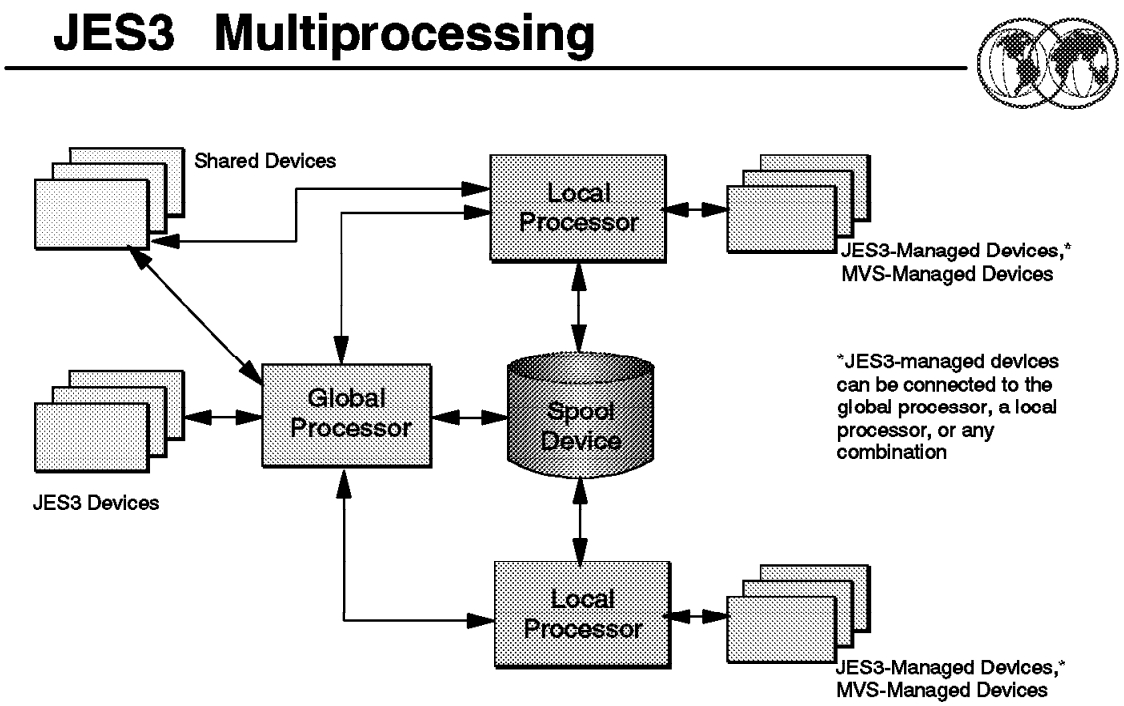


Figure 117. JES3 multiprocessing

3.8.4 Multi-system image

Multiprocessing is a way of doing work with two or more connected processors. In a multiprocessing environment, also known as a multisystem sysplex, JES3 allows up to 32 processors, also known as mains, to be configured into the complex. JES3 uses one processor (called the global) to do work and also to distribute work to up to 31 other processors (called locals). Some of the advantages of running JES3 in this environment are:

- Elimination of much of the overhead of scheduling work for and operating separate processors
- Sharing devices by processors, which means that the devices can be used more efficiently
- Movement of work to other processors, should one processor become overworked, need maintenance, or need to be removed from the complex for any reason.

The global processor and the local processors communicate using the OS/390 cross-system Coupling Facility (XCF). JES3 devices (the devices from which JES3 reads jobs, and on which JES3 stores jobs awaiting processing, stores job output, and writes job output) are connected to the global processor. Some JES3 devices that write job output can also be connected to the local processors.

The visual shows a JES3 multiprocessing system that has three processors (that communicate via XCF signalling). The global processor runs most of JES3, for it is from the global processor that JES3 allocates resources to jobs and sends jobs to local processors for processing. In addition, JES3 can also pass jobs to OS/390 in the global processor for processing.

Each local processor has a complete OS/390 system and JES3 routines for spooling and for communication with the global processor. When OS/390 in a local processor needs a job, JES3 in that local processor sends the job request to JES3 in the global processor via XCF signalling. JES3 in the global processor returns identifying information about a job, and JES3 in the local processor uses that information to retrieve the job from the spool device.

If a problem arises on the global processor while JES3 is running, you can transfer global functions to a local processor. This transfer is called dynamic system interchange (DSI). The processor you choose must be capable of assuming the same workload as the previous global.

3.8.5 Availability

If a problem develops with the global processor, you can have one of the other processors assume the global functions. (Operators have JES3 commands to cause the switch.) Jobs running on other processors, including the one to become the new global processor, will usually be unaffected (except for a waiting period until global activities are resumed).

If part of JES3 fails, JES3 collects failure symptoms, records error data, and attempts recovery. All major JES3 components are protected by at least one special JES3 recovery routine. If recovery is unsuccessful, the failing component gets insulated from the rest of JES3, resources are freed, and the failing component will not be used again. If the component is not critical to overall JES3 processing, complete JES3 failure may be avoided.

3.8.6 Workload balancing

JES3 balances workload among processors by considering the resource requirements of jobs. The method JES3 uses is the same whether one or several processors make up the configuration. Thus, addition of another processor does not mean a new operational and scheduling environment.

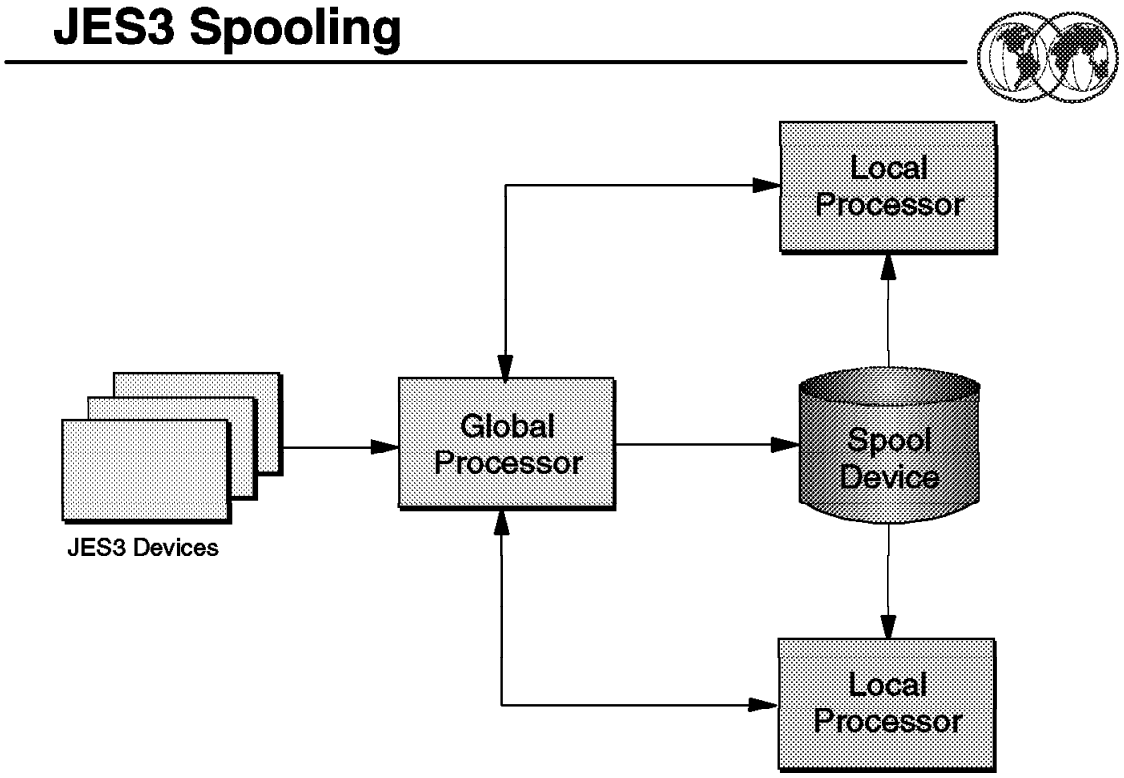


Figure 118. Spooling to collect job output

3.8.7 Spool devices

Most multiprocessing systems use shared data storage. In JES3 the spool device becomes the shared data storage. Thus, besides being a buffer (as for single-processor systems), in multiprocessing systems the spool device becomes a collection point for data to be distributed to individual processors. Also, the spool device becomes a collection point for data coming from local processors routed to JES3 output devices connected to the global processor.

3.8.8 Control flexibility

Operating systems must be easy to control. Internal complexity must be offset by features that make the systems easy to operate, to monitor, and to change. JES3 has designed-in features for operators, application programmers, and system programmers:

- Operators have special JES3 commands. Some commands activate programs to handle I/O devices, while others obtain or change the status of jobs being processed. With multiple processing-unit systems, JES3 operators have less to do than for an equal number of individual systems, because they can control the entire complex from a central point, and because JES3 decides where and when jobs will be processed.

  JES3 applies installation policies, as defined in the JES3 initialization stream, to perform job scheduling, thus freeing the operator from this task.

  Even though JES3 handles job flow control, there are operational controls in JES3 for the operator to use at his or her discretion to override JES3 decisions, and to take control in unusual situations.

- Application programmers have special JES3 control statements (similar to JCL statements). There are control statements to make some jobs run only after successful (or unsuccessful) processing of other jobs, and for specifying the time of day, week, month, or even the year when jobs should run.
- System programmers have special JES3 initialization statements to define the way JES3 is to manage devices and jobs. Operators or application programmers can override many of these initialization options on a job-by-job basis. JES3 gives system programmers a unique way of setting installation policy for device and job management. They define separate groups of processing rules. Application programmers select and apply the groups of rules in various ways to individual jobs.

Where JES3 does not provide the exact function that your complex requires, such as special printing requirements, the system programmers can write their own routines and make them part of the JES3 program. This is done through installation exits provided with JES3 and through dynamic support programs that can be added to JES3. For diagnostic purposes, JES3 provides traces and dumps. Traces are copies of data made during normal processing to help monitor activity in the system, and dumps are copies of data made at times of failure.

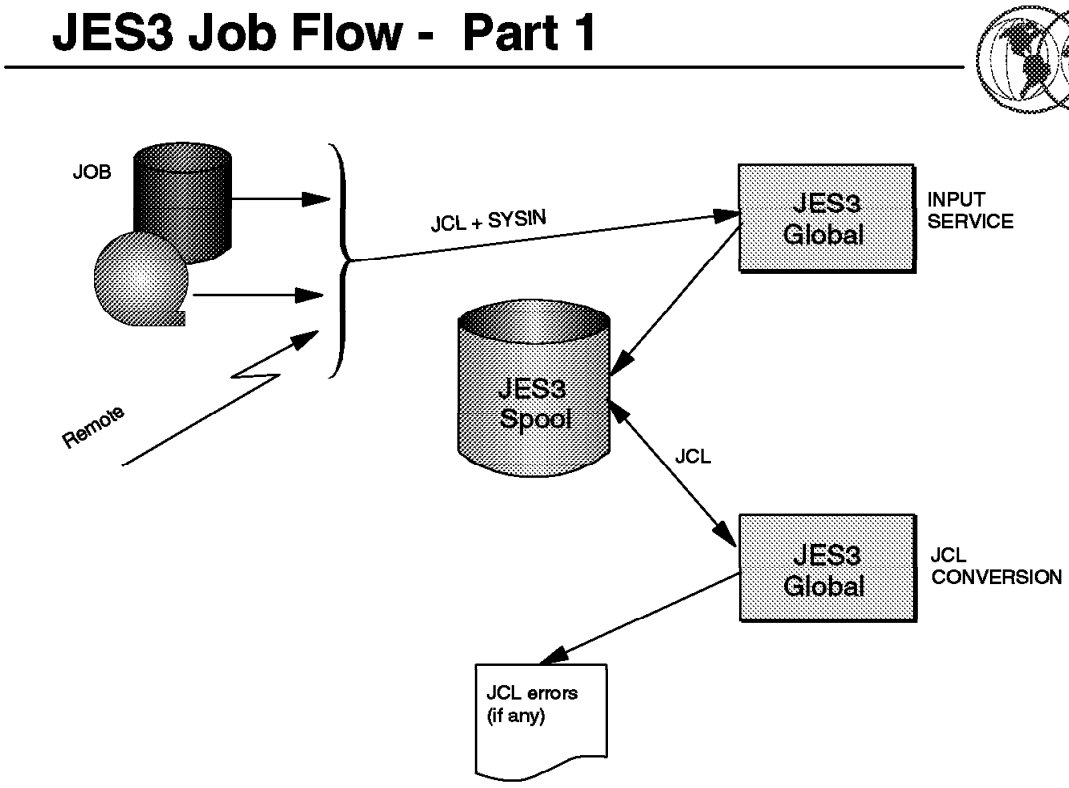


Figure 119. JES3 job flow

3.8.9 JES3 phases of job processing

Following are the first two phases of processing for a job in a JES3 system:

- Input service

  JES3 initially reads all jobs into the global and assigns, to each job, a unique JES3 job number from the available job number pool. Jobs can be submitted from a locally-attached tape, disk, or card reader. In addition, jobs can be submitted from remote job processing (RJP) workstations, time-sharing option (TSO/E) terminals, other systems in a job entry network, or by the internal reader.

  START and MOUNT commands and TSO/E LOGONs cause jobs to be started from predefined procedures. Input service processes the JCL created for these jobs in the same manner as any other standard job.

  Jobs initially placed on direct-access storage devices (DASD) and subsequently analyzed by JES3 input service are placed on the JES3 spool.

- Converter/interpreter processing (JCL conversion)

  JES3 chooses the address space where it converts the job′s JCL. The selected address space then reads the job′s JCL from spool and converts and interprets the JCL. During this processing, JES3 flushes any job with JCL errors and determines the job-referenced data sets that require volume mounting. JES3 passes the information about the required resources to the JES3 main device scheduler (MDS) if a job requires devices, volumes, or data sets.

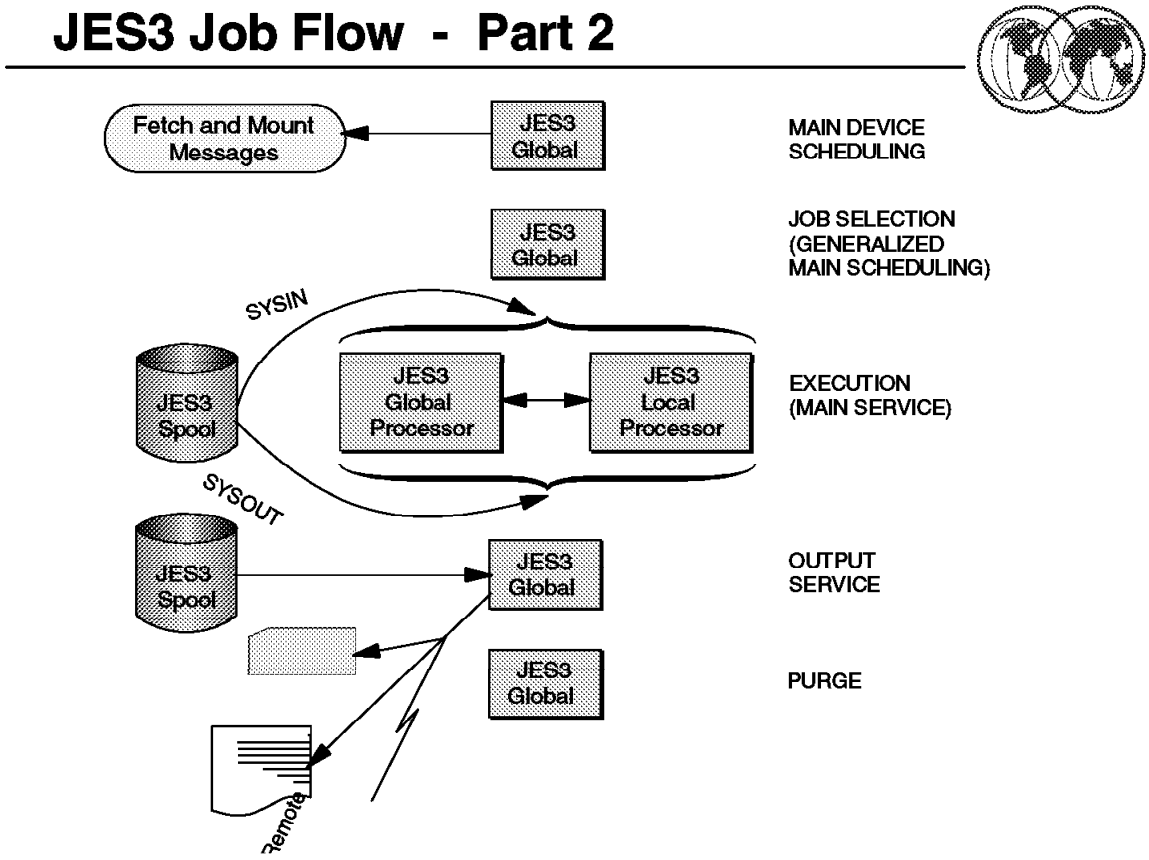


Figure 120. JES3 job flow

3.8.10 JES3 phases of job processing

After the first two phases where jobs are entered into the system and then passed through the converter and interpreter, the next phases are:

- Main device scheduling (job resource management)

  MDS ensures that resources (devices, volumes, and data sets) needed by the job are allocated before the job processes. MDS sends fetch messages to the tape and disk libraries and mount messages to the setup operators to mount the volumes needed.

- Generalized main scheduling (job selection)

  JES3 schedules the job for processing based on specifications determined by the installation.

- Processing (main service - job execution)

  Jobs can run on the global or a local. OS/390 controls the job during this phase. During processing, output generated by the job is usually written to the JES3 spool.

- Output processing

  Once processing is complete, JES3 processes the job′s output data from the JES3 spool. Output data sets are printed and punched by JES3 output service when an available device matches the data set′s requirements, such as forms, carriage forms control buffer (FCB), and train. JES3 output service informs the operator of any setup requirements of the data sets. The devices involved can be either local or remote, as defined by the job.

- Purge processing

After output processing is complete, the purge function releases all spool space associated with the completed job.

3.8.10.1 JES3 from a system programmer′s point of view

One way of visualizing JES3 is as a control program that performs job entry and job exit services. Another way of visualizing JES3 is as a language. Saying that JES3 is a control program is analogous to saying FORTRAN is a compiler. In reality, there is a FORTRAN language and a FORTRAN compiler. Similarly, there is a JES3 language and there is a JES3 program. Just as a FORTRAN programmer uses the FORTRAN language to define work the compiler is to do, JES3 system programmers use the JES3 language to define the work that JES3 is to do.

The JES3 language is made up of initialization statements, control statements, and commands. Operators and application programmers use parts of the language, but it is system programmers who define the total JES3 environment. Often, what is specified on JES3 initialization statements can be overridden with JES3 commands or control statements. While system programmers may not themselves use commands or control statements, they may have to instruct operators or application programmers on when and how they should be used.

A further level of defining how JES3 should do its work comes in the form of installation exits supplied with JES3. For example, in the area of controlling the print output from jobs, a system programmer can:

- Define initial values in the initialization stream
- Allow the operator to make certain modifications to the initial values
- Write an installation exit routine to further modify or control what should happen to the output.

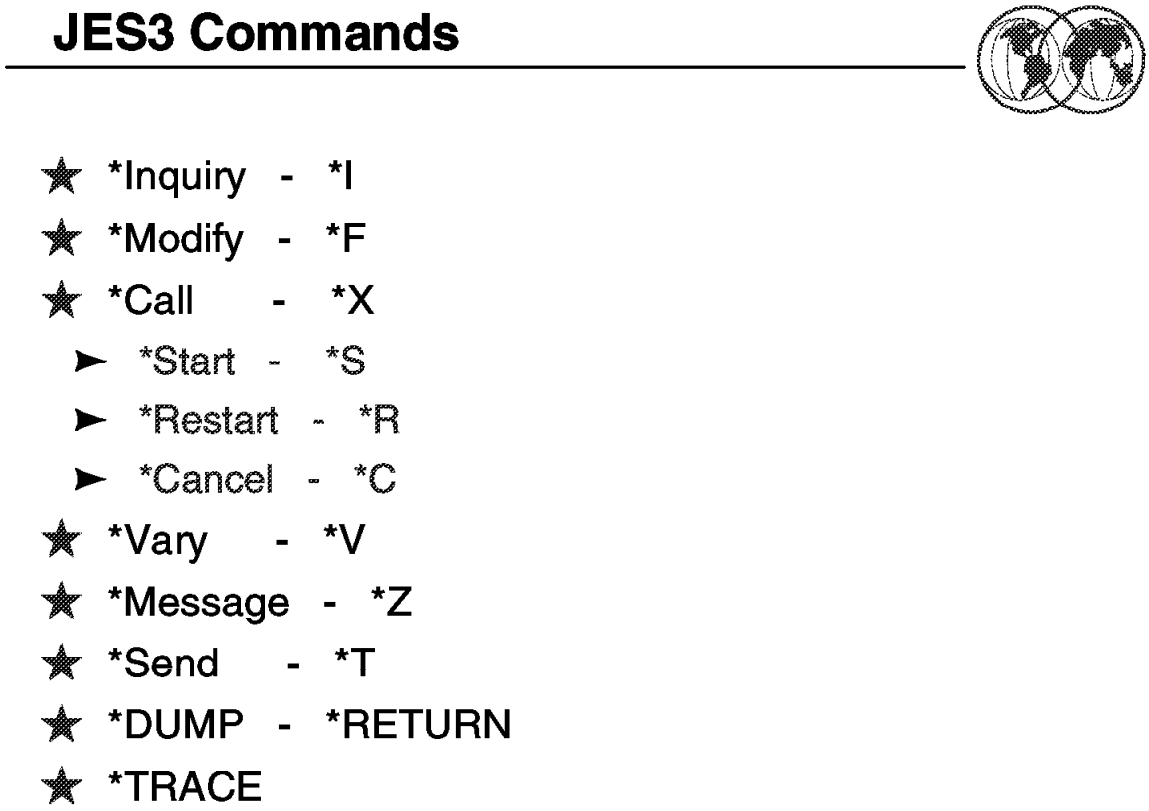


Figure 121. JES3 commands

3.8.11 JES3 operator control

Commands are requests you make to the system. You can use commands to control devices, to change specifications previously made to the system, or to display information about the operator's console. The first element of a command is a prefix that informs JES3 that a command is being entered. The default JES3 command prefix is the character (*).

These are the basic JES3 commands and their purposes:

- *INQUIRY commands are used to request information. No action will be taken except that the requested information will be displayed.
- *MODIFY commands are used to change specifications given in jobs or previous commands, or during initialization.
- *CALL, *START, *RESTART, *CANCEL, and *FAIL commands are used to control dynamic support programs (DSPs), such as utilities.
- *VARY commands and the *MODIFY,V command are used to control devices and other JES3 resources.
- *SWITCH and *FREE commands are used exclusively to control RJP consoles.
- *MESSAGE commands are used to communicate with other consoles and with remote nodes.
- *SEND commands are used to route commands to a remote node.
- *DUMP and *RETURN commands are used to end JES3.
- *TRACE command is used to trace certain JES3 events.

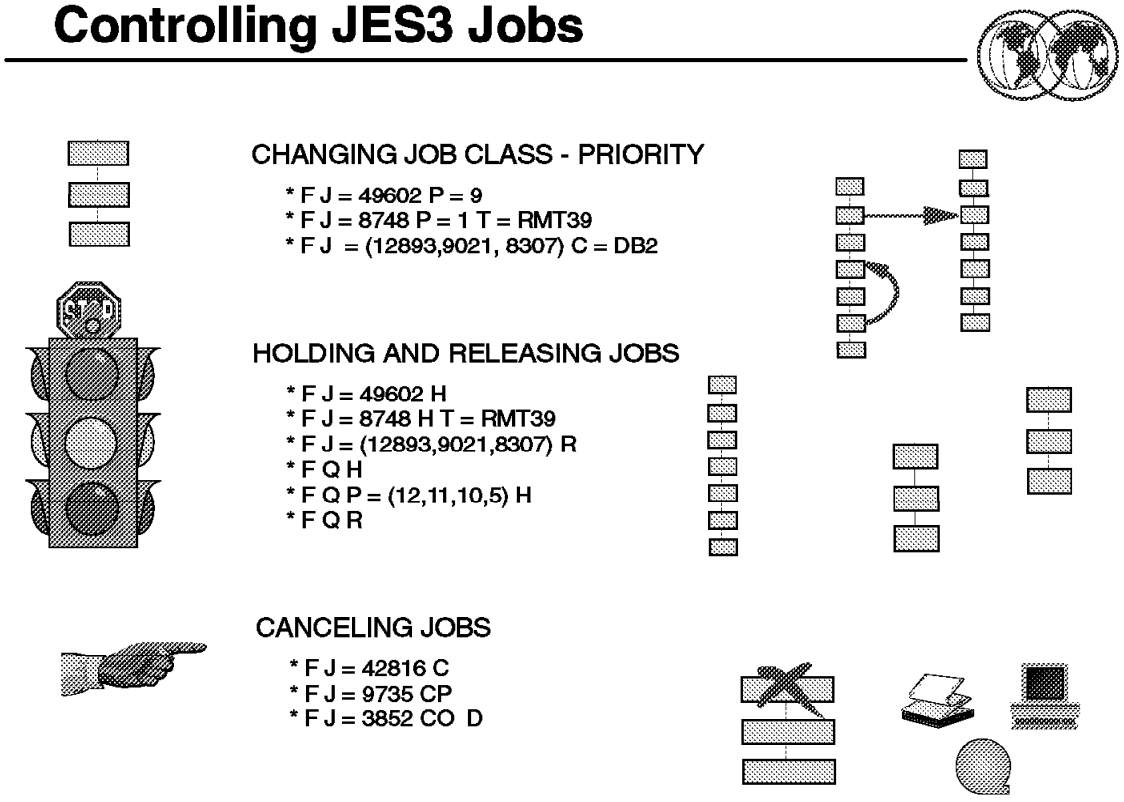


Figure 122. Controlling JES3 jobs

3.8.12 Controlling JES3 jobs

Use the *MODIFY,J= or the *F,J= command to:

- Hold a job, *F,J=nnnn H
- Release a job, *F,J=nnnn R
- Cancel a job, *F,J=nnnn C or CP or CO
- Change a job′s priority, *F,J=nnnn P=nn
- Change a job′s job class, *F,J=nnnn C=class
- Change a job′s JESMSGLG logging status, *F,J=nnnn LOG or NOLOG

Use the *MODIFY,Q or *F,Q command to:

- Hold or release all the jobs in the JES3 job queue, *F Q H or R
- Hold or release jobs of a designated priority, *F Q P=nnn H or R

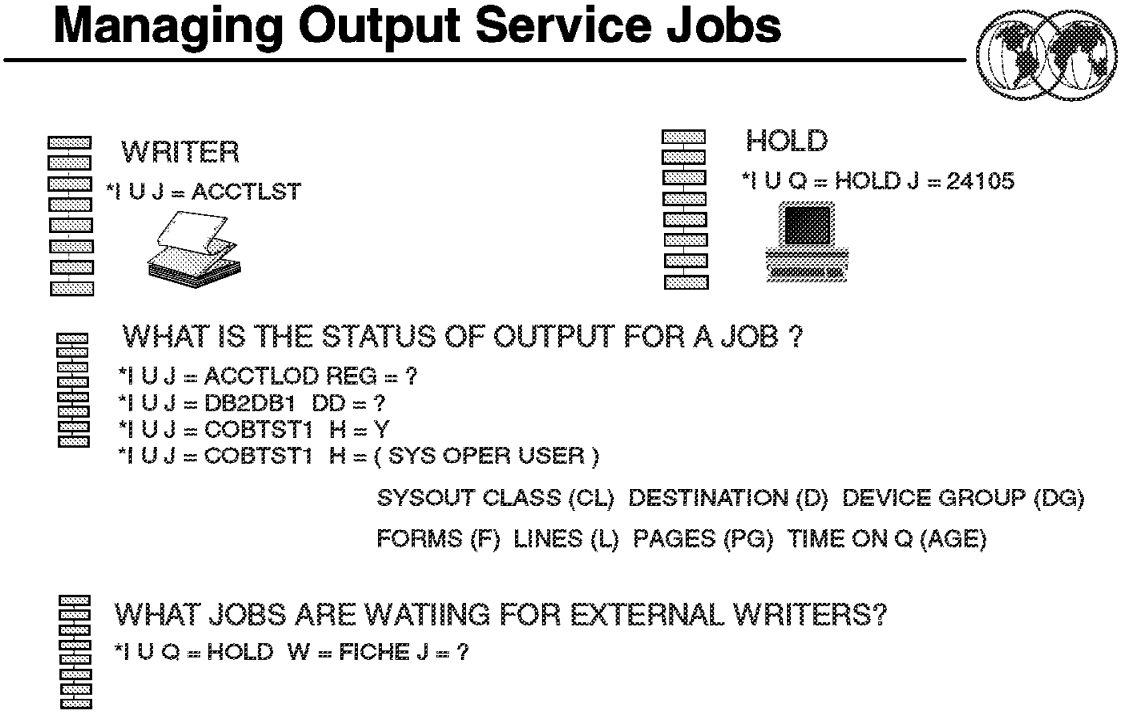


Figure 123. Managing output service jobs

3.8.13 Managing output service jobs

Use the *INQUIRY,U or the *I,U command to display job output in a JES3 system. The job output can be at various places within the system and your selection of the proper ″Q = ″ keyword value on the *INQUIRY,U command dictates what output you want.

The choices are generally the following:

- *INQUIRY,U,Q=BDT

  To display SNA/NJE job output.

- *INQUIRY,U,Q=HOLD

  To display job output on the HOLD queue.

- *INQUIRY,U,Q=WTR

  To display job output on the WTR service queue.

- *INQUIRY,U

  To display job output on the WTR service queue.

The commands may have a length of 126 characters if the command is issued from an input device that permits that command length.

JES3 Start Procedure

```
//JES3      PROC
//IEFPROC   EXEC PGM=IATINTK,DPRTY=(15,15)
//STEPLIB   DD DISP=SHR,DSN=SYS1.VnRnMn.SIATLIB
//CHKPNT    DD DISP=SHR,DSN=SYS1.JES3CKPT
//CHKPNT2   DD DISP=SHR,DSN=SYS1.JS3CKPT2
//JES3JCT   DD DISP=SHR,DSN=dsn
//spool1    DD DISP=SHR,DSN=dsn
    .
    .
//spoolnn   DD DISP=SHR,DSN=dsn
//JES3OUT   DD DISP=SHR,DSN=dsn
//JES3SNAP  DD UNIT=AFF=JES3OUT
//JESABEND  DD UNIT=AFF=JES3OUT
//SYSABEND  DD UNIT=AFF=JES3OUT
//IATPLBST  DD DISP=SHR,DSN=SYS1.PROCLIB
                          .
                          .
//IATPLBnn                .
//JES3IN    DD DISP=SHR,DSN=SYS1.PARMLIB(JES3IN00)

//JES3DRDS  DD DISP=SHR,DSN=dsn
```

Figure 124. JES3 start procedure

3.8.14 How to start JES3

JES3 runs as a started task in the OS/390 environment. Therefore, OS/390 must be active before you can start JES3. Moreover, JES3 must be active on the global before you can start JES3 on the local mains.

The required JES3 procedure dd statements are explained as follows:

IEFPROC Specifies the name of the JES3 job step task, load module IATINTK.

CHKPNT and CHKPNT2 Define the the JES3 checkpoint data set(s). At least one of the two checkpoint data sets must be allocated and cataloged prior to JES3 operation. Each checkpoint data set must be allocated as a single extent which begins and ends on a cylinder boundary. The size of each data set should be at least 2 cylinders on a 3380, 3390 or 9345 spool volume.

JES3JCT Defines the JES3 job control table (JCT) data set. This data set must be allocated and cataloged prior to JES3 operation. The data set must be large enough to accommodate the maximum number of JCT records to be allocated concurrently during normal system operation.

Spool1 and Spoolnn Define the spool data sets. The installation selects the ddnames and data set names for these statements. The ddname for this statement must be the same ddname specified on the BADTRACK, FORMAT, or

	TRACK initialization statements. Spool data sets must be allocated and cataloged prior to JES3 operation.
JES3OUT	Defines the data set upon which the JES3 initialization stream and initialization error messages are printed. This data set is de-allocated after initialization completes You can tailor the block size (BLKSIZE) and logical record length (LRECL) values to improve performance. The values you can specify are device-dependent.
IATPLBST	Defines the installation's standard procedure library.
JES3IN	Defines the data set containing the JES3 initialization stream. This data set must be a blocked or unblocked partitioned data set. In our example, the initialization stream is read from SYS1.PARMLIB(member JES3IN00).

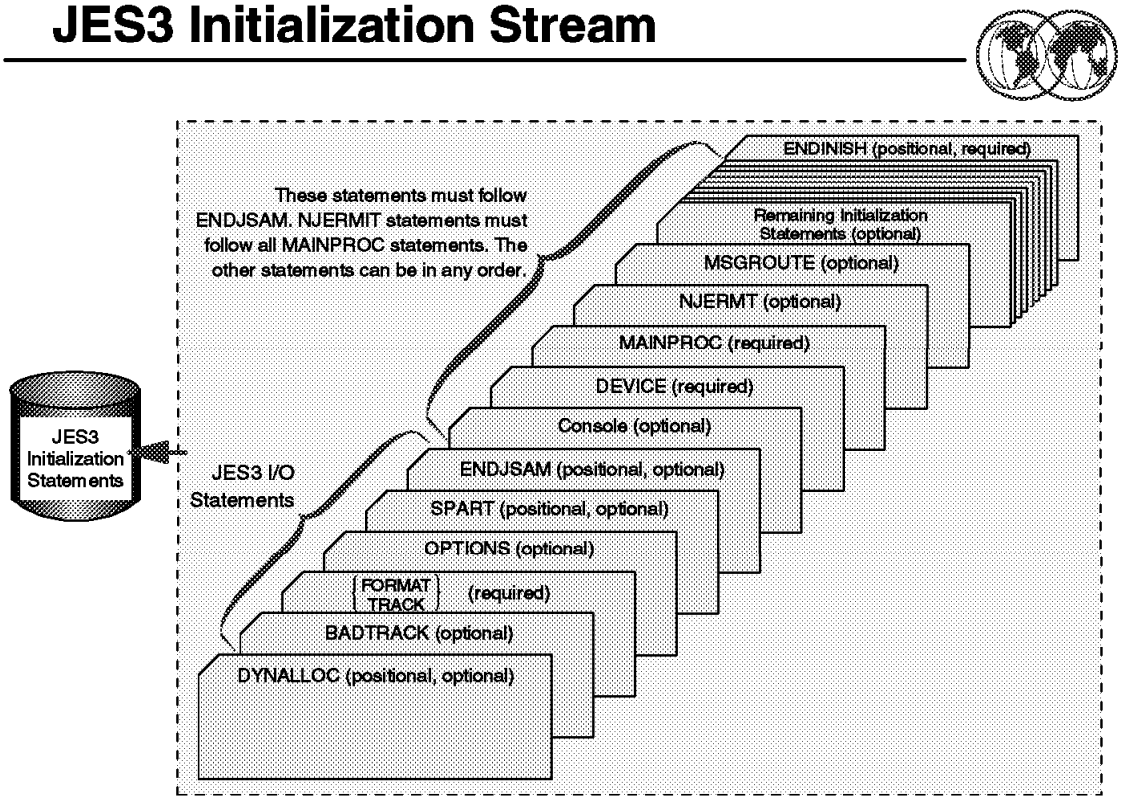


Figure 125. JES3 initialization deck

3.8.15 JES3 initialization deck

Here are the required JES3 initialization statements:

- **DEVICE,DTYPE=SYSMAIN,JNAME=main1,JUNIT=(,main2,,¬ OFF|ON],...)**

  Use this form of the DEVICE statement (there are two other forms of the DEVICE statement) to define the initial status of mains in a JES3 complex. You must code one of these DEVICE statements for each main on every processor in the complex, as follows:

  ```
  DEVICE,DTYPE=SYSMAIN,JNAME=SY1,JUNIT=(,SY1,,ON,)
  ```

- **ENDINISH**

  Use the ENDINISH statement to identify the end of the initialization statements in the initialization stream.

- **ENDJSAM**

  Use the ENDJSAM initialization statement to indicate the end of the JES3 spool initialization statements.

- **FORMAT,DDNAME=ddname,SPART=partitionname {,STT=(cylnum,cylnum)} {,STTL=(cylnum,numtrkgps)}**

  Use the FORMAT statement to specify formatting for a data set residing on a direct-access spool volume during initialization. Specify this statement only when introducing an unformatted volume into a JES3 system or when you change the BUFSIZE parameter on the JES3 BUFFER initialization statement as follows:

```
FORMAT,DDNAME=SPOOL1,STT=(30,31),SPART=PART1
The corresponding DD statement in the JES3 start procedure is:
//SPOOL1 DD DSN=dsn,DISP=OLD,UNIT=SYSDA,VOL=SER=xxxxxx
```

- **MAINPROC,NAME=main,SYSTEM=JES3,FIXPAGE=,ID=msgprefix,MDEST=, PRTPAGE=,RID=,SELECT=,SID=,SPART=partitionname,TRKGRPS=,USRPAGE=**

  Use the MAINPROC initialization statement to define a uniprocessor or a multiprocessor as a JES3 main. The initialization stream must include one MAINPROC statement for each main that you wish to define to JES3. Each MAINPROC statement must have a DEVICE statement that specifies DTYPE=SYSMAIN associated with it.

  The NAME=main specifies the name of a JES3 main. The name should match the purpose of the main, or at least the hardware it is using. You also use this name in operator commands and in the JES3 CLASS, DEVICE, GROUP, MSGROUTE, and SETACC initialization statements to refer to the main. It must be the first or secound parameter on the statement.

  The SPART=partitionname specifies the spool partition that JES3 is to use for jobs that execute on the main defined by this statement. To specify the default spool partition, omit the SPART parameter.

- **TRACK,DDNAME=ddname,SPART=partitionname {,STT=(cylnum,cylnum)} {,STTL=(cylnum,numtrkgps)}**

  Use the TRACK initialization statement to replace a corresponding FORMAT statement in an initialization stream after the spool data set specified by the FORMAT statement has been formatted. The TRACK statement indicates that the corresponding data set has been formatted.

  ```
  TRACK,DDNAME=SPOOL1
  TRACK,DDNAME=SPOOL2
  Associated DD statements in the JES3 procedure:
  //SPOOL1 DD DSN=dsn,DISP=OLD,UNIT=3330,VOL=SER=xxxxxx
  //SPOOL2 DD DSN=dsn,DISP=OLD,UNIT=3330,VOL=SER=MVSRW2
  ```

Depending on how your OS/390 initial program load (IPL) procedures are set up, JES3 can be started automatically or you can start JES3 manually by entering an OS/390 START command from the master console. The manual method allows you to determine if the IPL and system configuration conditions are acceptable before starting JES3. If the JES3 subsystem definition in the IEFSSNxx parmlib member specifies the NOSTART subparameter, JES3 must be started manually. Otherwise JES3 will start automatically.

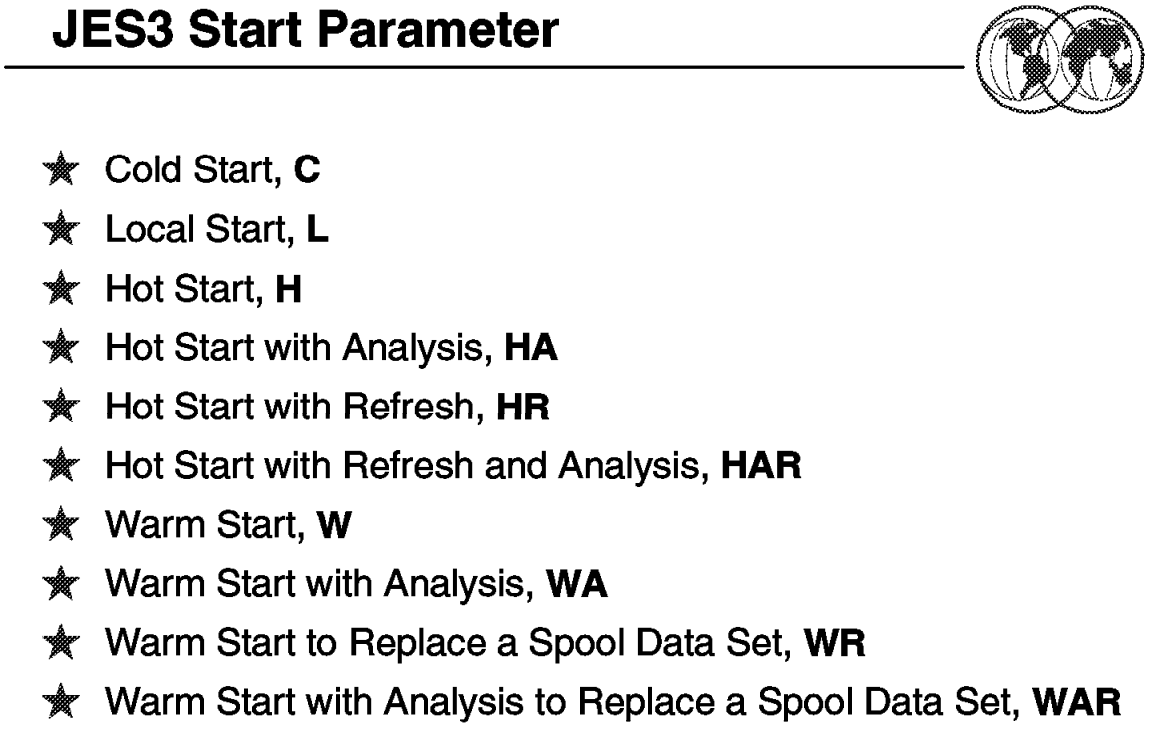


Figure 126. JES3 start parameter

3.8.16 JES3 start parameter

The types of starts and restarts for the JES3 global processor are shown on the visual. You must use a cold start when starting JES3 for the first time. For subsequent starts (restarts), you can use any one of the start types, depending on the circumstances at the time. Thus, the other types of starts are actually restarts.

JES3 initialization statements are read as part of cold start, warm start, and hot start with refresh processing. If JES3 detects any error in the initialization statements, it prints an appropriate diagnostic message on the console or in the JES3OUT data set. JES3 ends processing if it cannot recover from the error. In this case, you must notify the system programmer to ensure that the error is corrected before restarting JES3.

JES3 configuration and processing options are specified with the initialization statements. Because many options affect the overall performance of the system, initialization statements must be provided by your system programmer. Starting a JES3 on a local main you can use the ROUTE command to start JES3 on all the local processors, once JES3 global initialization is complete, for example:

```
ROUTE *OTHER,S JES3
```

Use a local start to restart JES3 on a local main after a normal shutdown on the local, after JES3 ends due to a failure in either the local JES3 address space or OS/390, or after partitioning a multiprocessor. You must also start each local main after you perform a cold start or any type of warm start on the global or after you use a hot start to remove or reinstate a spool data set on the global processor.

You can perform a local start any time the global is active. Local start processing uses the initialization stream placed on the spool data sets during global main initialization.

Note: Do not start a main if another main having the same name is active in the complex.

Before you begin, be sure that all spool data sets available to the global are also available to this local.

If you are performing an IPL of the local main and the MSTRJCL contains the automatic START command, the command is displayed on your console immediately after the OS/390 master scheduler is initialized. You can enter the OS/390 command:

```
START  JES3
```

The first message from JES3 is:

```
  IAT3040     STATUS OF JES3 PROCESSORS IN COMPLEX
               main(status) (,main(status)...)
```

A series of entries on the following lines of the message contains the name of each main followed by a code for that main's status as recorded in the complex status record (CSR). During a start for the system, this entry (the status of the global) is:

```
  main< >
On subsequent starts, this entry could be:
  main<IN>
```

This indicates that the previous start ended abnormally during initialization; this main is still recorded in the CSR as being in initialization as follows:

```
  main<UP>
```

This main is recorded in the CSR as currently being active (up). Consult with the system programmer before continuing with a cold start.

Next, JES3 issues the following message:

```
nn IAT3011 SPECIFY JES3 START TYPE:(C, L, H, HA, HR, HAR, W, WA, WR, WAR
OR CANCEL)
```

To continue the start, enter the appropriate letters shown in the message, as follows:

```
       R nn,C
To cancel the start, enter:
       R nn,CANCEL
```

Note: Because cold start processing involves reinitializing the spool and losing all jobs, JES3 issues the following message requiring you to confirm your request for a cold start before JES3 takes irreversible action:

```
  nn IAT3033     CONFIRM JES3 COLD START REQUEST (U) OR CANCEL
```

To continue the cold start, enter:

```
       R nn,U
```

To cancel the cold start, enter:

```
       R nn,CANCEL
```

Any other reply restarts the sequence, beginning with message IAT3011.

After JES3 initialization has finished, you should start the job segment scheduler (JSS). You start job scheduling after JES3 issues message IAT3100 notifying you that initialization processing is complete on the global:

```
IAT3100  JES3 xxxxx SYSTEM type START ON yyyy.ddd AS main
```

where xxxxx is the release level of JES3; type is COLD, WARM, or HOT; yyyy.ddd is the Julian date; and main is the JES3 name of the main.

Before starting job scheduling, you can use JES3 commands to cancel jobs, change the status of jobs, and change the status of devices. During a hot start with analysis, you can release jobs in spool hold status after reinstating a spool data set that contains data for the jobs, and you can vary devices online or offline. You can make adjustments for any system data that might have been lost during the restart. You can also make any changes to the system that were made before a hot start or a warm start but did not remain in effect after the restart.

When you are satisfied that the system is ready to begin processing, enter a *START,JSS command to start job scheduling:

```
*START,JSS
```

After you enter the *START,JSS command, ensure that the global is varied online to JES3. If it is not, enter the *(MODIFY,)V,main,ON command to vary the processor online, ensuring that the subsystem interface, the OS/390 system commands, and the system log are initialized. JES3 then issues the following message:

```
IAT2645     ***** main CONNECT COMPLETE *****
```

If you do not want the global to run jobs, you can now vary the main offline to JES3 scheduling with the *(MODIFY,)V,main,OFF command. At this point, you can resubmit any jobs that were canceled or whose processing was altered as a result of the restart.

3.8.17 How to stop JES3

Stopping local processors

Before you remove a local main for maintenance or other reasons, allow processing jobs to complete normally. Use the following steps:

1. Enter a *F, V,main,OFF command for the local main to prevent JES3 from scheduling any further jobs on the processor.
2. Enter the *RETURN command with the proper password to end JES3 after all jobs on the local main have completed processing.
3. Enter the HALT EOD command on the local main to ensure that important statistics and data records in storage are not permanently lost.

Your installation may not want to wait for all jobs to complete normally, for example:

- Jobs will not end due to system problems (hardware and software)
- An IPL or JES3 restart is scheduled to take place at a predetermined time and jobs will not be able to complete. In this case, you must make the decision to delay the IPL or to cancel the jobs.

Note: Enter the HALT EOD command only if you are performing an IPL or SYSTEM RESET, not to restart JES3.

Stopping the global processor

Before stopping the global, you should stop the local mains as described before. You should stop all JES3 processing by entering the *F,Q,H command, which puts all jobs in hold status before stopping JES3. System initiators, printers, and punches do not begin any new work and become inactive after completing their current activity. Jobs in JES3 queues remain in their current position.

You should also queue the system log for printing by entering the WRITELOG command. This prevents log messages from being lost if you later restart JES3 with a hot start.

Once all system activity has completed, enter the *RETURN command to end JES3.

```
*RETURN,password,FSS=fssname  -or-
*RETURN,password,FSS=ALL      -or
*RETURN,password,FSS=NONE
```

After you enter the *RETURN command, enter the HALT EOD command to ensure that important statistics and data records in storage are not permanently lost. As a result of this command, the internal I/O device error counts are stored in the SYS1.LOGREC data set; the SMF buffers are emptied onto one of the SYS1.MANx data sets; and the system log, if still active, is closed and put on the print queue. When these actions are complete, the system issues message IEE334I, stating that the HALT EOD operation was successful.

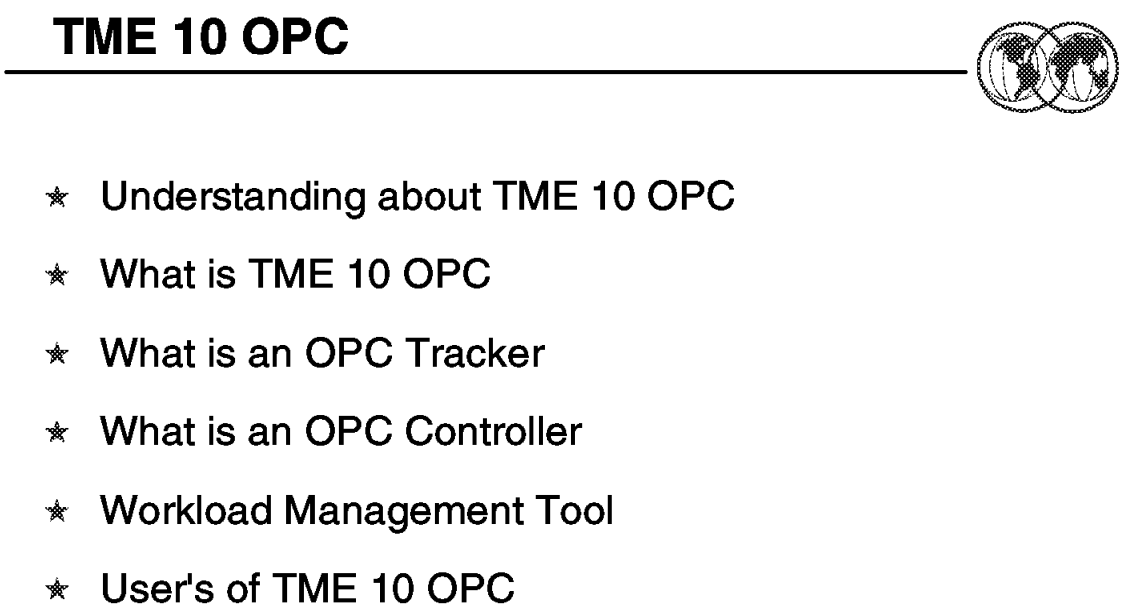


Figure 127. TME 10 OPC

3.9 Operations planning and control (OPC)

The TME 10 Operations Planning and Control (TME 10 OPC) Licensed Program is IBM′s foundation for enterprise workload management. TME 10 OPC provides a comprehensive set of services for managing and automating the workload. Whether you manage a single-image OS/390 system or multivendor networks and systems from a single point of control, TME 10 OPC helps you manage and automate the production workload.

TME 10 OPC builds operating plans from your descriptions of the production workload.

TME 10 OPC consists of a base product, the tracker, and a number of features. All the systems in your complex require the base product. The *tracker* is the link between the system that it runs on and the TME 10 OPC controller.

One OS/390 system in your complex is designated the controlling system and runs the controller feature. From this system, you can automatically plan, control, and monitor your entire production workload. Only one controller feature is required, even when you want to start standby controllers on other OS/390 systems in a sysplex.

3.9.1 Who uses TME 10 OPC

- The **scheduler** maintains the current and future production processing across your enterprise.
  - Automatically scheduling all production workload activities.
  - Automatically resolving the complexity of production workload dependencies and driving the work in the most efficient way.
  - Supporting the simulation of future workloads on the system. The scheduler can evaluate, in advance, the effect of changes in production workload volumes or processing resources.
  - Giving a real-time view of the status of work as it flows through the system so that the scheduler can quickly:
    - Respond to customer queries about the status of their work.
    - Identify problems in the workload processing.
  - Providing facilities for manual intervention.
  - Managing many workload problems automatically. The TME 10 OPC production-workload-restart facilities, hot standby, automatic recovery of jobs and started tasks, and dataset cleanup provide the scheduler with comprehensive error- and disaster-management facilities.
  - Providing a log of changes to the TME 10 OPC production workload data through the audit-trail facility. This assists the scheduler in resolving problems caused by user errors.
  - Managing unplannable work.
- The **operations manager** uses the reporting, planning, and control functions in:
  - Improving the efficiency of the operation.
  - Improving control of service levels and quality.
  - Setting service level agreements for end-user applications and for services provided.
  - Improving relationships with end-user departments.
  - Increasing the return on your IT investment.
  - Developing staff potential.
- The **Shift Supervisor** uses TME 10 OPC especially in multisystem complexes, where local and remote systems are controlled from a central site. Also it can help the shift supervisor in:
  - Monitoring and controlling the flow of production work through multisystem complexes.
  - Controlling the use of mountable devices.
  - Separating information about work status from system and other information.
  - Providing end users with status information directly.
  - Managing the workload if a system failure occurs.
  - Making changes to the current plan in response to unplanned events, such as equipment failures, personnel absences, and rush jobs.
- The **application programmer**. The TME 10 OPC user-authority checking enables application development groups to use all the planning and control functions of TME 10 OPC in parallel with--but in isolation from--production systems and services. For the application programmer, it can be a valuable tool for application development staff when they are:
  - Packaging new applications for regular running
  - Testing new JCL in final packaged form
  - Testing new applications and changes to existing ones

  - Restarting or rerunning unsuccessful jobs.
- The **console operators** can be freed from the these time-consuming tasks:
  - Starting and stopping started tasks.
  - Preparing JCL before job submission.
  - Submitting jobs.
  - Verifying the sequence of work.
  - Reporting job status.
  - Performing dataset cleanup in recovery and rerun situations.
  - Responding to workload failure.
  - Preparing the JCL for step-level restarts.
- The **workstation operators** gets help in:
  - Providing complete and timely status information.
  - Providing up-to-date ready lists that prioritize the work flow.
  - Providing online assistance in operator instructions.
- The **end users** and **help desks** can be informed about the status of workload processing. They can use the ISPF dialogs or the Workload Monitor/2 to check the status of the processing of their applications themselves, from a personal workstation. End users can make queries using the Workload Monitor/2 without having to be familiar with TME 10 OPC, ISPF, or TSO, and without having to be logged on to a local system. The Workload Monitor/2 also provides alerts that can automatically advise end users when their work has completed or ended in error. The help desk can use the Workload Monitor/2 in the same way to answer queries from end users about the progress of their workload processing.

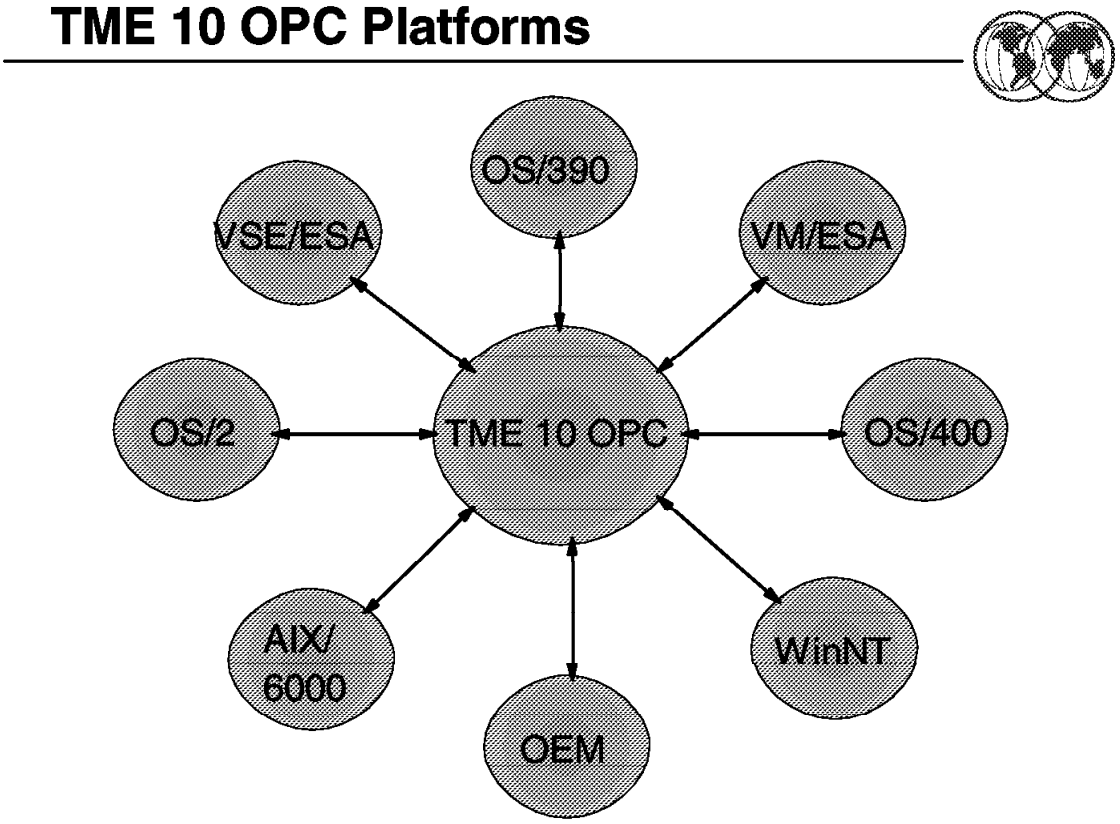


Figure 128. TME 10 OPC platforms

3.9.2 TME 10 OPC platforms

The Tivoli Management Environment 10 Operations Planning and Control for OS/390 (TME 10 OPC) is the state-of-the-art production workload manager, designed to help you meet your present and future data processing challenges. Its scope encompasses your entire enterprise information system, including heterogeneous environments.

Pressures on today's DP environment are making it increasingly difficult to maintain the same level of services to customers. Many installations find that their batch window is shrinking. More critical jobs must be finished before the morning online work begins. Conversely, requirements for the integrated availability of online services during the traditional batch window put pressure on the resources available for processing the production workload. More and more 7/24 (7 days a week, 24 hours a day) is not only a DP objective but a requirement.

Users and owners of DP services are also making more use of batch services than ever before. The batch workload tends to increase each year at a rate slightly below the increase in the online workload. Combine this with the increase in data usage by batch jobs, and the end result is a significant increase in the volume of work.

Furthermore, there is a shortage of people with the required skills to operate and manage increasingly complex DP environments. The complex interrelationships between production activities--between manual and machine tasks--have become unmanageable without a workload management tool.

TME 10 OPC provides leading-edge solutions to problems in production workload management. It can automate, plan, and control the processing of your enterprise's entire production workload, not just the

batch subset. TME 10 OPC functions as an “automatic driver” for your production workload to maximize the throughput of work, and optimize your resources, but also allows you to intervene manually as required.

When TME 10 OPC interfaces with other system management products, it forms part of an integrated automation and systems management platform for your DP operation.

TME 10 OPC forms operating plans based on user descriptions of the operations department and its production workload. These plans provide the basis for your service level agreements and give you a picture of the production workload at any point in time. You can simulate the effects of changes in your production workload and in resource availability by generating trial plans.

Good planning is the cornerstone of any successful management technique. Effective planning also helps you maximize return on your investments in information technology.

TME 10 OPC automates, monitors, and controls the flow of work through your enterprise′s entire DP operation--on both local and remote systems. From a single point of control, TME 10 OPC analyzes the status of the production work and drives the processing of the workload according to installation business policies. It supports a multiple end-user environment, enabling distributed processing and control across sites and departments within your enterprise.

OS/390 OPC Configuration

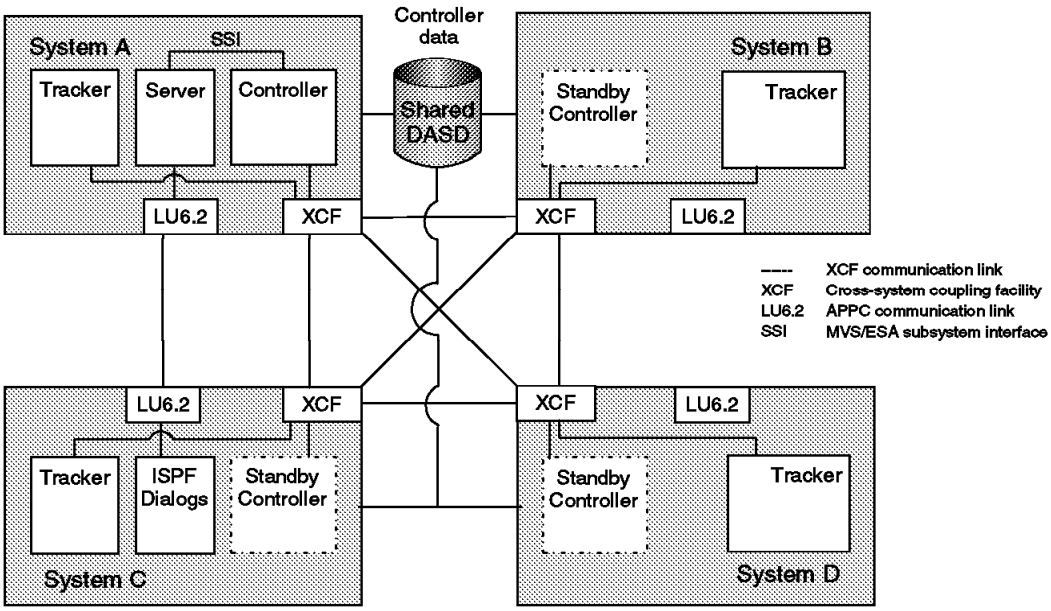


Figure 129. OS/390 OPC Configuration

3.9.3 What is a TME 10 OPC Tracker

A tracker is required for every controlled OS/390 system in an TME 10 OPC configuration. This includes, for example, local controlled systems within shared DASD or sysplex configurations.

The tracker runs as an OS/390 subsystem and interfaces with the operating system (through JES2 or JES3, and SMF), using the subsystem interface and the operating system exits. The tracker monitors and logs the status of work, and passes the status information to the controller via shared DASD, XCF, or ACF/VTAM.

You can exploit the OS/390 cross-system Coupling Facility (XCF) to connect your local OS/390 systems. Rather than being passed to the controlling system via shared DASD, work status information is passed directly via XCF connections. XCF lets you exploit all of TME 10 OPC′s production-workload-restart facilities and its hot standby function.

The tracker on a remote OS/390 system passes status information about the production work in progress to the controller on the TME 10 OPC controlling system. All communication between TME 10 OPC subsystems on controlling and remote systems is done via ACF/VTAM.

TME 10 OPC lets you link remote systems using ACF/VTAM networks. Remote systems are frequently used locally “on premises” to reduce the complexity of the DP installation.

Every controlled AS/400 system needs the Tracker Agent for OS/400 feature installed. When this feature is used, status information and job logs for the production workload are passed back to the controlling OS/390 system using APPC services. The TME 10 OPC controller submits jobs and

commands to one system in an AS/400 cluster; from there the work is directed to the target AS/400 system.

Controlled UNIX systems require the OPC Tracker Agent feature for the appropriate implementation of UNIX, including the OS/390 Open Edition environment. When one of these features is used, status information and job logs for the production workload are passed back to the controlling OS/390 system using TCP/IP services. The TME 10 OPC controller submits jobs and commands directly to the target UNIX system.

If you use LoadLeveler to manage workload distribution in your heterogeneous workstation environment, the OPC Tracker Agents for AIX and the other UNIX implementations can integrate with LoadLeveler 1.2.

To control the workgroup workload on OS/2 workstations and servers, you need the Tracker Agent for OS/2 feature. When this feature is used, status information and job logs for the production workload are passed back to the controlling OS/390 system using TCP/IP services. The TME 10 OPC controller submits jobs and commands directly to the target workstation system. To control the workgroup workload on Windows NT workstations and servers, you need the Tracker Agent for Windows NT feature. When this feature is used, status information and job logs for the production workload are passed back to the controlling OS/390 system using TCP/IP services. The TME 10 OPC controller submits jobs and commands directly to the target workstation system.

Remote systems controlled by other operating environments can also communicate with the TME 10 OPC controller using supplied programs. Although not a full-function tracker agent today, these routines provide you with the opportunity to centralize control and automate the workload on these operating environments. The supplied sample programs demonstrate communication using a variety of methods. The programs can be tailored for a specific environment.

TME 10 OPC Tracker Agents are functionally equivalent to the base OS/390 tracker, with the exception that automatic dataset cleanup and the job-completion checker (JCC) functions are not provided for non-OS/390 platforms. This means you can use TME 10 OPC functions like automatic recovery and automatic job tailoring for your non-OS/390 workload. Many installations run business applications on a variety of UNIX platforms. TME 10 OPC provides tracker agents that can manage your workload in several operating system environments. At the time of publication, tracker agents for OS/400, AIX/6000, HP-UX, SunOS, Sun Solaris, Pyramid MIPS ABI, Digital OpenVMS, Digital UNIX, AIX, and System/390 Open Edition operating systems are available, as well as the tracker agents for OS/2 and Windows NT platforms, and the base tracker for OS/390.

3.9.4 What is a TME 10 OPC controller

The controller is the focal point of your TME 10 OPC configuration. It uses the information in the database to determine which jobs to run, when they should run, and where they should run. It contains the controlling functions, the dialogs, and TME 10 OPC′s own batch programs. Only one TME 10 OPC controller is required to control the entire TME 10 OPC installation, including local and remote systems.

The system that the controller is started on is called the TME 10 OPC controlling system. TME 10 OPC systems that communicate with the controlling system are called controlled systems. You need to install in your OS/390 system at least one controller for your production systems It can control the entire TME 10 OPC configuration, both local and remote. You can use the TME 10 OPC controller to provide a single, consistent, control point for submitting and tracking the workload on any operating environment.

TME 10 OPC provides open interfaces to let you integrate the planning, scheduling, and control of work units such as online transactions, file transfers, or batch processing in any operating environment that can communicate with OS/390.

3.9.5 TME 10 OPC as a workload management tool

TME 10 OPC allows you to:

- Manage your workload on a variety of platforms from a single point of control.
- Run jobs on the right day at the right time.
- Start jobs in the correct order.
- Resolve complex dependencies between jobs.
- Take into account business days and holidays across divisions, states, and countries.
- Provide plans for the future workload to help you manage peak processing (year-end work, for example).
- Optimize hardware resources by allocating work to specific machines.
- Initiate automatic recovery actions in the event of hardware or software failure.
- Maintain logs of work that has run--available for viewing or post-processing.

Chapter 4. LPA, LNKLST, and authorized libraries

In order to maximize the retrieving modules performance, the MVS operating system, and now OS/390, has been designed to maintain in memory those modules that are needed for fast response to the operating system as well as for the critical applications. Link pack area (LPA), LNKLST, and authorized libraries described here are the cornerstone of the fetching process. The role of VLF and LLA components are described at the appropriated level.

Modules (programs), whether stored as load modules or program objects, must be loaded into both virtual storage and central storage before they can be run. When one module calls another module, either directly by asking for it to be run or indirectly by requesting a system service that uses it, it does not begin to run instantly. How long it takes before a requested module begins to run depends on where in its search order the system finds a usable copy and on how long it takes the system to make the copy it finds available.

You should consider these factors when deciding where to place individual modules or libraries containing multiple modules in the system-wide search order for modules:

- The search order the system uses for modules
- How placement affects virtual storage boundaries
- How placement affects system performance
- How placement affects application performance

4.1 Link pack area (LPA)

Link pack area (LPA) modules are loaded in common storage, shared by all address spaces in the system. Because these modules are reentrant and are not self-modifying, each can be used by any number of tasks in any number of address spaces at the same time. Modules found in LPA do not need to be brought into virtual storage, because they are already in virtual storage.

Modules placed anywhere in LPA are always in virtual storage, and modules placed in FLPA are also always in central storage. Whether modules in LPA, but outside FLPA, are in central storage depends on how often they are used by all the users of the system, and on how much central storage is available. The more often an LPA module is used, and the more central storage is available on the system, the more likely it is that the pages containing the copy of the module will be in central storage at any given time.

LPA pages are only stolen, and never paged out, because there are copies of all LPA pages in the LPA page data set. But the results of paging out and page stealing are usually the same; unless stolen pages are reclaimed before being used for something else, they will not be in central storage when the module they contain is called.

LPA modules must be referenced very often to prevent their pages from being stolen. When a page in LPA (other than in FLPA) is not continually referenced by multiple address spaces, it tends to be stolen. One reason these pages might be stolen is that address spaces often get swapped out (without the PLPA pages to which they refer), and a swapped-out address space cannot refer to a page in LPA.

When all the pages containing an LPA module (or its first page) are not in central storage when the module is called, the module will begin to run only after its first page has been brought into central storage.

Modules can also be loaded into CSA by authorized programs. When modules are loaded into CSA and shared by multiple address spaces, the performance considerations are similar to those for modules placed in LPA. (However, unlike LPA pages, CSA pages must be paged out when the system reclaims them.)

The Link Pack Area

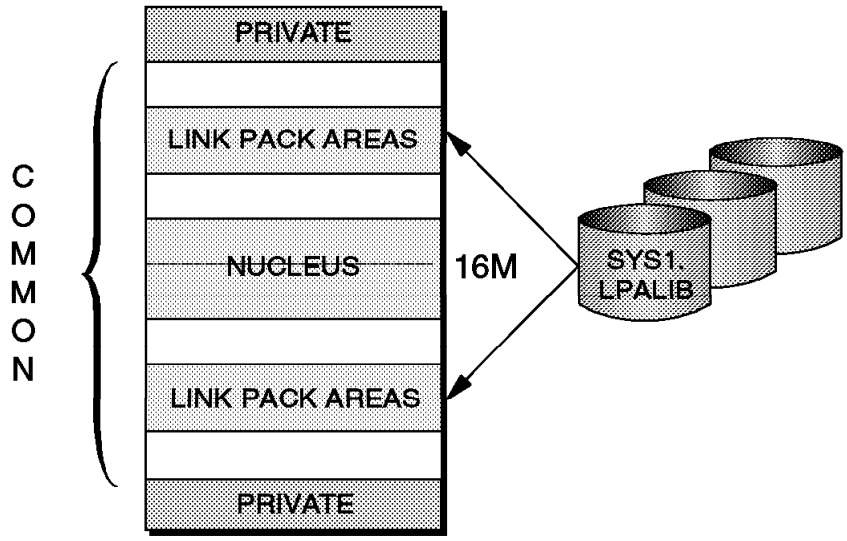


Figure 130. Link pack area

4.1.1 The LPA subareas

The link pack area (LPA) is a section of the common area of an address space. It exists below the system queue area (SQA) and consists of the pageable link pack area (PLPA), then the fixed link pack area (FLPA) if one exists, and finally the modified link pack area (MLPA).

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

Each component of the LPA has a counterpart in the extended common area (that is above the 16 megabyte line) as follows:

FLPA The FLPA exists only for the duration of an IPL. Therefore, if an FLPA is desired, the modules in the FLPA must be specified for each IPL (including quick-start and warm-start IPLs).

An installation can elect to have some modules that are normally loaded in the pageable link pack area (PLPA) loaded into the fixed link pack area (FLPA). This area should be used only for modules that significantly increase performance when they are fixed rather than pageable. Modules placed in the FLPA must be reentrant and refreshable.

It is the responsibility of the installation to determine which modules, if any, to place in the FLPA. Note that if a module is heavily used and is in the PLPA, the system′s paging algorithms will tend to keep that module in central storage. The best candidates for the FLPA are modules that are infrequently used but are needed for fast response to some terminal-oriented action.

PLPA During initialization, both primary and secondary (or duplexed) PLPAs are established. The PLPA is established initially from the LPALST concatenation.

On the PLPA page data set, ASM maintains records that have pointers to the PLPA and extended PLPA pages. During quick start and warm start IPLs, the system uses the pointers to locate the PLPA and extended PLPA pages. The system then rebuilds the PLPA and extended PLPA page and segment tables, and uses them for the current IPL.

If CLPA is specified at the operator′s console, indicating a cold start to be performed, the PLPA storage is deleted and made available for system paging use. A new PLPA and extended PLPA is then loaded, and pointers to the PLPA and extended PLPA pages are recorded on the PLPA page data set.

The secondary PLPA is saved on the optional duplex paging data set for recovery purposes. If any uncorrectable I/O error occurs on a page-in request from the PLPA, the duplexed PLPA is used.

MLPA This area may be used to contain reenterable routines from APF-authorized libraries that are to be part of the pageable extension to the link pack area during the current IPL. The MLPA exists only for the duration of an IPL. Therefore, if an MLPA is desired, the modules in the MLPA must be specified for each IPL (including quick start and warm start IPLs).

The MLPA is allocated just below the FLPA (or the PLPA, if there is no FLPA); the extended MLPA is allocated above the extended FLPA (or the extended PLPA if there is no extended FLPA). When the system searches for a routine, the MLPA is searched before the PLPA.

Note: Loading a large number of modules in the MLPA can increase fetch time for modules that are not loaded in the LPA. This could affect system performance.

The MLPA can be used at IPL time to temporarily modify or update the PLPA with new or replacement modules. No actual modification is made to the quick start PLPA stored in the system′s paging data sets. The MLPA is read-only, unless NOPROT is specified on the MLPA system parameter.

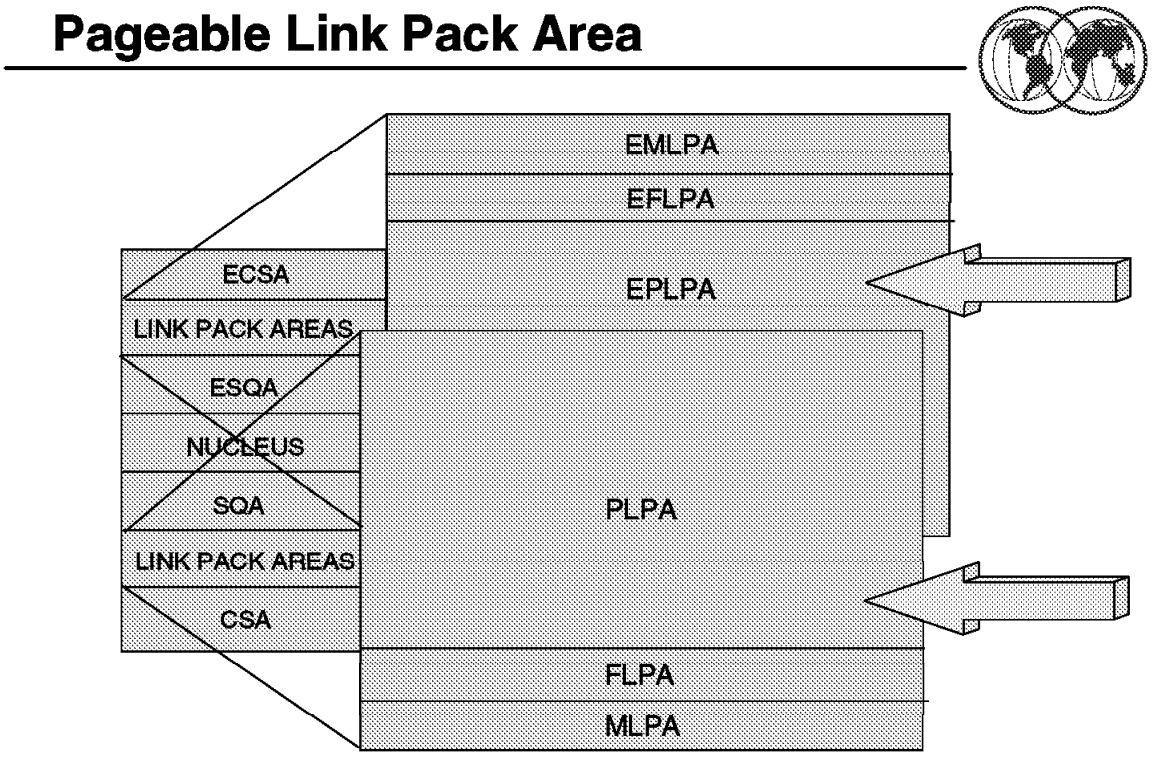


Figure 131. Pageable link pack area

4.1.2 Pageable link pack area (PLPA/extended PLPA)

The PLPA is an area of common storage which is loaded at IPL time (when you do a cold start, specify CLPA in your IPL).

This area contains SVC routines, access methods, and other read-only system programs along with any read-only reenterable user programs selected by an installation that can be shared among users of the system. It is also desirable to place all frequently used refreshable SYS1.LINKLIB and SYS1.CMDLIB modules in the PLPA.

4.1.2.1 Contents of the PLPA or extended PLPA

The PLPA and extended PLPA contain all members of SYS1.LPALIB and any other libraries that are specified in the active LPALSTxx through the LPA parameter in the IEASYSxx or from the operator's console at system initialization. The modules in the PLPA or extended PLPA must be reentrant and executable.

LPA Parmlib Definitions

★ LPALSTxx parmlib member specification

```
LPA= {aa                }
     {(aa,bb,...[,L])}
(YEASYSxx parameter)
```

★ Operator can override parmlib specification

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR OS/390 02.07.00 HBB6607
R 00,LPA=03
IEE600I REPLY TO 00 IS;LPA=03
```

Figure 132. LPA parmlib definitions

4.1.3 Specifying LPA parameters in parmlib

The two characters (A through Z, 0 through 9, @, #, or $) represented by aa (or bb and so forth) shown in the visual, are appended to LPALST to form the name of the LPALSTxx parmlib members.

If the L option is specified, the system displays the contents of the LPALSTxx parmlib members at the operator's console as the system processes the members.

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.

An example of overriding the value of the LPA parameter in the IEASYSxx during system initialization is shown in Figure 133.

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR OS/390 02.05.00 HBB6605
R 00,LPA=03 1
IEE600I REPLY TO 00 IS;LPA=03
```

Figure 133. Example of overriding the IEASYSxx LPA parameter

1 LPALST03 was selected during system initialization.

4.1.3.1 How to specify modules in the PLPA or extended PLPA

Use one or more LPALSTxx members in the SYS1.PARMLIB to concatenate your installation′s program library data sets to SYS1.LPALIB. You can also use the LPALSTxx member to add your installation′s read-only reenterable user programs to the pageable link pack area (PLPA). The system uses this concatenation, which is referred to as the LPALST concatenation, to build PLPA.

Note: The system does not check user catalogs when it searches for data sets to be added to the LPALST concatenation. Therefore, the data sets in the concatenation must be cataloged in the system master catalog.

During nucleus initializing process (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.

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 included the CLPA option). Otherwise, the system builds the PLPA from only those modules named in the SYS1.LPALIB.

Note: The LPALST concatenation can have up 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 are excluded in the concatenation.

Coding a LPALSTxx Parmlib Member

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
EDIT    SYS1.PARMLIB(LPALST05) - 01.01         Line 00000000 Col 001 080
Command ===>                                          Scroll ===> PAGE
***************************** Top of Data ********************************
SYS2.LPALIB,
SYS1.LPALIB,
SYS1.SICELPA,
SYS1.SORTLPA,
NETVIEW.V3R1M0.SCNMLPA1,
ISF.SISFLPA,
ISP.SISPLPA,
EOY.SEOYLPA,
SYS1.ISAMLPA,
REXX.V1R3M0.SEAGLPA,
TCPIP.SEZALPA,
EJES.V2R3M0.SEJELMD1,
CICSTS12.CICS.SDFHLPA(TOTCI2)
***************************** Bottom of Data *****************************
```

Figure 134. Coding a LPALSTxx parmlib member

4.1.4 Coding a LPALSTxx member

Some important syntax rules for the creation of LPALSTxx are:

- 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.

The visual displays an example of LPALSTxx. The last member, CICSTS12.CICS.SDFHLPA(TOTCI2), is a user-cataloged data set on VOLSER TOTCI2.

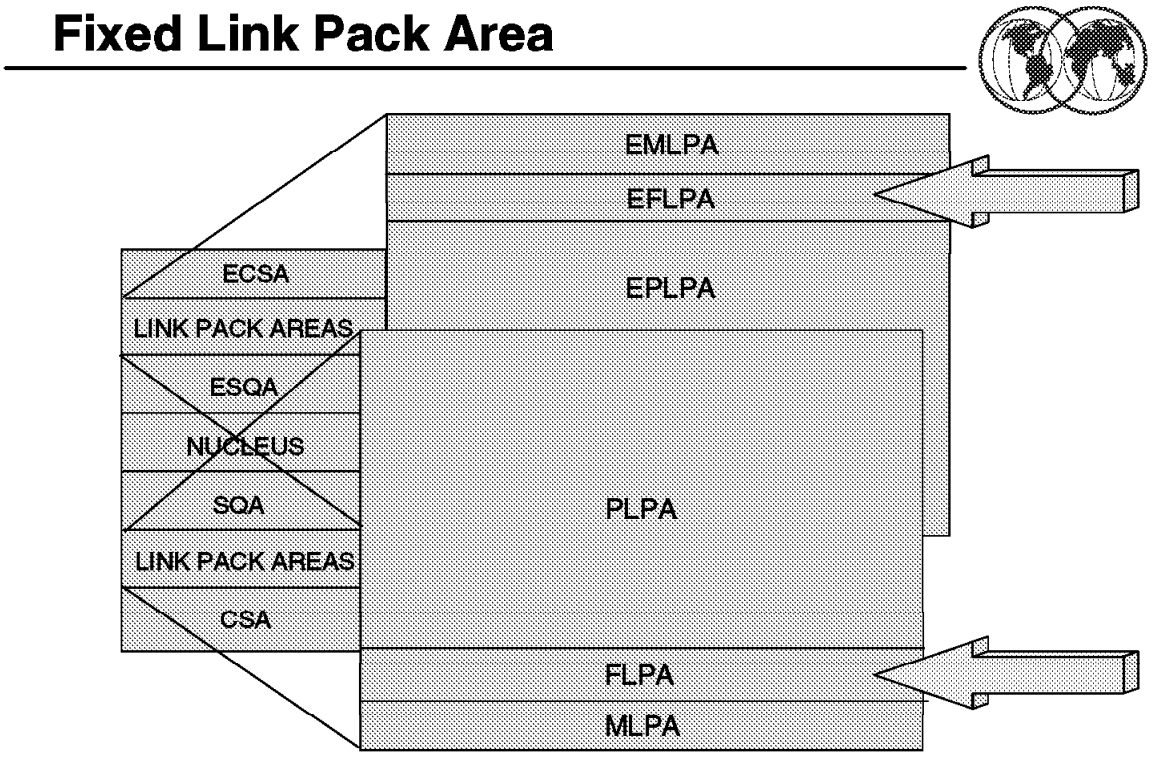


Figure 135. Fixed link pack area

4.1.5 Fixed link pack area (FLPA or extended FLPA)

The FLPA is loaded at IPL time, with those modules listed in the active IEAFIXxx member of SYS1.PARMLIB, and also contains modules which must be kept in fixed storage frames (that is cannot be paged out).

This area should be used only for modules that significantly increase performance when they are fixed rather than pageable. Modules placed in the FLPA must be reentrant and refreshable. The best candidate for the FLPA are modules that are infrequently used but are needed for fast response.

4.1.5.1 Contents of the FLPA or extended FLPA

Modules from the LPALST concatenation, the LNKLST concatenation, SYS1.MIGLIB, and SYS1.SVCLIB can be included in the FLPA. FLPA is selected through specification of the FIX parameter in IEASYSxx which is appended to IEAFIX to form the IEAFIXxx parmlib member, or from the operator′s console at system initialization.

Modules specified in IEAFIXxx are placed in the FLPA or extended FLPA, depending on the RMODE of the modules. Modules with RMODE of 24 are placed in the FLPA, while those with an RMODE of ANY are placed in the extended FLPA.

Coding the IEAFIXxx Member

- ★ Specify - (INCLUDE - LIBRARY - MODULES)
  - ► Consider module order
    - ⋯ Saves search time

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(IEAFIX01) - 01.01         Line 00000000 Col 001 080
Command ===>                                          Scroll ===> PAGE
*************************** Top of Data ********************************
INCLUDE LIBRARY(SYS1.LPALIB)
        MODULES(IEAVAR00,       /*  7K  RCT INIT/TERM     */
                IEAVAR06,       /*  RCT INIT/TERM ALIAS   */
                IGC0001G,       /*  456 RESTORE(SVC17)    */
                ICHRFC00,       /*  RACF IMS/CICS         */
                ICHRFR00,       /*  RACF IMS/CICS         */
                ICHSFR00)       /*  RACF IMS/CICS         */
INCLUDE LIBRARY(SYS1.SVCLIB) MODULES(IGC000RC IGC09302 IGC09303)
************************** Bottom of Data ******************************
```

Figure 136. Coding the IEAFIXxx member

4.1.5.2 Specifying modules in the FLPA or extended FLPA

Use the IEAFIXxx member in the SYS1.PARMLIB to specify the names of the modules that are to be fixed in central storage for the duration of an IPL.

Note: These libraries should be cataloged in the system master catalog.

Modules specified in IEAFIXxx are loaded and fixed in the order which they are specified in the member (to keep search time within reasonable limits, do not allow the FLPA to become excessively large).

Coding IEAFIXxx in SYS1.PARMLIB: The statements or parameters used in IEAFIXxx are:

INCLUDE Specifies modules to be loaded as temporary extensions to the existing pageable link pack area (PLPA).

LIBRARY Specifies a qualified data set name. The specified library must be cataloged in the system master catalog.

MODULES Specifies a list of 1 to 8 character module names.

Rules for specification 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 modules names can be specified.
- 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.

The visual shows an example of IEAFIXxx

4.1.5.3 Advantages of FLPA or extended FLPA

Because fixed modules are not paged, you 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 (PLPA) directory.

Note

The price for this performance improvement is the reduction in the central storage available for paging old jobs and starting new jobs. Remember that pages referenced frequently tend to remain in central storage even when they are not fixed.

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

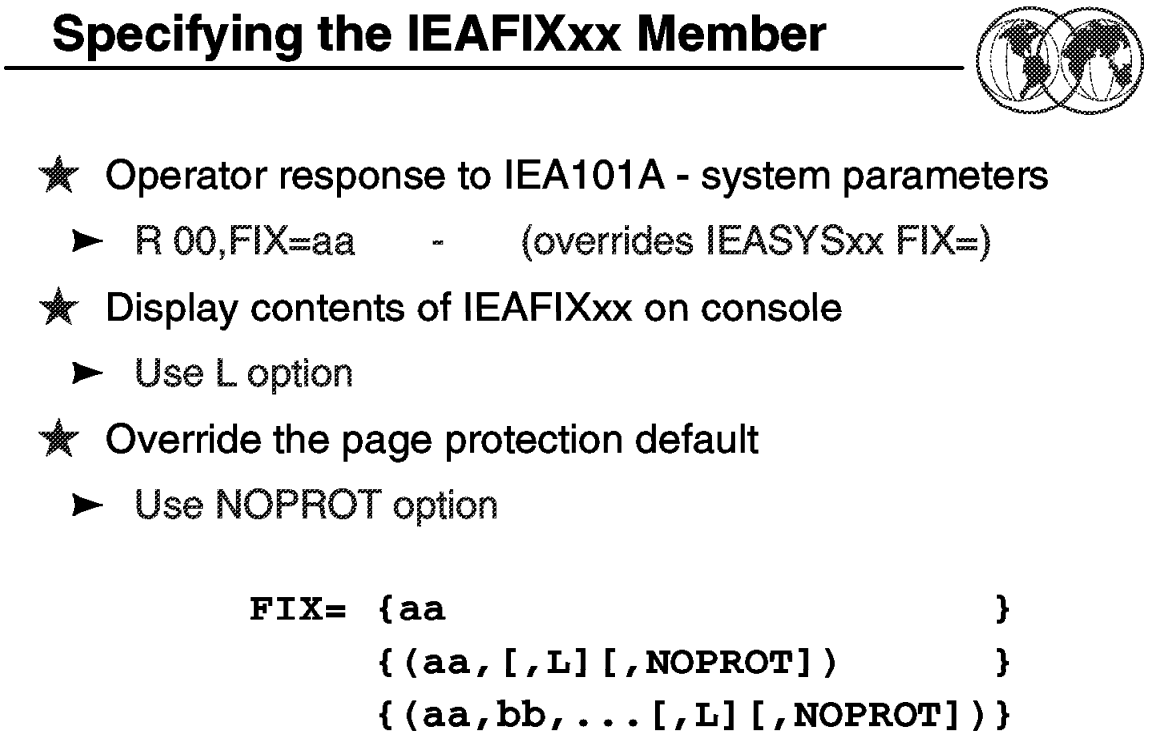


Figure 137. Specifying the IEAFIXxx member

4.1.5.4 How to specify FIX parameter during system initialization

The two characters (A through Z, 0 through 9, @, #, or $) represented by aa (or bb and so forth), are appended to IEAFIX to form the name of the IEAFIXxx parmlib members. The options are:

L 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.

NOPROT By default, the LPA modules in the IEAFIXxx parmlib members are page-protected in storage. However, the NOPROT option allows an installation to override the page protection default.

Syntax format for FIX parameter

```
FIX= {aa                    }
     {(aa,[,L][,NOPROT])    }
     {(aa,bb,...[,L][,NOPROT])}
```

An example of overriding the value of the FIX parameter in the IEASYSxx during system initialization is shown in Figure 138 on page 205.

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR OS/390 02.05.00 HBB6605
R 00,FIX=01 1
IEE600I REPLY TO 00 IS;FIX=01
```

Figure 138. Example of overriding the FIX parameter value

1 IEAFIX01 was selected during system initialization.

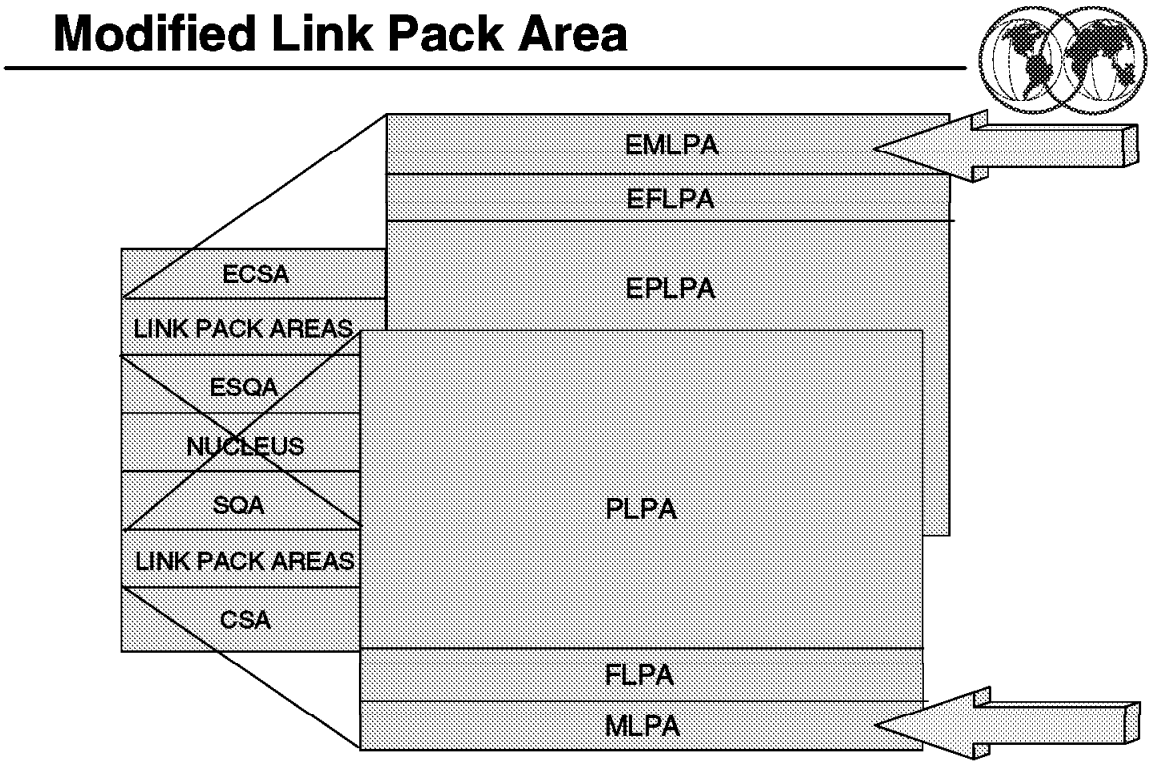


Figure 139. Modified link pack area

4.1.6 Modified link pack area (MLPA/extended MLPA)

This area may be used to contain reenterable routines from APF-authorized libraries that are to be part of the pageable extension to the link pack area during the current IPL.

The MLPA exists only for the duration of the IPL. Therefore, if an MLPA is desired, the modules in the MLPA must be specified for each IPL (including quick start and warm start IPLs).

4.1.6.1 Contents of the MLPA or extended MLPA

MLPA contains modules listed in the active IEALPAxx member of SYS1.PARMLIB, through the specification of the MLPA parameter in the IEASYSxx or from the operator's console at system initialization.

LPA modules specified in the IEALPAxx are placed in the MLPA or 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.

Coding the IEALPAxx Member

★ Specify - (INCLUDE - LIBRARY - MODULES)

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(LEALPA02) - 01.01         Line 00000000 Col 001 080
Command ===>                                              Scroll ===> PAGE
************************** Top of Data ********************************
INCLUDE LIBRARY(ISP.SISPLOAD)
        MODULES(ISPASUBS,ISPEX,ISPEXEC,
                ISPLINK,ISPLNK,
                ISPQRY)
INCLUDE LIBRARY(SYS1.LINKLIB)
        MODULES(IFBSEXIT)
************************** Bottom of Data *****************************
```

Figure 140. Coding the IEALPAxx member

4.1.6.2 How to code IEALPAxx of the SYS1.PARMLIB

The visual shows an example of the IEALPAxx member.

The important syntax rules for IEALPAxx are:

- 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.
- 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.

Specifying the IEALPAxx Member

- Operator response to IEA101A - system parameters
  - R 00,MLPA=02 - (overrides IEASYSxx MLPA=)
- Display contents of IEALPAxx on console
  - Use L option
- Override the page protection default
  - Use NOPROT option

```
MLPA= {aa                          }
      {(aa,[,L][,NOPROT])          }
      {(aa,bb,...[,L][,NOPROT])}
```

Figure 141. Specifying the IEALPAxx member

4.1.6.3 Specifying the MLPA member at system initialization

The two characters (A through Z, 0 through 9, @, #, or $) represented by aa (or bb and so forth), are appended to IEALPA to form the name of the IEALPAxx parmlib members. The options are:

L 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.

NOPROT By default, the LPA modules in the IEALPAxx parmlib members are page-protected in storage. However, the NOPROT option allows an installation to override the page protection default.

Syntax format for MLPA parameter

```
MLPA= {aa                    }
      {(aa[,L][,NOPROT])     }
      {(aa,bb,...[,L][,NOPROT])}
```

An example of overriding the value of the MLPA parameter in the IEASYSxx during system initialization is shown in Figure 142.

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR OS/390 02.05.00 HBB6605
R 00,MLPA=02 1
IEE600I REPLY TO 00 IS;MLPA=02
```

Figure 142. Example of overriding the MLPA parameter value

1 IEALPA02 was selected during system initialization.

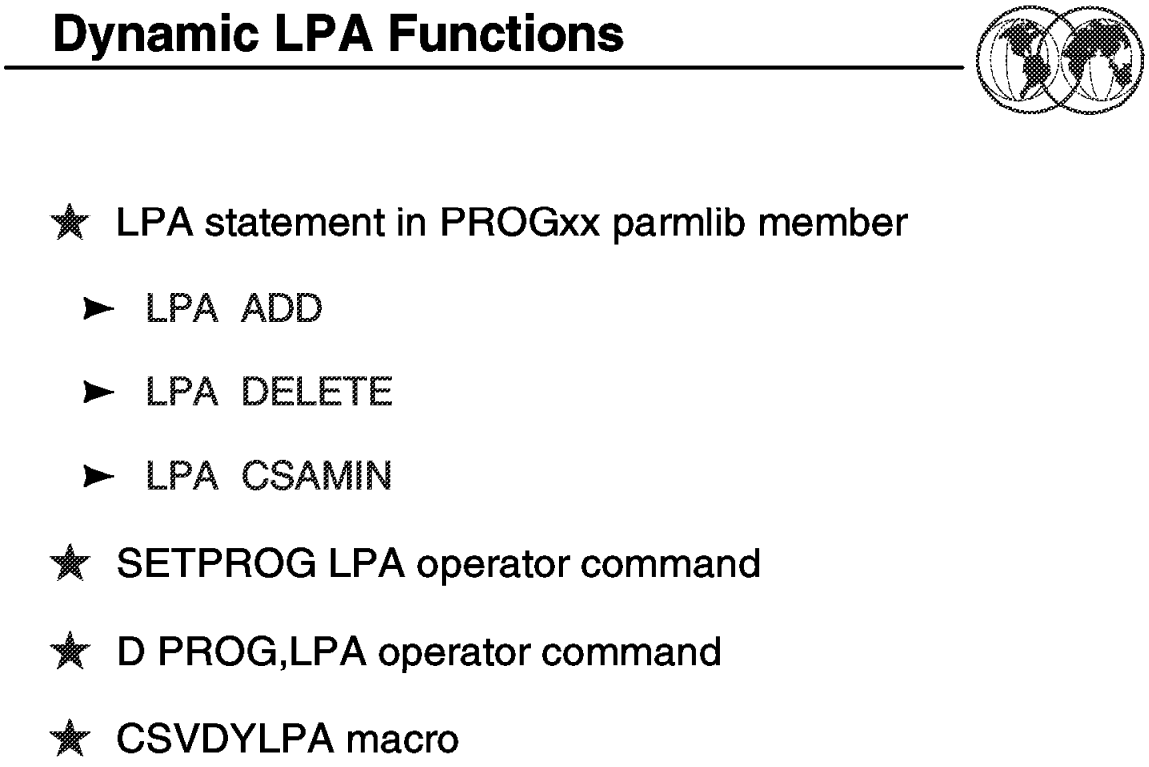


Figure 143. Managing dynamic LPA content

4.1.7 Managing dynamic LPA content

Dynamic link pack area (DLPA) was introduced in OS/390 Release 4, which has the ability to dynamically add or delete modules from the link pack area (LPA) after the IPL. This facility allows optional and new products to be loaded into the system without an IPL. It also enables you to display modules residing in LPA.

A module added dynamically will be found before one of the same name added during the IPL. This allows you to test new modules before you put them into production. Modules added dynamically to the LPA will be loaded into the common area (CSA) or extended common area (ECSA).

4.1.7.1 How to perform dynamic LPA functions

The dynamic LPA functions may be invoked through one of the following functions:

- The PROGxx parmlib member includes the LPA statements, which are used to define what modules can be added to or deleted from LPA after the IPL. You use the SET command in order to validate the PROGxx parmlib member; for example, SET PROG=xx.
- The SETPROG LPA command may be used to initiate a change (add or delete) to the LPA.
- The DISPLAY PROG,LPA command may be used to display information about modules that have been added to LPA.
- The CSVDYLPA macro allows an authorized program to initiate dynamic LPA services.

  For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

4.2 The linkList (LNKLST)

This topic explains the LNKLST and the related mechanisms utilized to improve the performance of locating and fetching modules. It is important to understand the relationship between the LNKLST and LLA/VLF.

When determining which data sets LLA is to manage, try to limit these choices to production load libraries that are rarely changed. Because LLA manages LNKLST libraries by default, you need only determine which non-LNKLST libraries LLA is to manage. If, for some reason, you do not want LLA to manage particular LNKLST libraries, you must explicitly remove such libraries from LLA management.

Because you obtain the most benefit from LLA when you have both LLA and VLF functioning, you should plan to use both. When used with VLF, LLA reduces the I/O required to fetch modules from DASD by causing selected modules to be staged in VLF data spaces. LLA does not use VLF to manage library directories. When used without VLF, LLA eliminates only the I/O the system would use in searching library directories on DASD.

All LLA-managed libraries must be cataloged. This includes LNKLST libraries. A library must remain cataloged for the entire time it is managed by LLA.

The benefits of LLA apply only to modules that are retrieved through the following macros: LINK, LINKX, LOAD, ATTACH, ATTACHX, XCTL, and XCTLX.

LLA does not stage load modules in overlay format. LLA manages the directory entries of overlay format modules, but the modules themselves are provided through program fetch. If you want to make overlay format modules eligible for staging, you must re-linkedit the modules as non-overlay format. These reformatted modules might occupy more storage when they execute and, if LLA does not stage them, might take longer to be fetched.

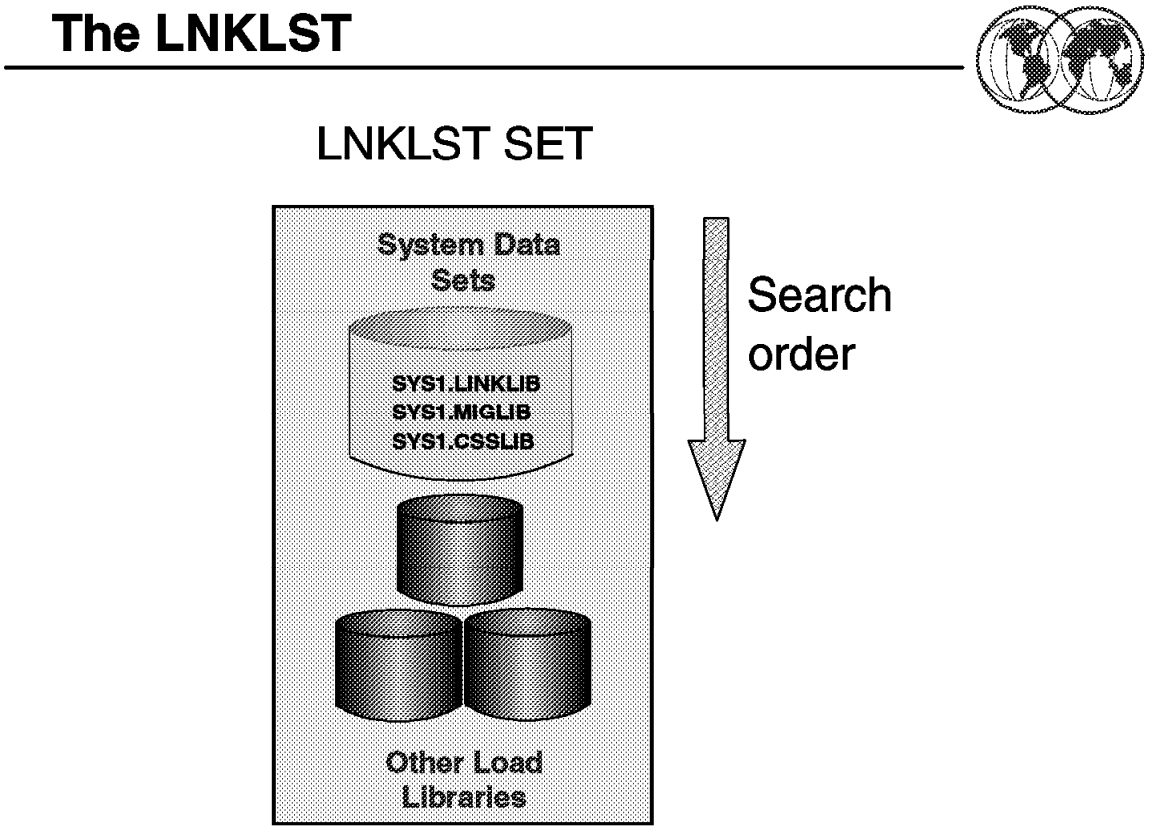


Figure 144. The LNKLST

4.2.1 An overview of linkList (LNKLST)

The LNKLST is a group of system and user-defined load libraries, which form part of the search order the system uses for programs.

Modules (programs), whether stored as load modules or programs objects, must be loaded into both virtual storage and central storage before they can be run.

By default, the LNKLST begins with these system data sets:

- SYS1.LINKLIB
- SYS1.MIGLIB
- SYS1.CSSLIB

You can change this order and add other load libraries to the LNKLST concatenation.

4.2.1.1 The LNKLST concatenation

The LNKLST concatenation is established at IPL time. It consists of SYS1.LINKLIB, followed by the libraries specified in the LNKLSTxx member(s) of SYS1.PARMLIB. The LNKLSTxx member is selected through the LNK parameter in the IEASYSxx member of the SYS1.PARMLIB. In addition, the system also automatically concatenates data sets SYS1.MIGLIB and SYS1.CSSLIB to SYS1.LINKLIB.

The building of the LNKLST concatenation happens during an early stage in the IPL process, before

any user catalogs are accessible, so only those data sets whose catalog entries are in the system master catalog may be included in the linklist. However, to include user cataloged data set in the LNKLST concatenation, you have to specify both the name of the data set and the volume serial number (VOLSER) of the DASD volume on which the data set resides.

Note: The number of data sets that 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. When the limit has been exceeded, the system writes error message IEA328E to the operator's console.

These data sets are concatenated in the order in which they appear in the LNKLSTxx member(s), and the system creates a data extent block (DEB) that describes the data sets concatenated to SYS1.LINKLIB and their extents. This contains details of each physical extent allocated to the linklist. These extents remains in effect for the duration of the IPL. After this processing completes, the library lookaside (LLA) is started and manages the LNKLST data sets and can be used to control updates to them.

4.2.1.2 Specifying the LNK parameter

An example of overriding the value of the LNK parameter in the IEASYSxx during system initialization is shown in Figure 145.

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR OS/390 02.05.00 HBB6605
R 00,LNK=03 1
IEE600I REPLY TO 00 IS;LNK=03
```

Figure 145. Example of overriding the LNK parameter value

1 LNKLST03 was selected during system initialization.

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752, and *OS/390 MVS System Commands*, GC28-1781.

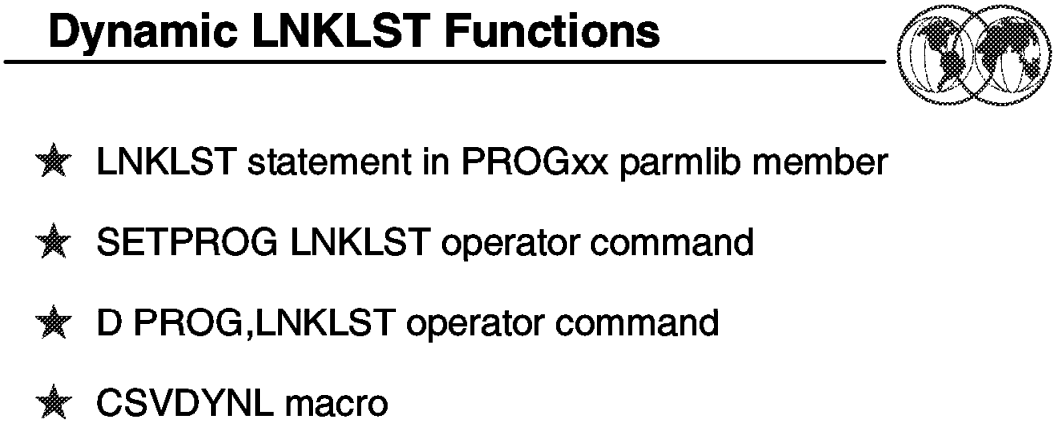


Figure 146. Dynamic LNKLST functions

4.2.2 Managing dynamic LNKLST content

You can update the LNKLST content dynamically using the following methods:

- You can create a new PROGxx parmlib member with the new changes to the LNKLST set, then issue the SET PROG=xx operator command to activate the changes.
- You can simply use the SETPROG LNKLST operator command to update the LNKLST directly.
- You can also use the D PROG,LNKLST command to display information about the LNKLST set.
- Finally, you can use CSVDYNL macro programming service in an authorized program to change the LNKLST concatenation for associated jobs and address spaces.

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

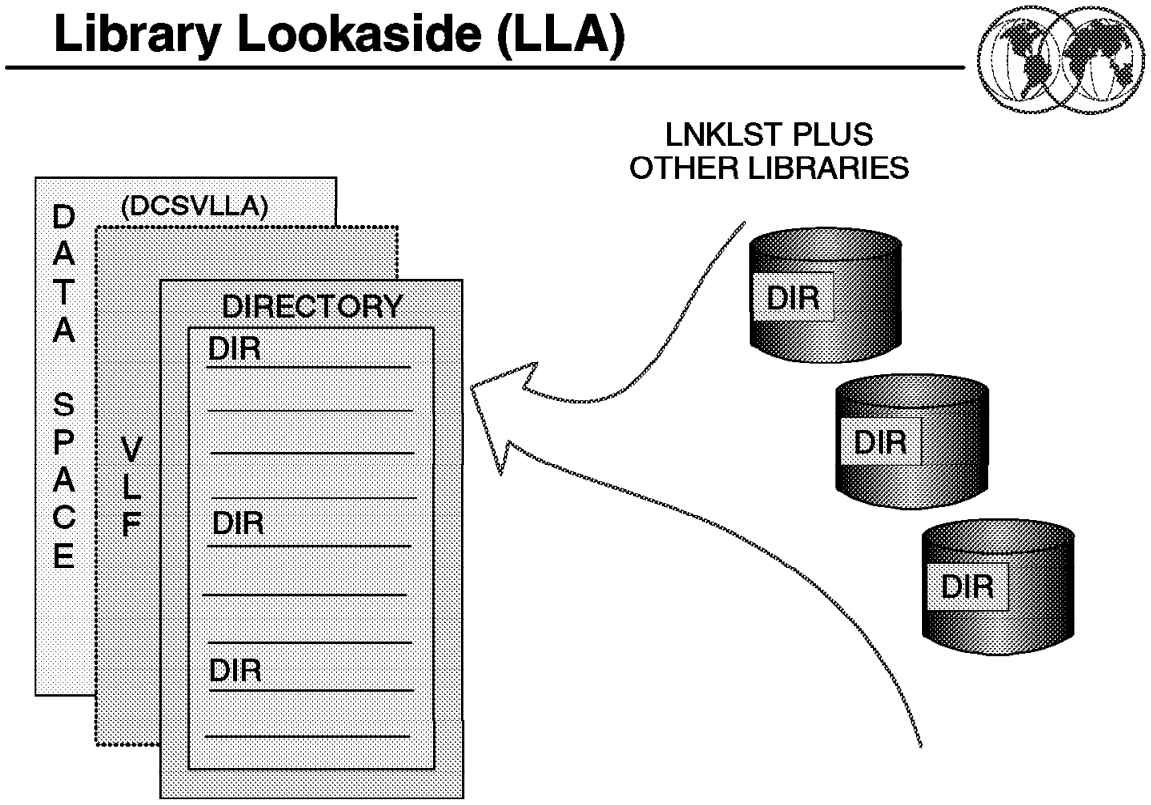


Figure 147. Library lookaside

4.2.3 The library lookaside (LLA) overview

Library lookaside (LLA) is an address space that maintain a copy of the directory entries of the libraries that it manages. Since the entries are cached, the system does not need to read the data set directory entries to find out where a module is stored before fetching it from DASD. This greatly reduces I/O operations.

The main purpose of using LLA is to improve the performance of module fetching on your system.

4.2.3.1 How LLA improves performance

LLA improves the module fetch performance in the following ways:

1. By maintaining (in the LLA address space) copies of the library directories the system uses to locate load modules. The system can quickly search the LLA copy of a directory in virtual storage instead of using costly I/O to search the directories on DASD.
2. By placing (or staging) copies of selected modules in a virtual lookaside facility (VLF) data space DCSVLLA when you define the LLA class to VLF, and start VLF. The system can quickly fetch modules from virtual storage, rather than using slower I/O to fetch the modules from the DASD.
3. By determining which modules, if staged, would provide the most benefit to module fetch performance. LLA evaluates modules as candidates for staging based on statistics it collects about the members of the libraries it manages (such as module size, frequency of fetches per module (fetch count), and the time required to fetch a particular module). If necessary, you can directly influence LLA staging decisions through installation exit routines (CSVLLIX1 and CSVLLIX2).

4.2.3.2 Planning LLA use

Before you can use LLA, you have to do the following:

1. Determine which libraries should be LLA-managed libraries.
2. Code the CSVLLAxx parmlib member to include, modify, and remove the LLA-managed libraries as well as how to optimize the performance of the search processing and selectively refresh LLA directory entries.
3. Learn how to control the LLA through the operator commands Start, Stop, and Modify LLA. LLA directory entries may be selectively refreshed via the operator command F LLA,REFRESH.

4.2.3.3 LLA-managed libraries

By default, library lookaside (LLA) address space caches directory entries for all the modules in the data sets included in the linklist (LNKLST) concatenation (defined in LNKLSTxx or PROGxx parmlib member).

You can also identify other user-defined load libraries that contain frequently used modules to be managed by LLA.

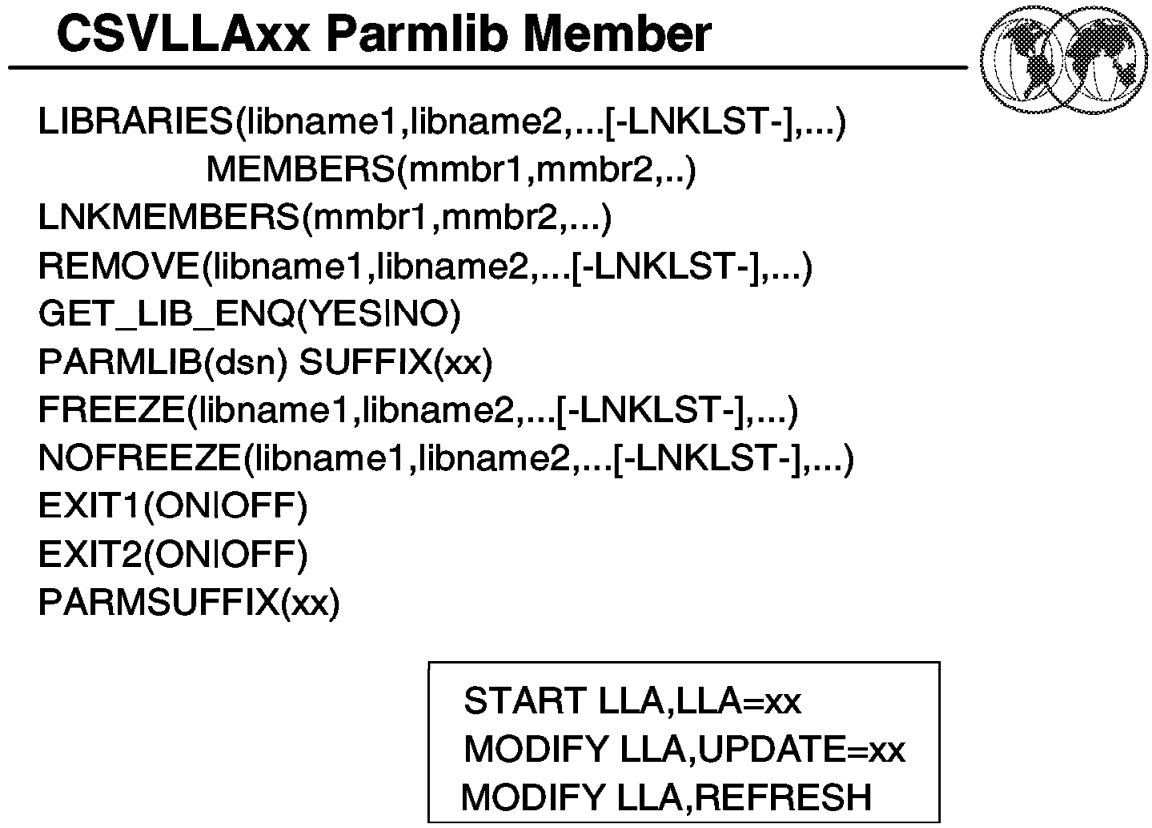


Figure 148. Dynamic LNKLST functions

4.2.3.4 CSVLLAxx SYS1.PARMLIB member

You use the CSVLLAxx parmlib member to specify which libraries (in addition to the LNKLST concatenation) library lookaside (LLA) is to manage.

Note: If you do not supply a CSVLLAxx member, LLA will by default manage only those libraries defined in the LNKLST concatenation.

You can also use CSVLLAxx to specify:

- 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.
- Additional CSVLLAxx members to be used to control LLA processing. These members can reside in data sets other than SYS1.PARMLIB.
- Whether exit routines are to be called during LLA processing.

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

4.2.3.5 How to start and stop LLA

To start LLA. you use the START LLA,LLA=xx command. This command identifies the CSVLLAxx parmlib member to build the LLA directory. This command is issued during system initialization by the IBM-supplied IEACMD00 parmlib member; the command can also be entered by the operator.

The parameters for the S LLA command are:

LLA Invokes the LLA procedure and creates the LLA address space.

LLA=xx Indicates which CSVLLAnn parmlib member LLA is to use. If you omit LLA=xx, LLA will build its directory using only the LNKLST libraries.

SUB=MSTR Indicates that the name of the subsystem that will process the task is the master subsystem. If you omit this parameter, the JES subsystem starts LLA. The resulting dependency on JES requires LLA to be stopped when stopping JES.

Note: We recommend that you specify SUB=MSTR on the START LLA command to prevent LLA from failing if JES fails.

Syntax format for S LLA command

```
S LLA[,SUB=MSTR][,LLA=xx]
```

To stop LLA, you use the P LLA operator command.

The parameter for the P LLA command is:

LLA The job name assigned to the LLA address space.

Syntax for P LLA command

```
P LLA
```

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752, and *OS/390 MVS System Commands*, GC28-1781.

4.2.3.6 How to modify LLA

You can use the MODIFY LLA command to change LLA dynamically, in one of the following ways:

- MODIFY LLA,REFRESH

  This command rebuilds LLA's directory for the entire set of libraries managed by LLA. This action is often called the complete refresh of LLA.

- MODIFY LLA,UPDATE=xx

  This command rebuilds LLA's directory only for specified libraries or modules. xx identifies the CSVLLAxx member that contains the names of the libraries for which directory information is to be refreshed. This action is often called a selective refresh of LLA.

Note

There are several situations where you need to refresh the LLA with the latest version of the directory information from the DASD:

1. Whenever you update a load module in a library that LLA manages, make sure you follow the update by issuing the appropriate form of the MODIFY LLA command to refresh LLA′s cache with the latest version of the directory information from the DASD. Otherwise, the system will continue to use an older version of a load module.
2. If you accidentally delete a data set from the current LNKLST set, the system will continue working, and continue finding modules in the deleted library! This is because the physical addresses of the members are still held by LLA even though it is no longer possible to open the directory. This only works, of course, until the physical space on the DASD is reused by something else.
3. You will also find yourself in the situation where you need to compress a data set from the LNKLST set. You should refresh LLA′s cache after the compress, otherwise, the directory entries in the LLA for every module moved during the compress process would be invalid, leading to abends whenever a user attempted to load one of these, until a refresh is done.

To do a complete update of the LLA:

```
F LLA,REFRESH
CSV210I LIBRARY LOOKASIDE REFRESHED
```

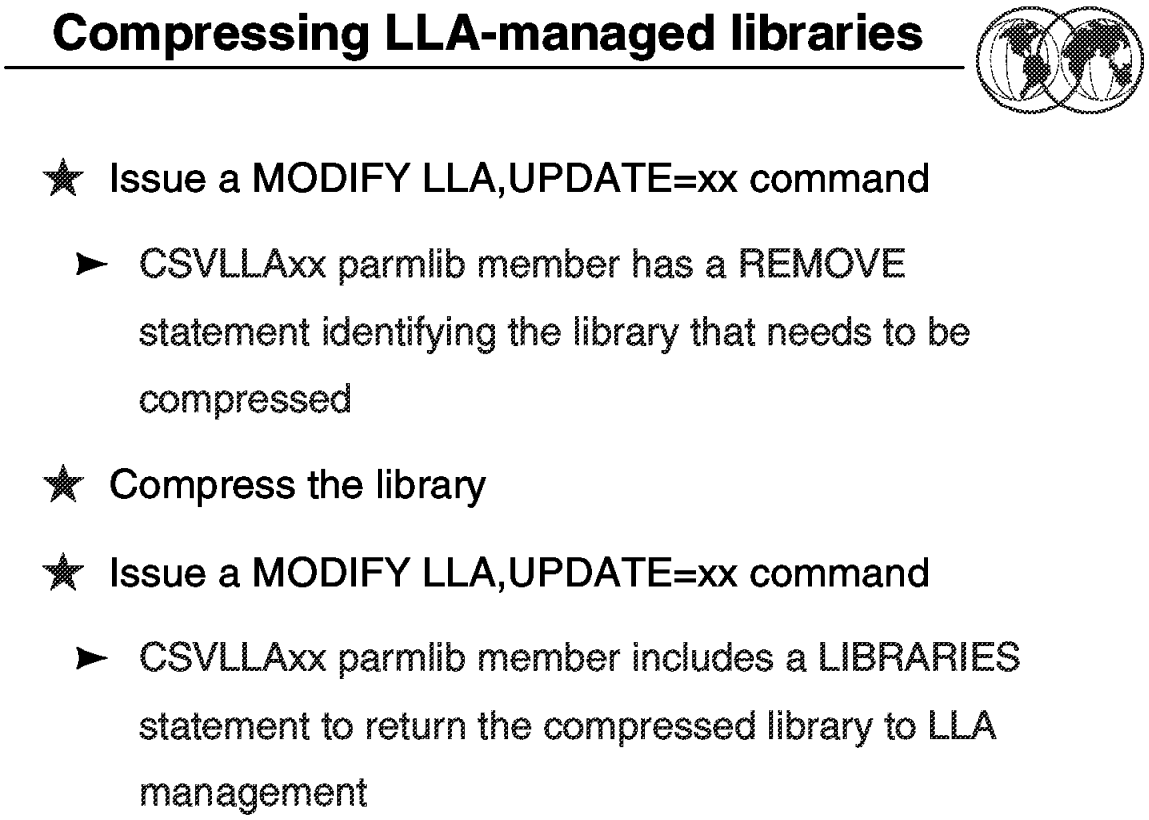


Figure 149. Compressing LLA-managed libraries

4.2.3.7 How to compress LLA-managed libraries

The procedure for compressing an LLA-managed library is:

1. Create a new CSVLLAxx member that includes a REMOVE statement identifying the library that needs to be compressed.

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(CSVLLA02)  - 01.01        Line 00000000 Col 001 080
************************** Top of Data ********************************
REMOVE(SYS1.LINKLIB)
************************** Bottom of Data *****************************
```

   Then issue the F LLA,UPDATE=xx command:

```
F LLA,UPDATE=02
IEE252I MEMBER CSVLLA02 FOUND IN SYS1.PARMLIB
CSV210I LIBRARY LOOKASIDE UPDATED
```

2. Compress the library.

   There are two ways of compressing a data set:

   - You can issue the line command Z against the data set you want to compress from the ISPF panel.

```
Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------
DSLIST - Data Sets Matching SYS1.LINKLIB                    Row 1 of 1
Command ===>                                              Scroll ===> PAGE

Command - Enter "/" to select action           Message          Volume
------------------------------------------------------------------------
Z        SYS1.LINKLIB                                           MPRES1
********************** End of Data Set list ****************************
```

- Or you can submit a job to compress the data set using the utility IEBCOPY as shown in the following sample JCL.

```
//COMPRES JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//*****************************************
//*                                       *
//*    COMPRESSING DATA SETS USING IEBCOPY  *
//*                                       *
//*****************************************
//STEP1  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//INPUT    DD DSNAME=SYS1.LINKLIB,DISP=SHR
//OUTPUT   DD DSNAME=SYS1.LINKLIB,DISP=SHR
//SYSIN    DD *
    COPY INDD=INPUT,OUTDD=OUTPUT
/*
```

3. Create a new CSVLLAxx member that includes a LIBRARIES statement to return the compressed library to LLA management.

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(CSVLLA03)  - 01.01        Line 00000000 Col 001 080
************************** Top of Data ********************************
LIBRARIES(SYS1.LINKLIB)
************************** Bottom of Data *****************************
```

Then issue the F LLA,UPDATE=xx command:

```
F LLA,UPDATE=03
IEE252I MEMBER CSVLLA03 FOUND IN SYS1.PARMLIB
CSV210I LIBRARY LOOKASIDE UPDATED
```

Virtual Lookaside Facility (VLF)

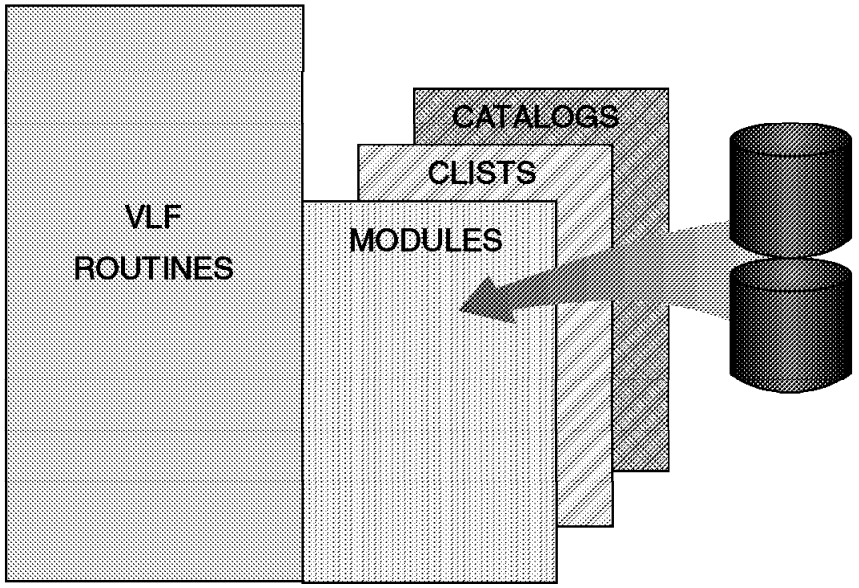


Figure 150. Virtual lookaside facility

4.2.4 Virtual lookaside facility (VLF) overview

The virtual lookaside facility (VLF) is a set of services that can improve the response time of applications that must retrieve a set of data for many users. VLF creates and manages data spaces to store an application's most frequently used data. When the application makes a request for data, VLF checks its data space to see if the data is there. If the data is present, VLF can rapidly retrieve it without requesting I/O to DASD.

To take advantage of VLF, an application must identify the data it needs to perform its task. The data is known as a data object. Data objects should be small-to-moderate in size, named according to the VLF naming convention discussed later in this section, and associated with an installation-defined class of data objects.

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 spaces, there are performance considerations that each installation must weigh when planning for the resources required by VLF.

Note: VLF is intended for use with major applications. Because VLF runs as a started task that the operator can stop or cancel, it cannot take the place of any existing means of accessing data on DASD. Any application that uses VLF must also be able to run without it.

4.2.4.1 Using VLF with LLA

You will obtain the most benefit from LLA when you have both LLA and VLF functioning, you should plan to use both. When used with VLF, LLA reduces the I/O required to fetch modules from the DASD by causing selected modules to be staged in VLF data spaces.

LLA does not use VLF to manage library directories. When using without VLF, LLA eliminates only the I/O that the system would use in searching for library directories on DASD.

4.2.4.2 VLF considerations

Before you can take full advantage of implementing VLF to improve your system performance, there are several factors you have to consider; some of these factors are:

- VLF works best with two kinds of data:
  - Data objects that are members of partitioned data sets, located through a partitioned data set (PDS) concatenation.
  - Data objects that, while not PDS members, could be easily described as a collection of named objects that are repeatedly retrieved by many users.

  If neither description fits your data objects, it is likely that you would not obtain any performance benefit from VLF. Remember, there are storage overhead costs associated with using VLF.
- Like data in private storage, data stored through VLF is subject to page stealing. That is, VLF works best when a significant portion of the data is likely to remain in processor storage and not be paged out to auxiliary storage.
- VLF works best with relatively small objects because less virtual storage is expended to reduce the number of I/O operations.
- VLF is designed to improve performance by increasing the use of virtual storage to reduce the number of I/O operations. For a system that is already experiencing a central, expanded, or auxiliary storage constraint, this strategy is probably not a good choice.

4.2.4.3 VLF planning

Before you can use VLF, you have to:

1. Start VLF using the IBM-supplied default COFVLFxx parmlib member.
2. Update COFVLFxx to include the VLF classes associated with the applications or products.

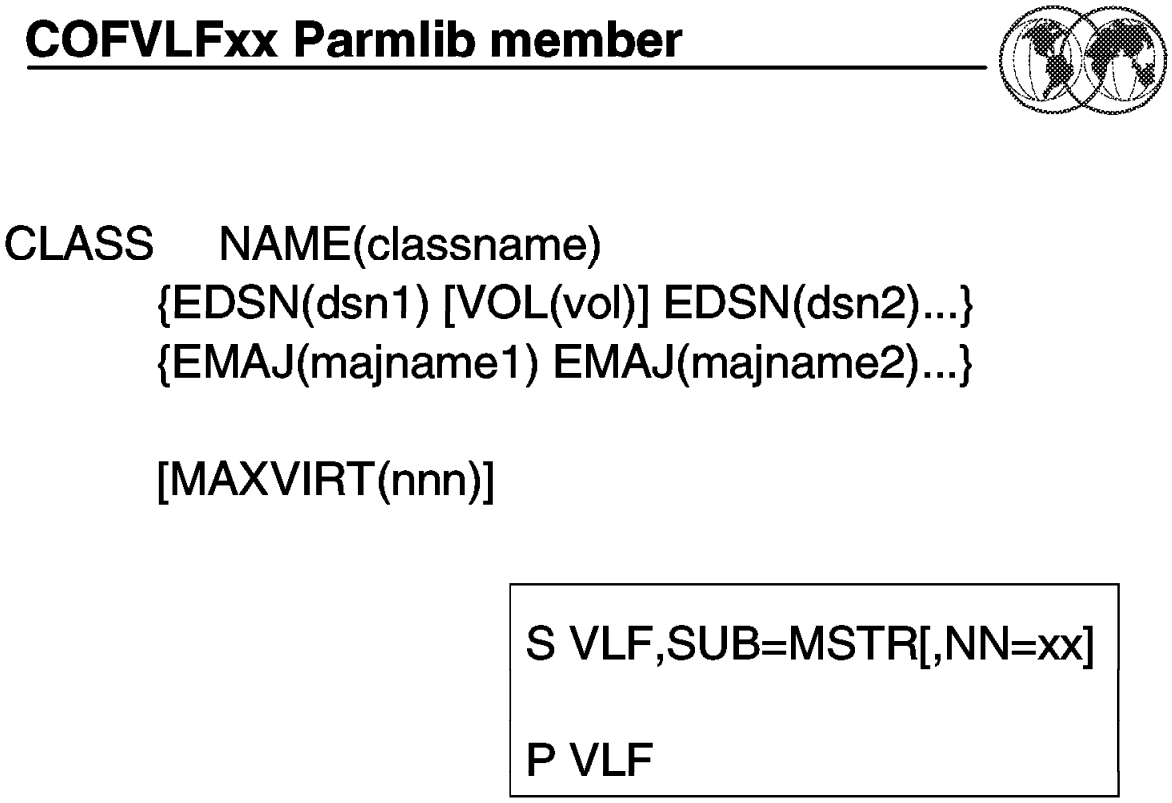


Figure 151. COFVLFxx parmlib member

4.2.5 COFVLFxx parmlib member

You use the COFVLFxx of the SYS1.PARMLIB to define classes of VLF objects. 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).

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

Some of the more important statement and parameters for COFVLFxx are:

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 1 to 7 alphanumeric characters including @, #, and $. IBM-supplied VLF class names begin with the letters A through I, for example, NAME(CSVLLA).

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 is not included in the eligible data set name list. The system issues an informational message to the operator to identify any data set where the system cannot find the catalog entry to extract the volume.

Multiple occurrences of the same data set name with a different volume 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.

EMAJ(majname) EMAJ identifies an eligible major name (majname) for a non-PDS class, a class that does not have a major name and minor name 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).

Note: Do not use the EMAJ and the EDSN parameter on the same CLASS statement.

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

Figure 152 shows an example of COFVLFxx:

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(COFVLF04)  - 01.01        Line 00000000 Col 001 080
*************************** Top of Data ********************************
CLASS NAME(CSVLLA)              /* CLASS NAME FOR LIBRARY LOOKASIDE  */
      EMAJ(LLA)                 /* MAJOR NAME FOR LIBRARY LOOKASIDE  */
                                /* Default MAXVIRT = 4096 4K blocks  */
                                /*                 = 16Mb            */
/*                                                                    */
CLASS NAME(IKJEXEC)         /* TSO/E VERSION 2 CLIST/REXX INTERFACE  */
      EDSN(SYS1.OS390.CLIST)
      EDSN(MVSTOOLS.SHARED.CLIST)
      MAXVIRT(1024)         /* MAXVIRT = 512 4K blocks               */
                            /*         = 4Mb                         */
/*                                                                    */
CLASS NAME(IGGCAS)          /* MVS/DFP 3.1 CATALOG in Data space     */
      EMAJ(CATALOG.SC54.MCAT)
      EMAJ(CATALOG.TOTICF1.VITS001)
      MAXVIRT(1024)         /* MAXVIRT = 512 4K blocks               */
                            /*         = 2Mb (minimum value)         */
*************************** Bottom of Data *****************************
```

Figure 152. An example of COFVLFxx

4.2.5.1 Starting and stopping VLF

To start VLF, you use the START VLF,SUB=MSTR,N=xx command. This command identifies the COFVLFxx parmlib member to build VLF. This command is issued by the IBM-supplied COMMND00 parmlib member during system initialization; the command can also be entered by the operator.

The parameters for the S VLF command are:

S VLF,SUB=MSTR[,NN=xx]

Invokes the VLF procedure that starts the virtual lookaside facility (VLF).

NN=xx Indicates that the system is to start VLF using the COFVLFxx member of the SYS1.PARMLIB (default COFVLF00).

Syntax for S VLF command

```
S VLF,SUB=MSTR[,NN=xx]
```

To stop VLF, you use the P VLF command.

The parameter for the P VLF command is:

VLF The jobname assigned to the virtual lookaside facility (VLF). Using this parameter stops VLF with the message COF033I.

Note: Stopping VLF can degrade your system performance.

Syntax for P VLF command

```
P VLF
```

4.3 The authorized libraries

OS/390 offers a mechanism called the authorized program facility (APF) to restrict the access to sensitive system functions or user programs. APF was designed to avoid integrity exposures. The installation identifies what libraries contain those special functions or programs. Those libraries are then called APF (authorized program facility) libraries.

The modified link pack area (MLPA) may be used to contain reenterable routines from APF-authorized libraries that are to be part of the pageable extension to the link pack area during the current IPL. The MLPA exists only for the duration of an IPL. Therefore, if an MLPA is desired, the modules in the MLPA must be specified for each IPL (including quick start and warm start IPLs).

The MLPA is allocated just below the FLPA (or the PLPA, if there is no FLPA); the extended MLPA is allocated above the extended FLPA (or the extended PLPA if there is no extended FLPA). When the system searches for a routine, the MLPA is searched before the PLPA.

Note: Loading a large number of modules in the MLPA can increase fetch time for modules that are not loaded in the LPA. This could affect system performance.

The MLPA can be used at IPL time to temporarily modify or update the PLPA with new or replacement modules. No actual modification is made to the quick start PLPA stored in the system′s paging data sets. The MLPA is read-only, unless NOPROT is specified on the MLPA system parameter.

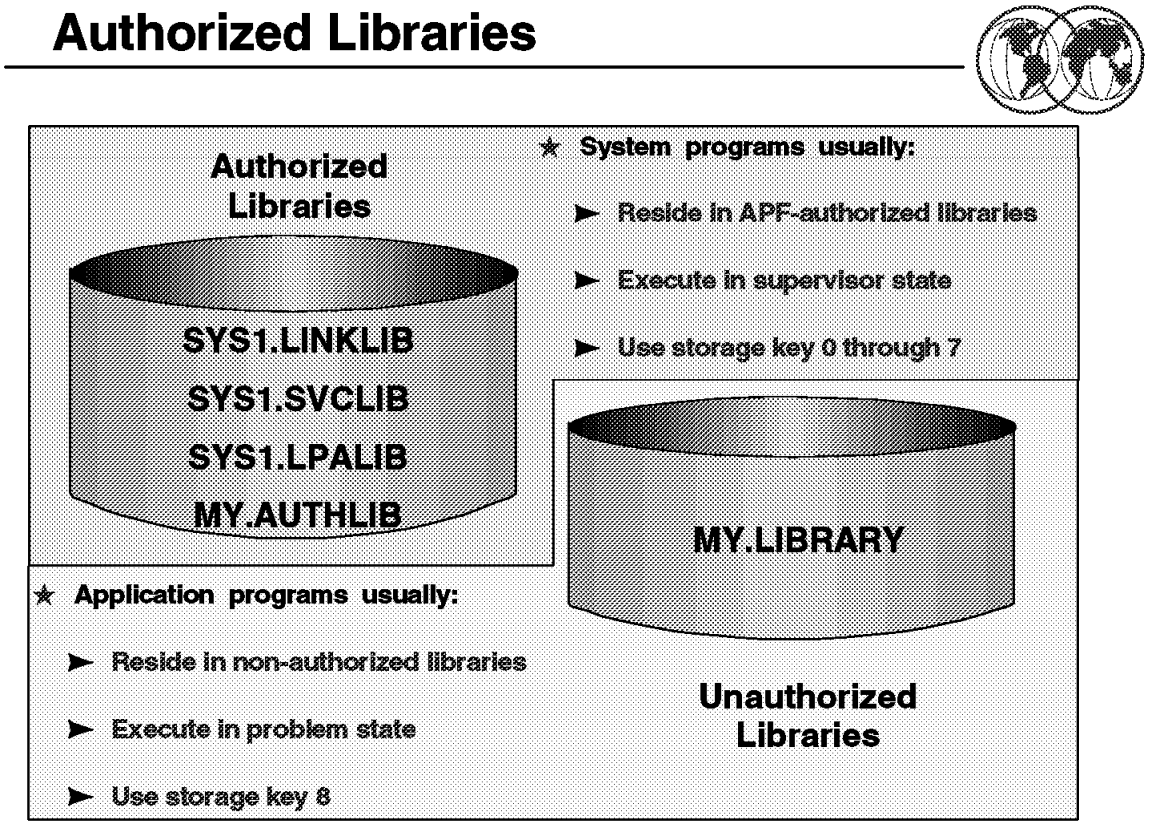


Figure 153. The authorized libraries

4.3.1 Authorized libraries overview

Many system functions, such as entire supervisor calls (SVC) or special paths through SVCs, are sensitive. Access to these functions must be restricted to only authorized programs to avoid compromising the security and integrity of the system.

The system considers a program authorized if the program has one or more of the following characteristics:

- Runs in supervisor state (bit 15 of the PSW is zero).
- Runs with PSW key 0 to 7 (bits 8 through 11 of the PSW are in the range 0 to 7).
- Runs under an APF-authorized job step task

The authorized program facility (APF) allows your installation to identify system or user programs that can use sensitive system functions.

Libraries that contain authorized programs are known as authorized libraries. APF-authorized programs must reside in one of the following authorized libraries:

- SYS1.LINKLIB
- SYS1.SVCLIB
- SYS1.LPALIB
- An authorized library specified by your installation

Note

1. By default, the libraries in the LNKLST concatenation are considered authorized unless you specified LNKAUTH=APFTAB in the IEASYSxx parameter list. However, if the system accesses the libraries in the LNKLST concatenation through JOBLIB or STEPLIB DD statements, the system does not consider those libraries authorized unless you specified the library name in the APF list by using either the IEAAPFxx or the PROGxx parmlib member.
2. If a JCL DD statement concatenates an authorized library in any order with an unauthorized library, the entire set of concatenated libraries is treated as unauthorized.
3. SYS1.LPALIB is treated as an authorized library only while the system builds the pageable link pack area (PLPA) using the LPALSTxx parmlib member. All modules in PLPA are marked as coming from an authorized library. When accessed through a STEPLIB, JOBLIB, or TASKLIB DD statement, SYS1.LPALIB is considered an authorized library only if you have included it in the APF list.

4.3.1.1 Authorized program facility (APF)

You can use the authorized program facility (APF) to identify system or user programs that can use sensitive system functions; for example APF:

- Restricts the use of sensitive system SVC routines (and sensitive user SVC routines, if you need them) to APF-authorized programs.
- Allows the system to fetch all modules in an authorized job step task only from authorized libraries, to prevent programs from counterfeiting a module in the module flow of an authorized job step task.

4.3.1.2 Authorizing a program

To authorize a program, you must first assign the authorized code to the first load module of the program. APF prevents authorized programs from accessing any load module that is not in an authorized library. When the system attaches the first load module of a program, the system considers the program APF-authorized if the module meets both the following criteria:

1. The module is contained in an authorized library.
2. The module is link-edited with authorized code, AC=1 (to indicate that you want to authorize the job step task). This code is contained in a bit setting in the partitioned data set (PDS) directory entry for the module.

4.3.1.3 Link-editing modules with AC=1

You can use the PARM field on the link-edit step to assign the APF-authorization code to a load module. To assign an authorization code using JCL, code AC=1 in the operand field of the PARM parameter of the EXEC statement.

```
//LKED   EXEC  PGM=HEWL,PARM='AC=1',...
```

This method causes the system to consider every load module created by the step to be authorized.

The authorization code of a load module has meaning only when it resides on an APF-authorized data set and when the load module executes as the first program of a job-step attach. If no authorization code is assigned in the linkage editor step, the system assigns the default of authorized. Thus, if a load module tries to execute functions or SVCs that requires authorization, the system abnormally ends the program and issues abend code X'306' reason code X'04'.

4.3.1.4 Guidelines for using APF

When you use APF authorization, you must control which programs are stored in the authorized libraries. If the first module in a program sequence is authorized, the system assumes that the flow of control to all subsequent modules is known and secure as long as these subsequent modules come from authorized libraries. To ensure that this assumption is valid you should:

- Ensure that all programs that run as authorized programs adhere to your installation's integrity guidelines.
- Ensure that no two load modules with the same name exist across the set of authorized libraries. Two modules with the same name could lead to accidental or deliberate mix-up in module flow, possibly introducing an integrity exposure.
- Link edit with the authorization code (AC=1) only the first load module in a program sequence. Do not use the authorization code for subsequent load modules, thus ensuring that a user cannot call modules out of sequence, or bypass validity checking or critical-logic flow.

It is recommended that you protect the libraries in the APF list with generic security product profiles so only selected users can store programs in those libraries.

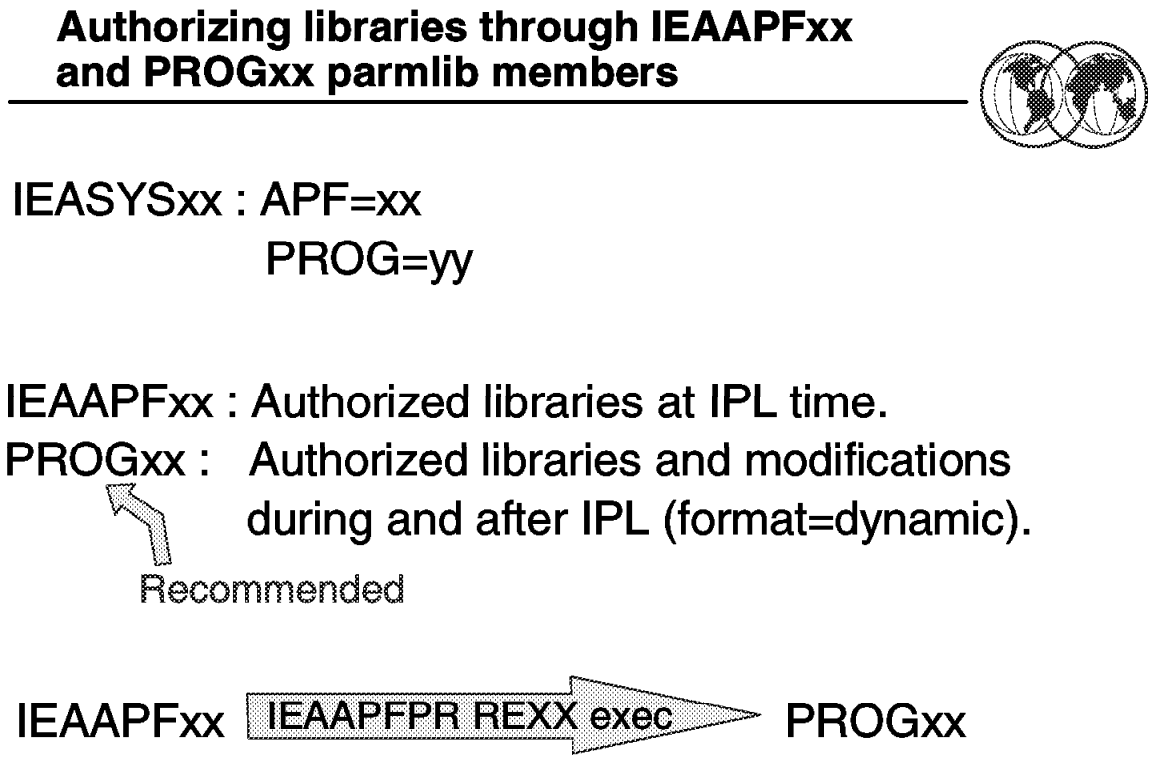


Figure 154. How to get APF authorization

4.3.1.5 Specifying program libraries to be APF authorized

The APF list is built during IPL using those libraries specified in the IEAAPFxx or PROGxx parmlib members, indicated by the APF and PROG parameters in IEASYSxx, or from the operator's console at system initialization.

Note: You can also dynamically modify the APF list after IPL, but only when you have used the dynamic APF format in the PROGxx.

4.3.1.6 IEAAPFxx parmlib member coding

Use the IEAAPFxx member in the SYS1.PARMLIB to specify program libraries that are to receive APF authorization. List the data set names of the libraries, along with the volume where the library resides.

To specify the volume where the library resides, use 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 the library resides on the volume on which the system master catalog resides.
- Nothing after the library name, to indicate that the library is managed by the storage management system (SMS).

Figure 155 on page 232 shows an example of IEAAPFxx.

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(PROGTT)   - 01.01         Line 00000000 Col 001 080
Command ===>                                          Scroll ===> PAGE
************************** Top of Data ********************************
SYS1.VTAMLIB        ******,
SYS1.SICELINK       ******,
SYS1.LOCAL.VTAMLIB  TOTCAT,
ISP.SISPLOAD        *MCAT*
************************** Bottom of Data *****************************
```

Figure 155. An example of IEAAPFxx

4.3.1.7 PROGxx parmlib member coding

You can use the APF statement to specify:

- The format of the APF-authorized library list, whether it is dynamic or static.
- Program libraries to be added to the APF list.
- Program libraries to be deleted from the APF list.

Notes:

1. The system automatically adds SYS1.LINKLIB and SYS1.SVCLIB to the APF list at IPL.
2. If you specify a dynamic APF list format in PROGxx, you can update the APF list at any time during normal processing or at IPL. You can also enter an unlimited number of libraries in the APF list.
3. 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).

The statements and parameters for the APF statement are:

FORMAT(DYNAMIC|STATIC) Specifies that the format of the APF list is dynamic or static.

ADD|DELETE Indicates whether you want to add or delete a library from the APF list.

DSNAME(dsname) The 44-character name of a library that you want to add or delete from the APF list.

SMS|VOLUME(volume) 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.

Syntax format for APF statement

```
APF FORMAT(DYNAMIC|STATIC) APF ADD | DELETE
   DSNAME(dsname)
   SMS | VOLUME(volname)
```

Figure 156 on page 233 shows an example of PROGxx.

```
   Menu  Utilities  Compilers  Help
------------------------------------------------------------------------------
BROWSE  SYS1.PARMLIB(PROGTT)   - 01.01         Line 00000000 Col 001 080
Command ===>                                          Scroll ===> PAGE
************************** Top of Data ********************************
APF FORMAT(DYNAMIC)
APF ADD
    DSNAME(SYS1.VTAMLIB)
    VOLUME(******)
APF ADD
    DSNAME(SYS1.SICELINK)
    VOLUME(******)
APF ADD
    DSNAME(SYS1.LOCAL.VTAMLIB)
    VOLUME(TOTCAT)
APF ADD
    DSNAME(ISP.SISPLOAD)
    VOLUME(*MCAT*)
************************** Bottom of Data *****************************
```

Figure 156. Example of PROGxx

Note

It is recommended that you use the PROGxx instead of IEAAPFxx parmlib member to define the APF list, regardless of whether you plan to take advantage of the dynamic update capability.

If you specify both the PROG=xx and the APF=xx parameters, the system places into the APF list those libraries listed in the IEAAFPxx, followed by those libraries in the PROGxx.

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

For more information, see *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

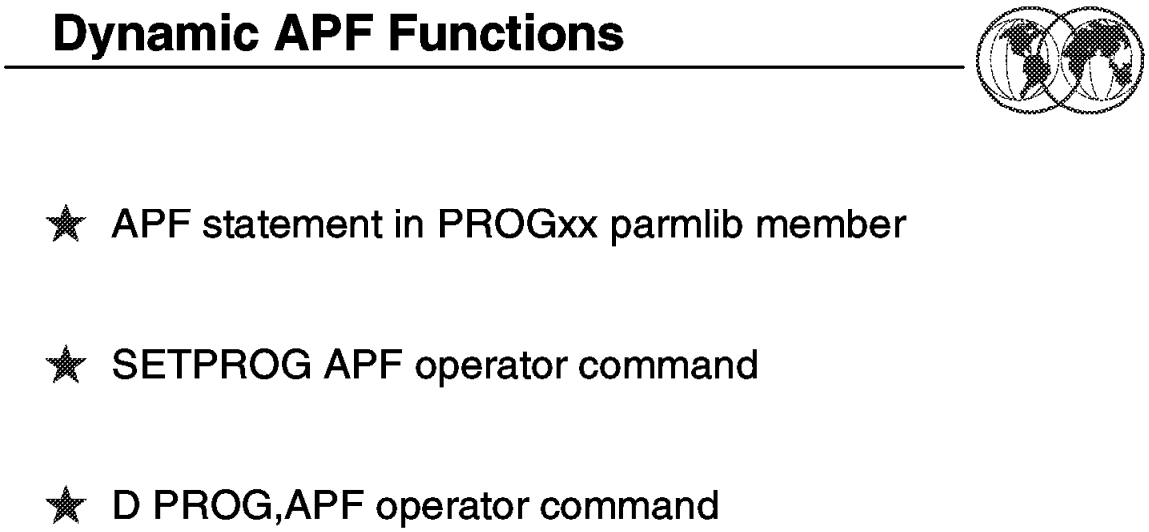


Figure 157. Dynamic APF functions

4.3.2 Managing dynamic APF

You can use the following ways to update the content of the APF table dynamically:

- Use PROGxx parmlib member which includes the appropriate APF statement to define the change.
- The SETPROG AFP operator can also initiate a change to the APF table.
- The DISPLAY AFP command may be used to display the list of libraries authorized by APF.

4.3.2.1 Using the PROGxx parmlib member

As mentioned in the previous section, you can use the APF statement to add or delete libraries from the APF list.

To delete a library from the APF list use the following command:

Example of the APF DELETE

```
APF DELETE
   DSNAME(SCRIPT.R40.DCFLOAD)
   VOLUME(MPRES2)
```

Activate the change by using a SET PROG=xx command as shown in Figure 158 on page 235.

```
SET PROG=T4
IEE252I MEMBER PROGT4   FOUND IN SYS1.PARMLIB
CSV410I DATA SET SCRIPT.R40.DCFLOAD ON VOLUME MPRES2 DELETED FROM APF
LIST
IEE536I PROG     VALUE T4 NOW IN EFFECT
```

Figure 158. Example of SET PROGxx command to activate an APF list change

To add a library to the APF list use the following command:

Example of the APF ADD

```
APF ADD
   DSNAME(MYPROG.LOADLIB)
   VOLUME(MPRES3)
```

Activate the change by using a SET PROG=xx command as shown in Figure 159.

```
SET PROG=T5
IEE252I MEMBER PROGT5   FOUND IN SYS1.PARMLIB
CSV410I DATA SET MYPROG.LOADLIB ON VOLUME MPRES2 ADDED TO APF LIST
IEE536I PROG     VALUE T5 NOW IN EFFECT
```

Figure 159. Activating a change to add an APF library

4.3.2.2 Using the SETPROG APF command

Use the SETPROG APF operator command to perform the following functions:

- Change the format of the authorized program facility (APF) list from static to dynamic, or static to dynamic.
- Add a library to a dynamic APF list.
- Delete a library from a dynamic APF list.

Syntax format for SETPROG APF command

```
SETPROG APF{,FORMAT={DYNAMIC|STATIC}}
      {,{ADD|DELETE},DSNAME|LIBRARY=libname,{SMS|VOLUME=volume}}
```

To change the format of the APF list to dynamic:

```
SETPROG APF,FORMAT=DYNAMIC
CSV410I APF FORMAT IS NOW DYNAMIC
```

To add a library to the APF list:

```
SETPROG APF,ADD,DSNAME=SCRIPT.R40.DCFLOAD,VOLUME=MPRES2
CSV410I DATA SET SCRIPT.R40.DCFLOAD ON VOLUME MPRES2 ADDED TO APF LIST
```

To delete a library from the APF list:

```
SETPROG APF,DELETE,DSNAME=SCRIPT.R40.DCFLOAD,VOLUME=MPRES2
CSV410I DATA SET SCRIPT.R40.DCFLOAD ON VOLUME MPRES2 DELETED FROM APF
```

Note: If you try to add or delete from a APF list that is in a static format, the system responds with a CSV411I message.

```
SETPROG APF,DELETE,DSNAME=SCRIPT.R40.DCFLOAD,VOLUME=MPRES2
CSV411I ADD/DELETE IS NOT ALLOWED BECAUSE APF FORMAT IS STATIC
```

4.3.2.3 Using the DISPLAY APF command

You can use the DISPLAY APF command to display or more entries in the list of APF-authorized libraries. Each entry in the APF list display contains:

- An entry number.
- The name of an authorized library.
- An identifier for the volume on which the authorized library resides (or *SMS*, if the library is SMS-managed).

Syntax for DISPLAY APF command

```
D PROG,APF[,ALL         ]
          |,DSNAME=libname
          |,ENTRY=xxx
          |,ENTRY=(xxx-yyy)

   ],L={a|cc|cca|name|name-a} ]
```

To display all the whole APF list:

```
D PROG,APF
CSV450I 05.18.16 PROG,APF DISPLAY 971
FORMAT=DYNAMIC
ENTRY VOLUME DSNAME
   1  MPRES1 SYS1.LINKLIB
   2  MPRES1 SYS1.SVCLIB
   3  MPRES1 CPAC.LINKLIB
   1  .
              .
              .
  36  MPRES2 NETVIEW.V2R4M0.USERLNK
  37  MPRES2 NPM.V2R3M0.SFNMLMD1
  38  MPRES2 EMS.V2R1M0.SEMSLMD0
```

1 This is the decimal entry number for each of the libraries defined in the APF list.

To display the specific library at entry number 1:

```
D PROG,APF,ENTRY=1
CSV450I 05.24.55 PROG,APF DISPLAY 979
FORMAT=DYNAMIC
ENTRY VOLUME DSNAME
   1  MPRES1 SYS1.LINKLIB
```

To display the all the libraries in the range from 1 to 5:

```
D PROG,APF,ENTRY=(1-5)
CSV450I 05.26.27 PROG,APF DISPLAY 981
FORMAT=DYNAMIC
ENTRY VOLUME DSNAME
   1  MPRES1 SYS1.LINKLIB
   2  MPRES1 SYS1.SVCLIB
   3  MPRES1 CPAC.LINKLIB
   4  MPRES1 ISF.SISFLOAD
   5  MPRES1 ISF.SISFLPA
```

Chapter 5. Catalogs

A catalog describes data set attributes and records the location of a data set so that the data set can be retrieved without requiring the user to specify the data set's location. Multiple user catalogs contain information about user data sets, and a single master catalog contains entries for system data sets and user catalogs.

In OS/390 the component controlling catalogs, also called Catalog Manager, is embedded in DFSMSdfp, which supports three types of catalogs, which can coexist on the same operating system:

- ICF catalogs
- VSAM catalogs
- CVOLs

All data sets managed by the storage management subsystem (SMS) must be cataloged in a ICF catalog.

Most installations depend on the availability of catalog facilities to run production job streams and to support online users. For maximum reliability and efficiency, all permanent data sets should be cataloged and all catalogs should be integrated catalog facility catalogs. See *DFSMS/MVS Version 1 Release 4 Managing Catalogs*, SC26-4914, for information on VSAM catalogs and CVOLS and how to convert to integrated catalogs.

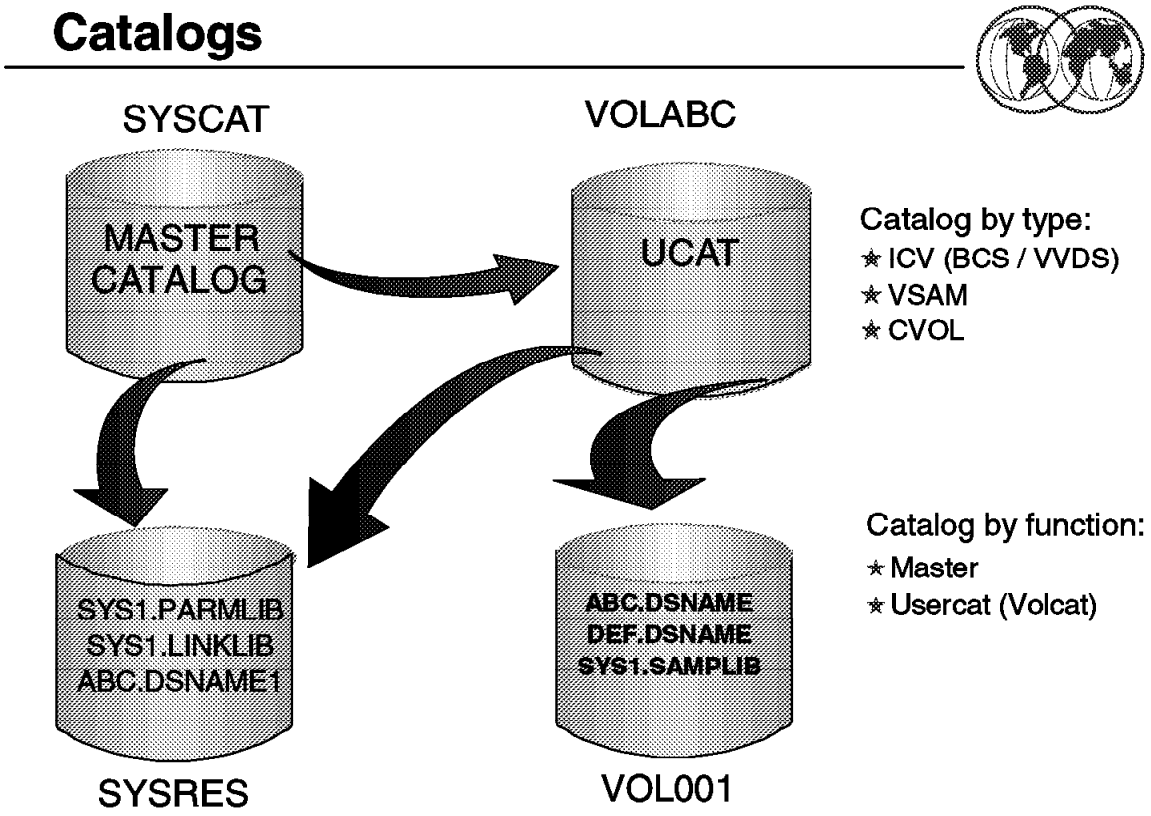


Figure 160. Catalogs

5.1 Catalogs

A catalog is a data set which contains information about other data sets. It provides users with the ability to locate a data set by name, without knowing where the data set resides. By cataloging data sets, your users need to know less about your storage setup. Thus, data sets can be moved from one device to another, without requiring a change in JCL DD statements which refer to an existing data set.

Cataloging data sets also simplifies backup and recovery procedures. Catalogs are the central information point for VSAM data sets; all VSAM data sets must be cataloged. In addition, all SMS-managed data sets must be cataloged.

DFSMS/MVS provides three types of catalogs: integrated catalog facility catalogs (ICV), VSAM catalogs, and operating system control volumes (OS CVOLs).

The integrated catalog facility should satisfy all your cataloging needs. Any type of data set or object can be cataloged in an integrated catalog facility catalog. Functions provided for VSAM catalogs and OS CVOLs are for compatibility only. Many advanced MVS/DFP functions require the use of integrated catalog facility catalogs (for example, the storage management subsystem).

This chapter will provide information only about ICF Catalogs, for further information about VSAM Catalogs and OS CVOLs catalogs, refer to *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

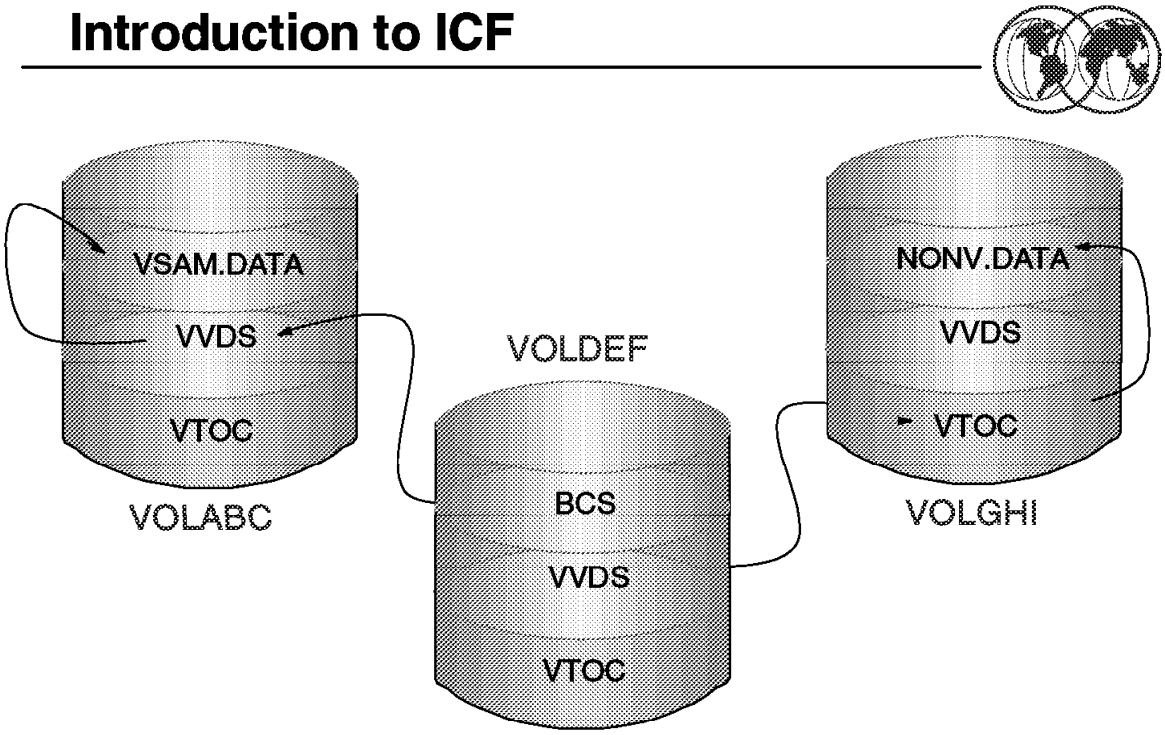


Figure 161. Introduction to ICF

5.2 Introduction to ICF

An integrated catalog facility has two components, a VSAM volume data set (VVDS) and a basic catalog structure (BCS), the following topics explain these components in more detail.

5.2.1 Basic catalog structure (BCS)

The basic catalog structure is a VSAM key-sequenced data set. It uses the data set name as a key to store and retrieve data set information. For VSAM data sets, the BCS contains volume, security, ownership, and association information. For non-VSAM data sets, the BCS contains volume ownership, and association information. In other words the BCS portion of the ICF catalogs contains the *static* information about the data set, the information that changes very seldom.

For non-VSAM data sets that are not SMS-managed all their catalog information is only contained in the BCS. For the other types of data sets, there is other information available in the VVDS.

Related information in the BCS is grouped into logical, variable-length, spanned records related by key. The BCS uses keys that are the data set names (plus one character for extensions). A control interval can contain multiple BCS records. To reduce the number of I/Os necessary for catalog processing, logically-related data is consolidated in the BCS.

5.2.2 VSAM volume data set (VVDS)

The VVDS is a VSAM entry-sequenced data set (ESDS) that has a 4 KB control interval size. It contains additional catalog information (not contained in the BCS) about the VSAM and SMS-managed non-VSAM data sets residing on the volume where the VVDS is located. Every volume containing any VSAM or any SMS-managed data sets *must* have a VVDS in it. In a sense, we may say that the VVDS is a sort for VTOC extension for certain type of data sets. A VVDS may have data set information about data sets cataloged in distinct BCSs.

VVDs contains the data set characteristics, extent information, and the volume-related information of the VSAM data sets cataloged in the BCS. If you are using the storage management subsystem (SMS), the VVDS also contains data set characteristics and volume-related information for the non-VSAM, SMS-managed data sets on the volume. As you can see the type of information retained in VVDS is more frequently modified or more *volatile* than the one in BCS.

A VVDS is recognized by the restricted data set name SYS1.VVDS.Vvolser, where *volser* is the volume serial number of the volume on which the VVDS resides.

You can explicitly (via IDCAMS) define the VVDS, or it is implicitly created after you define the first VSAM data set in the volume or the first non-VSAM SMS-managed data set.

An explicitly defined VVDS is not related to any BCS until a data set or VSAM object is defined on the volume. As data sets are allocated on the VVDS volume, each BCS with VSAM or SMS-managed data sets residing on that volume is related to the VVDS. An explicit definition of a VVDS does not update any BCS and, therefore, can be performed before the first BCS in the installation is defined. Explicitly defining a VVDS is usually appropriate when you are initializing a new volume. If you are not running SMS, and a volume already contains some non-VSAM data sets, it is appropriate to allow the VVDS to be defined implicitly with the default space allocation of TRACKS(10 10).

The VVDS is composed of a minimum of two records:

- A VSAM volume control record (VVCR)
- A VVDS self-describing volume record

The first logical record in a VVDS is the VSAM volume control record (VVCR). It contains information for management of DASD space and the BCS names which currently have cataloged VSAM or SMS-managed non-VSAM data sets on the volume. It might have a pointer to an overflow VVCR.

The second logical record in the VVDS is the VVDS self-describing VVR (VSAM volume record). This self-describing VVR contains information that describes the VVDS.

The remaining logical records in the VVDS are VVRs for VSAM objects or non-VSAM volume records (NVRs) for SMS-managed non-VSAM data sets. The hexadecimal RBA of the record is used as its key or identifier.

5.2.3 VSAM volume records (VVR)

VSAM volume records contain information about the VSAM data sets residing on the volume with the VVDS. The number of VVRs for VSAM data sets varies according to the type of data set and the options specified for the data set.

5.2.4 Non-VSAM volume record (NVR)

The non-VSAM volume record (NVR) is equivalent to a VVR record, but the NVR record is for SMS-managed non-VSAM data sets. The NVR contains SMS-related information.

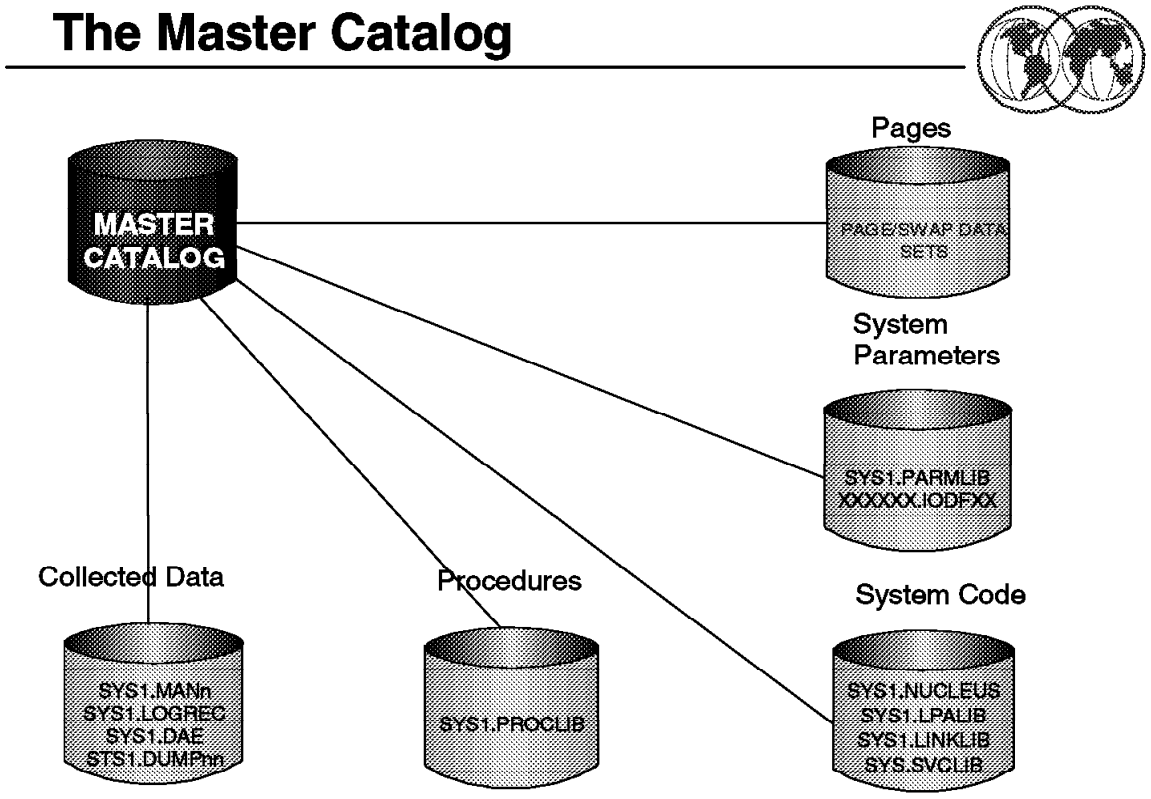


Figure 162. Master catalog

5.3 Catalogs by function

By function the catalogs can be classified as Mastercat (master catalog) and Usercats (user catalogs). A particular case of Usercat is the Volcat, that is a Usercat containing only tape library and tape volume entries.

There is no structural difference between a Mastercat and a Usercat. What makes a Mastercat different is how it is used, and what data sets are cataloged in it.

5.3.1 Master catalog

Each system has one active Mastercat. The Mastercat does not have to reside on the system residence volume. For performance, recovery, and reliability, we recommend that you only use integrated catalog facility catalogs. Also, a Mastercat can be shared between different MVS images.

The Mastercat for a system must contain entries for all Usercats (and their aliases), which the system uses. The only other data sets which you should catalog in the Mastercat are the system, or *SYS1* data sets. These data sets must be cataloged in the Mastercat for proper system initialization.

5.3.2 The Mastercat at system initialization

During a system initialization, the Mastercat is read so that system data sets and Usercats can be located. Their catalog entries are placed in the in-storage catalog cache as they are read.

Catalog aliases are also read during system initialization, but they are placed in an alias table separate from the in-storage catalog. The concept of aliases is covered in 5.3.4.1, “Using aliases” on page 246.

Thus, if the Mastercat only contains entries for system data sets, catalogs, and catalog aliases, the entire Mastercat is in main storage by the completion of the system initialization.

5.3.2.1 Identifying the Mastercat

At IPL, you must indicate the location (Volser and data set name) of the Mastercat, this can be done by:

- SYSCATxx member of SYS1.NUCLEUS data set or the default member name that is SYSCATLG (also in SYS1.NUCLEUS).
- LOADxx member of SYS1.PARMLIB / SYS1.IPLPARM. We recomend this method.

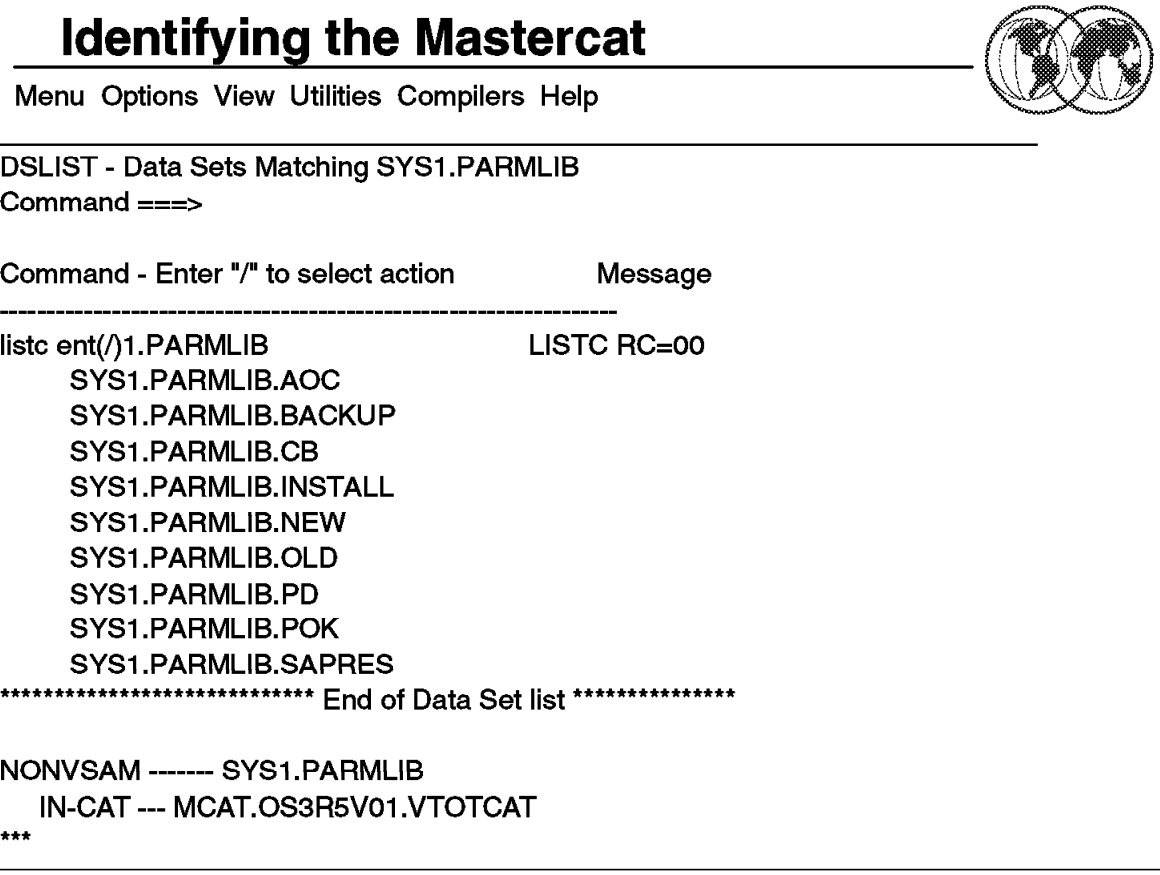


Figure 163. Identifying the Mastercat

5.3.3 Identifying the Mastercat

The LISTCAT command lists each catalog entry with the generic name GENERIC.*.BAKER, where * is any 1- to 8-character simple name. The parameters are:

ENTRIES Specifies the entryname of the object to be listed. Because GENERIC.*.BAKER is a generic name, more than one entry can be listed

An easy way to identify the Mastercat (after definition) is via ISPF 3.4. Type SYS1.PARMLIB and when the SYS1.PARMLIB data sets are listed, enter beside the SYS1.PARMLIB name type LISTC ENT(/), as shown in this visual.

Note: The / specifies to use the data set name on the line where the command is entered.

The response to the command is shown at the bottom of the visual as follows:

```
NONVSAM ------- SYS1.PARMLIB
     IN-CAT --- MCAT.OS3R5V01.VTOTCAT
```

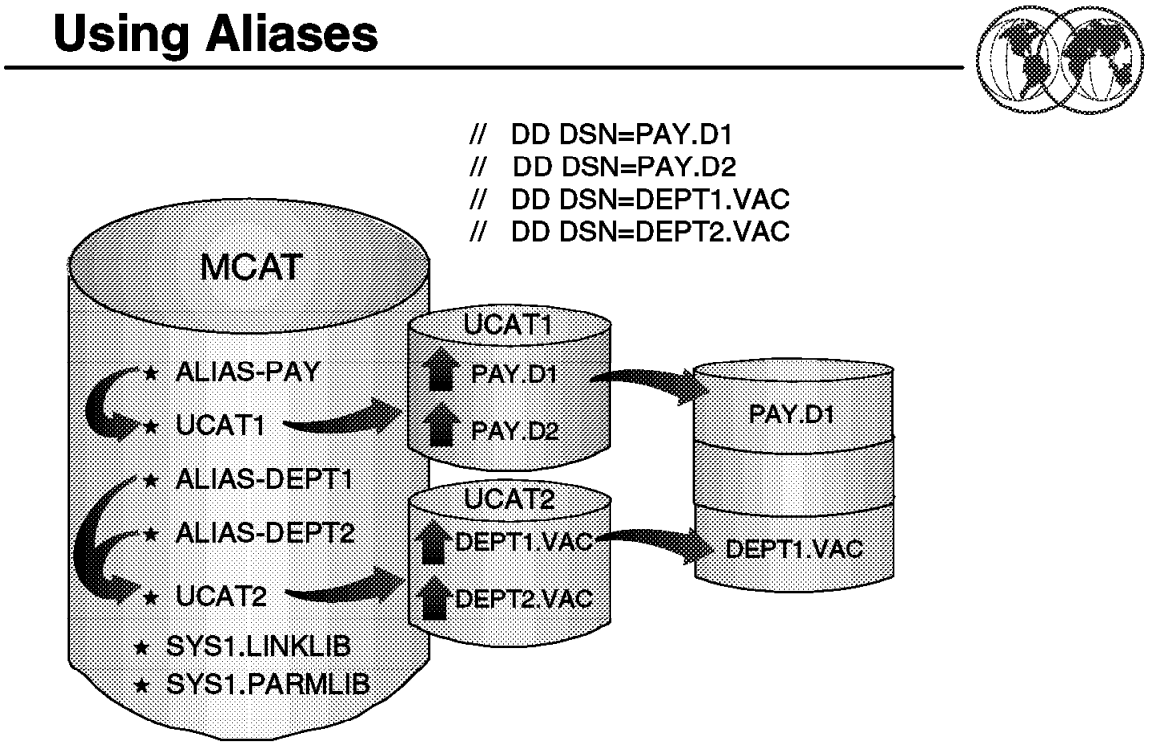


Figure 164. Using aliases

5.3.4 Usercats

An ICF Usercat has the same structure as an ICF Mastercat. The difference is in the Usercat *function*. Mastercat should be used to contain information about system data sets (SYS1.) and pointers to Usercats. Usercats should be used to contain information about your installation cataloged data sets, this is implemented through the alias concept.

5.3.4.1 Using aliases

The way to tell catalog management in which Usercat your data set is cataloged is through aliases. You define an appropriate alias name for an Usercat in the Mastercat. Next, you match the highest-level qualifier (HLQ) of your data set with the alias. This, identifies the appropriate Usercat to be used to satisfy the request.

In this visual all data sets with an HLQ of PAY, have their information in the Usercat UCAT1, because in the Mastercat there is an alias PAY pointing to UCAT1. The ones with DEPT1 and DEPT2 have their information in the Usercat UCAT2, because in the Mastercat there are aliases DEPT1 and DEPT2 pointing to UCAT2.

Note that aliases can also be used with non-VSAM data sets in order to create alternate names to the same data set.

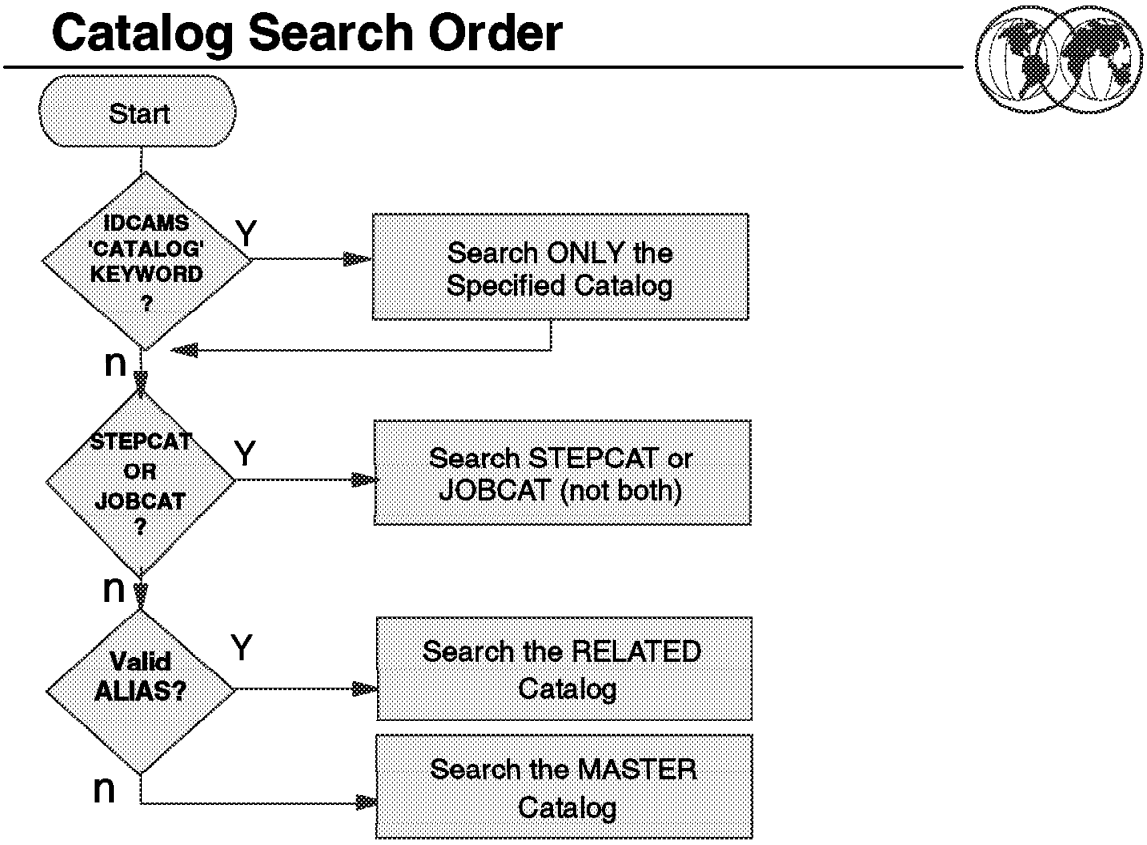


Figure 165. The catalog search order

5.3.5 Catalog search order

Most catalog searches should be based on catalog aliases. Some alternatives to catalog aliases are available for directing a catalog request, specifically the JOBCAT and STEPCAT DD statements, the CATALOG parameter of access method services, and the name of the catalog.

5.3.5.1 Search order for catalogs

For the system to determine where a data set is to be cataloged, the following search order is used to find the catalog:

1. If the IDCAMS Define (creation and cataloging) statement is used and the CATALOG parameter (directs the search to an specific catalog) is indicated, then use this referred catalog
2. Otherwise, use the catalog named in the STEPCAT DD statement
3. If no STEPCAT, use the catalog named in the JOBCAT DD statement
4. If no JOBCAT, and the HLQ is a catalog alias, use the catalog identified by the alias or the catalog whose name is the same as the same HLQ of the data set
5. If no catalog has been identified yet, use the Mastercat.

5.3.5.2 Search order for locating

The following search order is used to locate the catalog for an already-cataloged data set:

1. If at any IDCAMS invocation where there is a need to locate a data set and the CATALOG parameter (directs the search to an specific catalog) is indicated, then use this referred catalog. If the data set is not found, fail the job.
2. Otherwise, search all catalogs specified in a STEPCAT DD statement in order.
3. If not found, search all catalogs specified in a JOBCAT DD statement in order.
4. If not found, and the HLQ is an alias for a catalog, search the catalog; or if the HLQ is the name of a catalog, search the catalog. If the data set is not found, fail the job.
5. Otherwise, search the Mastercat

Note: For SMS-managed data sets, JOBCAT and STEPCAT DD statements are not allowed and cause a job failure. Also, they are not recommended even for non-SMS data sets. So, do not use them.

To use an alias to identify the catalog to be searched, the data set or object name, or the generation data group base name, must be a qualified name.

When you specify a catalog in the IDCAMS CATALOG parameter, and you have appropriate RACF authority to the FACILITY class profile STGADMIN.IGG.DIRCAT, the catalog you specify is used. For instance:

```
DEFINE CLUSTER (NAME(PROD.PAYROLL) CATALOG(SYS1.MASTER.ICFCAT))
```

defines a data set PROD.PAYROLL to be cataloged in SYS1.MASTER.ICFCAT.

You can use RACF to prevent the use of the CATALOG parameter and restrict the ability to define data sets in the Mastercat.

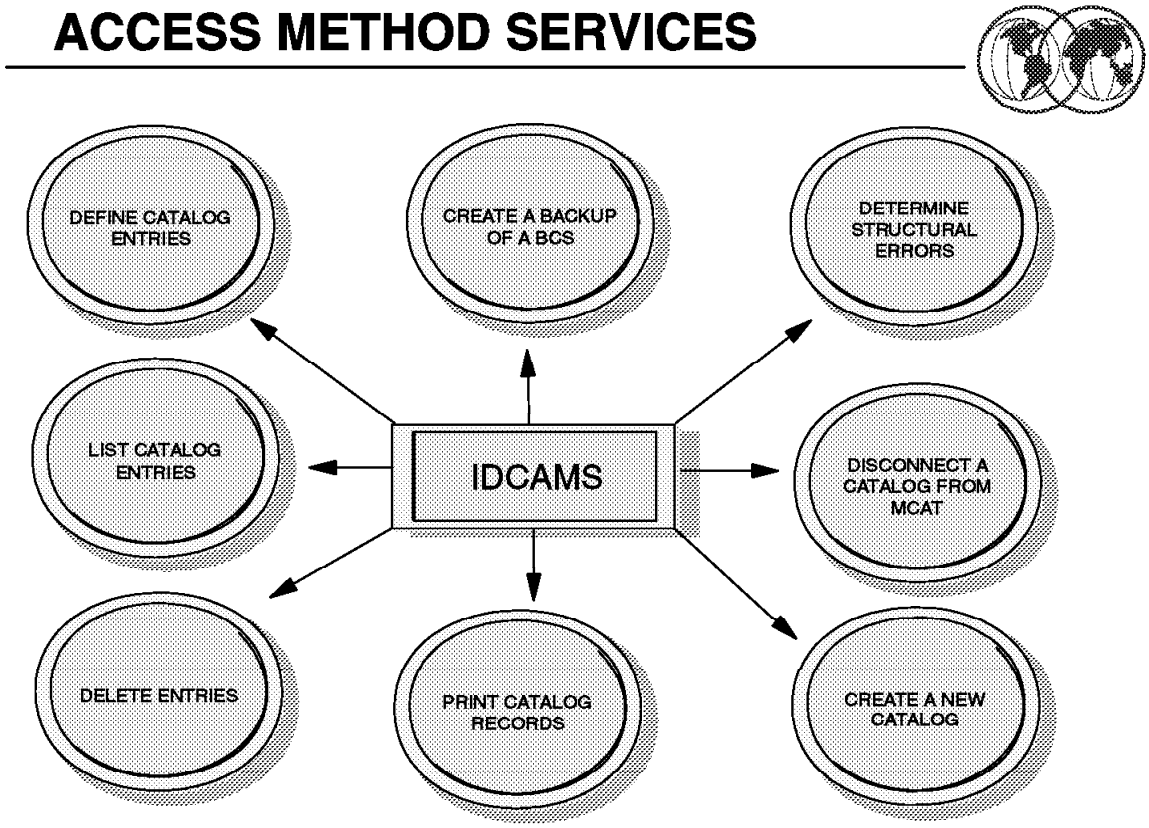


Figure 166. Access method service

5.4 Access method service (AMS)

You use access method services to define and maintain catalogs. Access method services commands can also be used to define and maintain VSAM and non-VSAM data sets. For a complete explanation of the usage of access method services, the required JCL, and examples, see *DFSMS/MVS Version 1 Release 4: Access Method Services for Integrated Catalog Facility*, SC26-4906.

Table 4 lists some Access Method Service commands that can be used with catalogs.

Table 4 (Page 1 of 2). Some AMS commands that can be used with catalogs

Command	**Explanation**
ALTER(1)	Alters previously defined catalog entries or certain catalog attributes, including entries for tape volume catalogs.
CNVTCAT	Converts entries in an OS CVOL or VSAM catalog into integrated catalog facility catalog entries.
CREATE(1)	Creates tape library or tape volume entries.
DEFINE	Defines catalog entries.
DEFINE CLUSTER	Defines VVDSs and catalog entries.
DEFINE USERCATALOG	Defines user catalogs including tape volume catalogs (DEFINE USERCATALOG VOLCATALOG).

Table 4 (Page 2 of 2). Some AMS commands that can be used with catalogs	
Command	**Explanation**
DELETE(1)	Deletes catalog and VVDS entries, data sets, catalogs, VVDSs, and data set control blocks in the VTOC.
DIAGNOSE(2)	Determines whether the content of BCS or VVDS records is invalid or unsynchronized.
EXAMINE	Determines whether structural errors exist in the index or data component of a BCS.
EXPORT	Creates a backup copy of a catalog or copy that can be moved to another system or volume.
EXPORT DISCONNECT	Disconnects the catalog from the Mastercat.
IMPORT	Restores an exported backup copy of a catalog, or makes a catalog that was previously exported from one system, available for use in another system.
IMPORT CONNECT	Connects the catalog to the Mastercat on the same system, or on any shared system.
LISTCAT	Lists catalog entries including entries for tape volume catalogs.
PRINT	Prints data set or catalog records.
REPRO	Copies catalogs; splits catalog entries between two catalogs; merges catalog entries into another user catalog or Mastercat. REPRO supports tape volume catalogs.
VERIFY	Causes a catalog to reflect the end of a data set correctly after an error that prevented closing a VSAM data set. The error might cause the catalog to be incorrect.

Notes:

1. The access method services ALTER, CREATE, and DELETE commands should only be used to recover from tape volume catalog errors. since access method services cannot change the library manager inventory in an automated tape library, ISMF should be used for normal tape library ALTER, CREATE, and DELETE functions.
2. DIAGNOSE recognizes tape library and tape volume record types. It checks the cell structure of the volume catalog.

Creating a Basic Catalog Structure

```
 //DEFCAT JOB....
 //DEFCAT EXEC PGM=IDCAMS
 //SYSPRINT DD SYSOUT=A
 //SYSIN DD *
   DEFINE USERCATALOG     -
   (NAME(SYS1.ICFCAT)     -
   MEGABYTES( 15 15)      -
   VOLUME(FERNAN)         -
   ICFCATALOG             -
   FREESPACE(10 10)       -
   STRNO(3)               -
   IMBED                  -
   REPLICATE )            -
   DATA( CISIZE(4096)     -
   BUFND(4) )             -
   INDEX( BUFNI(4)  )     -
```

Figure 167. Creating a basic catalog structure

5.5 Creating a basic catalog structure (BCS)

Use the access method services DEFINE USERCATALOG ICFCATALOG command to define the basic catalog structure of an ICF catalog. This visual shows an example of using this command.

Use the access method services DEFINE USERCATALOG VOLCATALOG to define a catalog that only contains tape library and tape volume entries. See *DFSMS/MVS Version 1 Release 4: Access Method Services for Integrated Catalog Facility*, SC26-4906, for more detail.

Defining a BCS with Model

```
//DEFCAT4   JOB     ...
//STEP1     EXEC    PGM=IDCAMS
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      *
      DEFINE USERCATALOG( -
             NAME(RSTUCAT2) -
             VOLUME(VSER03) -
             MODEL(USERCAT4 -
               USERCAT4)   ) -
           CATALOG(AMAST1)
/*
```

Figure 168. Defining a BCS with model

5.5.1 Defining a BCS with model

When you define a BCS or VVDS, you can use an existing BCS or VVDS as a model for the new one. The attributes of the existing data set are copied to the newly defined data set, unless you explicitly specify a different value for an attribute. You can override any of a model's attributes.

If you do not want to change or add any attributes, you need only supply the entry name of the object being defined and the MODEL parameter. When you define a BCS, you must also specify the volume and space information for the BCS.

In this visual there is one example of that; where:

- NAME specifies the name of the catalog to be defined as RSTUCAT2.
- VOLUME specifies that the catalog is to reside on volume VSER03. Volume VSER03 is dynamically allocated.
- MODEL identifies USERCAT4 as the catalog to use as a model for RSTUCAT2. The attributes and specifications of USERCAT4 that are not otherwise specified with the shown parameters are used to define the attributes and specifications of RSTUCAT2. The master catalog, AMAST1, contains a user-catalog connector entry that points to USERCAT4. This is why USERCAT4 is specified as MODEL's catname subparameter. Values and attributes that apply to RSTUCAT2 as a result of using USERCAT4 as a model are:
  - FOR = 365 days (retention period)
  - CYLINDERS = 3 (primary) and 2 (secondary) are allocated to the catalog

- BUFFERSPACE = 3072 bytes
- ATTEMPTS = 2
- NOWRITECHECK
- CODE is null
- AUTHORIZATION is null
- OWNER is null

Because catalogs contain their own entries, you must specify the name of the catalog twice in the MODEL parameter.

Defining Aliases

```
//DEFALIAS JOB MSGCLASS=X,NOTIFY=FPITA
//ALIAS    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
       DEFINE ALIAS -
              (NAME(FPITA)      -
              RELATE(SYS1.ICFCAT.UCAT))

       DEFINE ALIAS -
              (NAME(VERA)       -
              RELATE(SYS1.ICFCAT.UCAT))
   /*
```

Figure 169. Defining aliases

5.5.2 Defining aliases

To use a catalog, the system must be able to determine which data sets should be defined in that catalog. The simplest way to accomplish this is to define aliases in Mastercat for the catalog—if it is a Usercat. Before defining an alias, carefully consider the effect the new alias has on old data sets. A poorly chosen alias could make some data sets inaccessible.

You can define aliases for the Usercat in the same job in which you define the catalog by including DEFINE ALIAS commands after the DEFINE USERCATALOG COMMAND. You can use conditional operators to ensure the aliases are only defined if the catalog is successfully defined. After the catalog is defined, you can add new aliases or delete old aliases.

Catalog aliases are defined only in the Mastercat, which contains an entry for the user catalog. The number of aliases a catalog can have is limited by the maximum record size for the Mastercat. If the Mastercat is defined with the default record sizes, there is a practical maximum of 3000 aliases per catalog, assuming the aliases are only for HLQs. If you use multilevel aliases, fewer aliases per catalog can be defined.

You cannot define an alias if a data set cataloged in the Mastercat has the same high-level qualifier as the alias. The DEFINE ALIAS command fails with a *duplicate data set name* error. For example, if a catalog is named TESTE.TESTSYS.ICFCAT, you cannot define the alias TESTE for any catalog.

This visual shows a DEFINE ALIAS job.

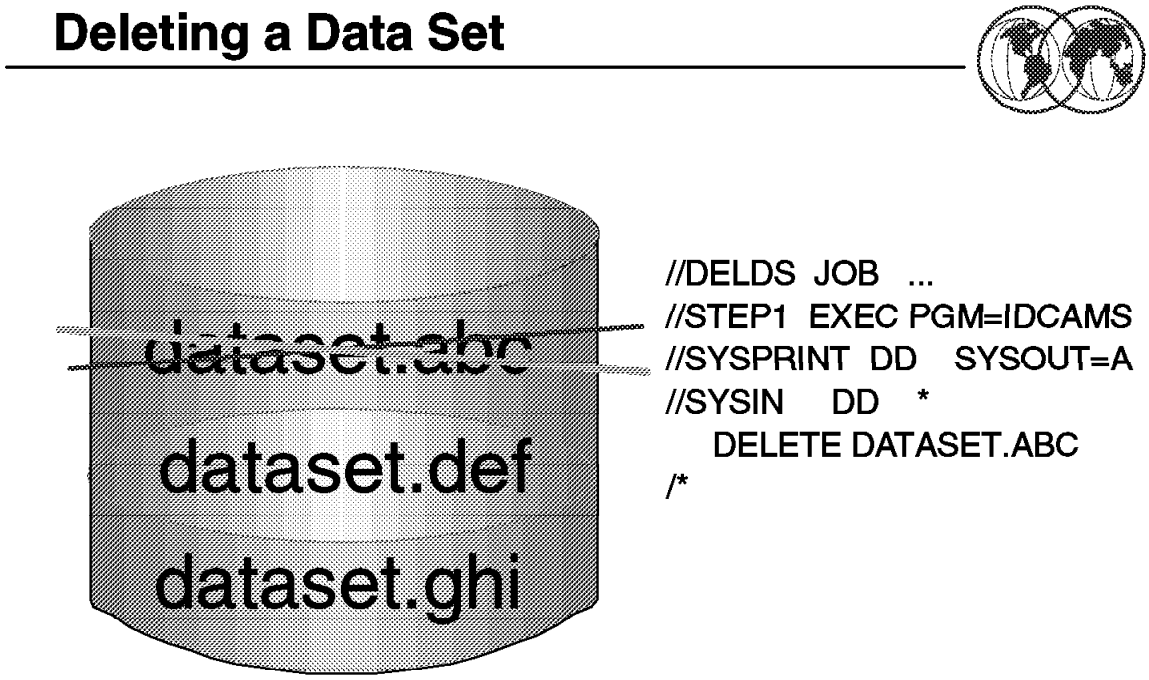


Figure 170. Deleting a data set

5.6 Deleting data entities

Sometimes in your installation you need to delete an alias, delete only a catalog entry, or you may have garbage in your DASD and need to delete a VSAM/non-VSAM data set that is not cataloged anymore. The next few topics will describe some of these situations and show you how to handle them.

To delete a data set (uncatalog and scratch from VTOC) that is cataloged and is in DASD, you can simply issue a command DELETE in ISPF 3.4 in front of its name, issue the DELETE dsname in ISPF option 6 (commands), or submit a batch IDCAMS job to delete it, as shown in this visual.

5.6.1 Alias

To simply delete an alias, use the IDCAMS DELETE ALIAS command, specifying the alias you are deleting.

To delete all the aliases for a catalog, use EXPORT DISCONNECT to disconnect the catalog. The aliases are deleted when the catalog is disconnected. When you again connect the catalog (using IMPORT CONNECT) the aliases remain deleted.

5.6.2 Deleting the catalog entry

To only delete a catalog entry you can the use the DELETE NOSCRATCH command, the VVDS and VTOC entry is not deleted. The entry deleted can be reinstated with the DEFINE RECATALOG command as shown in Figure 171 on page 256.

```
//DELNOSCR  JOB   ...
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
     DELETE VERA.DATASET NOSCR
/*
```

Figure 171. Example DEFINE RECATALOG command

5.6.3 VVR and NVR records

When the catalog entry is missing, and the data set remain in the DASD you can use the DELETE VVR for VSAM data sets and DELETE NVR for non-VSAM SMS-managed data sets as shown in Figure 172.

```
//DELGDG   JOB   ...
//STEP1     EXEC  PGM=IDCAMS
//DD1       UNIT=3390,VOL=SER=FERNAN
//SYSPRINT  DD  SYSOUT=A
            DELETE  -
            VERA.LUZ.VSAM.INDEX -
            FILDE(DD1) -
            VVR
```

Figure 172. Example DELETE VVR command

Caution

When deleting VSAM KSDS, you must issue a DELETE VVR for each of the components, the DATA, and the INDEX.

5.6.4 Generation data group (GDG)

In this example, a generation data group base catalog entry, GDGBASE is deleted from the catalog. The generation data sets associated with GDGBASE remain unaffected in the VTOC, as shown in the JCL shown in Figure 173.

```
//DELGDG   JOB   ...
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
            DELETE -
            GDGBASE -
            GENERATIONDATAGROUP -
            RECOVERY
/*
```

Figure 173. Example of deleting a GDB catalog entry

The DELETE command removes the GDG base catalog entry from the catalog. Its parameters are:

GDGBASE GDGBASE is the name of the GDG base entry.

GENERATIONDATAGROUP GENERATIONDATAGROUP specifies the type of entry being deleted. VSAM verifies that GDGBSE is a GDG entry, then deletes it. If GDGBASE is not a GDG entry, VSAM issues a message and does not delete it.

RECOVERY RECOVERY specifies that only the GDG base entry name in the catalog be deleted. Its associated generation data sets remain intact in the VTOC.

5.6.5 Delete an ICF

When deleting an ICF, you must take care if you want to delete only the catalog, or if you want to delete all associated data. The following examples show how to delete a catalog.

5.6.5.1 Delete with recovery

In this example, a user catalog is deleted in preparation for replacing with an imported backup copy. The VVDS and VTOC entries for objects defined in the catalog are not deleted and the data sets are not scratched, as shown in Figure 174.

```
//DELICF     JOB   ...
//STEP1      EXEC  PGM=IDCAMS
//DD1        DD    VOL=SER=CACSW3,UNIT=3380,DISP=OLD
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
     DELETE -
            USERCAT -
            FILE(DD1) -
            RECOVERY -
            USERCATALOG
/*
```

Figure 174. Example of deleting and replacing a user catalog

RECOVERY

Specifies that only the catalog data set is deleted without deleting the objects defined in the catalog.

5.6.5.2 Delete an empty user catalog

In the example shown in Figure 175, a user catalog is deleted. A user catalog can be deleted when it is empty—that is, when there are no objects cataloged in it other than the catalog's volume. If the catalog is not empty, it cannot be deleted unless the FORCE parameter is specified.

```
//DELICF   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DELETE -
            CATALOG.ICF.UCAT -
            PURGE -
            USERCATALOG
/*
```

Figure 175. Example of deleting an empty user catalog

Attention:

- The FORCE parameter deletes all clusters in the catalog.
- The DELETE command deletes both the catalog and the catalog's user catalog connector entry in the Mastercat.

For more information about the DELETE command, refer to *DFSMS/MVS Version 1 Release 4: Access Method Services for Integrated Catalog Facility*, SC26-4906.

5.6.6 Delete a migrated data set

A migrated data set is a data set moved by DFSMShsm to a cheaper storage device to make room in your primary DASD farm. MVS recognizes that a data set is migrated by the MIGRAT word in its catalog entry.

When you need to delete a data set with MIGRAT status in the catalog, but the data set is not referenced in the HSM control data sets, you should execute a DELETE NOSCRATCH command again for this data set.

To execute the DELETE command against a migrated data set, you must have a RACF group ARCCATGP defined, and logon in TSO using this group.

For further information about ARCCATGP group, refer to *DFSMS/MVS Version 1, Release 4: DFSMShsm Implementation and Customization Guide*, SH21-1078.

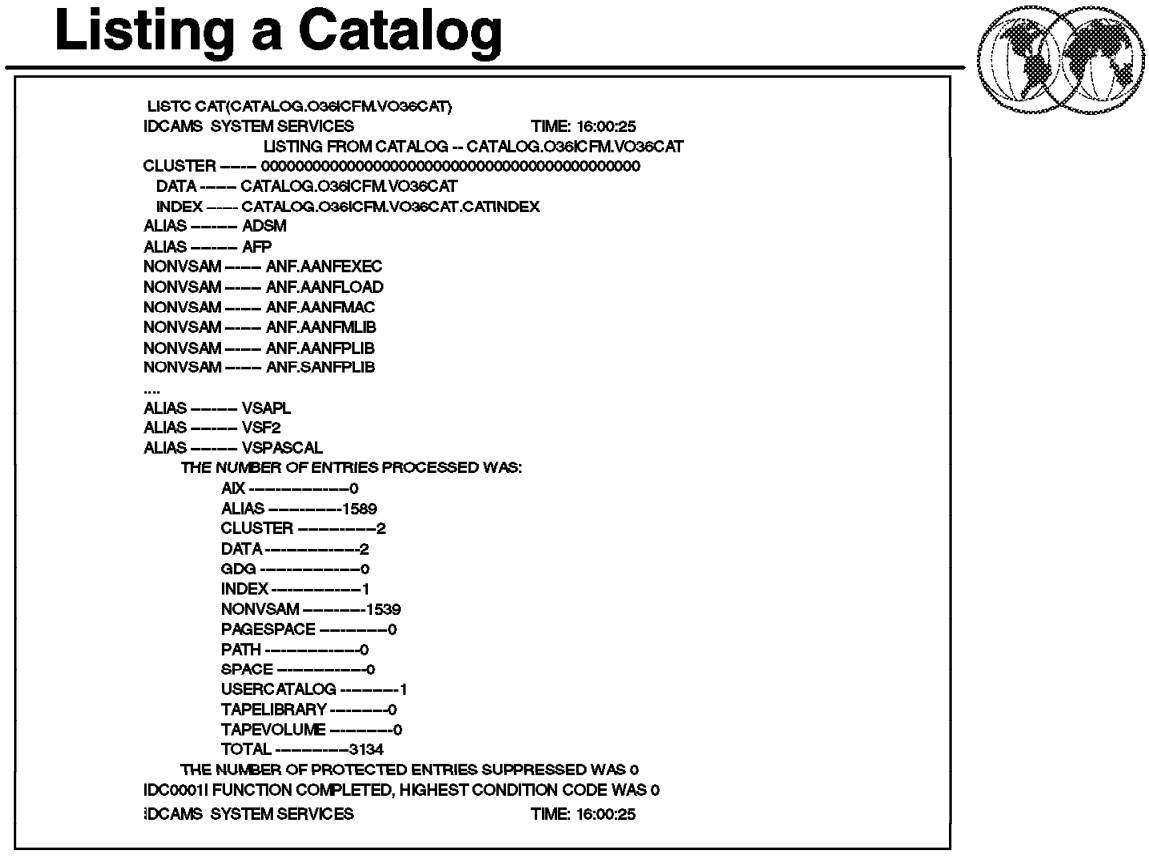


Figure 176. Listing a catalog

5.7 Requesting information from a catalog

You can list catalog records using the access method services LISTCAT command, or the ISMF line operator command CATLIST. CATLIST produces the same output as LISTCAT.

You can use the LISTCAT output to monitor VSAM data sets. The statistics and attributes listed can be used to help determine if you should reorganize, recreate, or otherwise alter a VSAM data set to improve performance or avoid problems.

This visual shows one example of such a listing.

5.7.1 LISTCAT examples

You can use LISTCAT to list the aliases associated with a catalog. Specify ALL with the catalog name in the ENTRIES parameter as shown in Figure 177.

```
//ALIAS    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
    LISTCAT ALL ENTRIES(SYS1.UCAT.VVOLUME)
/*
```

Figure 177. Example of using LISTCAT to list aliases for a catalog

You can use LISTCAT to list all USERCATALOGs connected to a Mastercatalog as shown in Figure 178 on page 260.

```
//LISTUCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
    LISTCAT CAT(MCAT.SYSCATLG) USERCATALOG)
/*
```

Figure 178. Example of using LISTCAT to list all user catalogs

Figure 179 shows an example of listing all information from a catalog.

```
//LISTCAT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
   LISTCAT CAT(CATALOG.ICF.VERALS)
 /*
```

Figure 179. Example of listing all information from a catalog

For more information about LISTCAT, see *DFSMS/MVS Access Method Services for ICF Catalogs*, ____-____; for details on using CATLIST, see *DFSMS/MVS Using ISMF*, ____-____.

5.7.2 Printing contents of a CATALOG and VVDS

You can print the contents of a BCS or VVDS with the PRINT command, but the only circumstance where it might be useful is when you need to determine which catalogs are connected to a VVDS. This might be necessary to determine which BCSs to specify in a DIAGNOSE command, or when you are recovering a volume.

Figure 180 shows an example of a VVDS print.

```
//PRNTVVDS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//VVDS     DD DSN=SYS1.VVDS.VSPOOL1,DISP=SHR,
//            UNIT=SYSDA,VOL=SER=SPOOL1,AMP=AMORG
//SYSIN    DD *
      PRINT INFILE(VVDS) COUNT(1)
/*
```

*

Figure 180. Example of a VVDS print

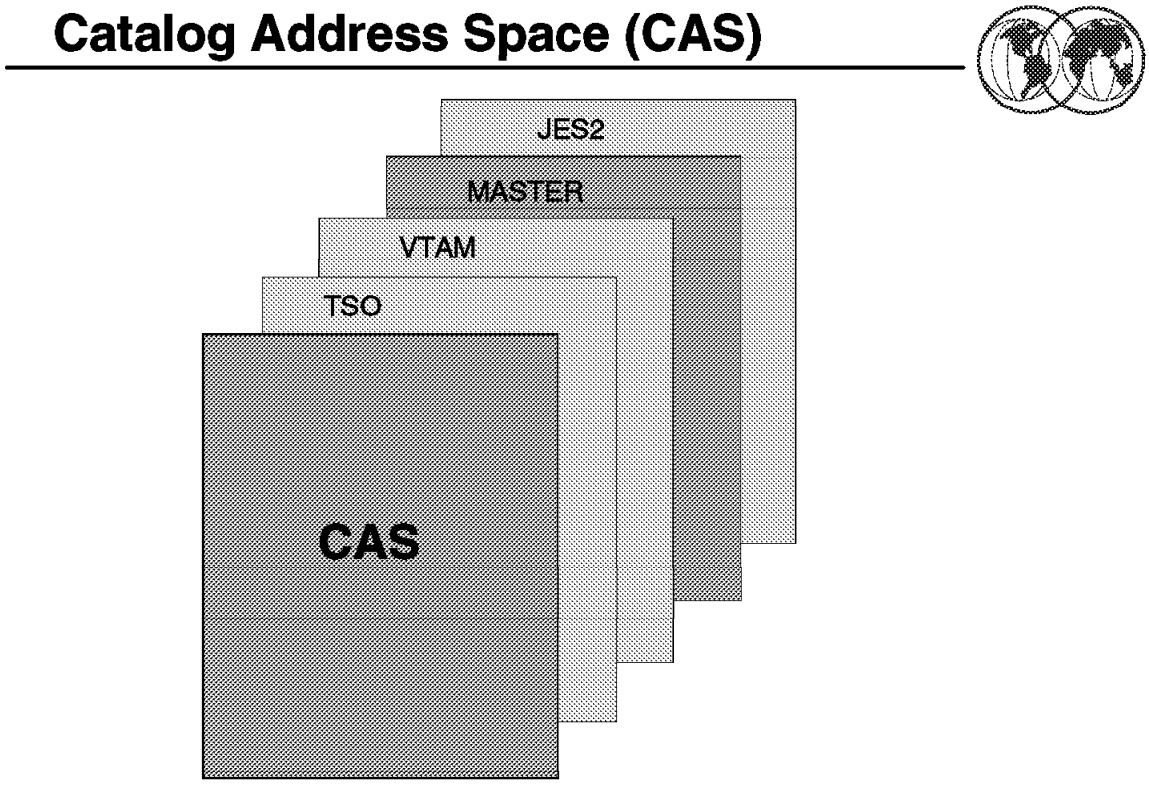


Figure 181. The catalog address space

5.8 The catalog address space

Catalog functions are performed in the *catalog address space* (CAS). Most catalog modules and control blocks are located in the catalog address space above the 16 MB virtual line. This reduces the required virtual storage in an user's private area needed to perform catalog functions.

During the initialization of an MVS system, all user catalog names identified in the Mastercat, their aliases, and their associated volume serial numbers are placed in tables in CAS. The number of lower limit CAS service tasks specified in the SYSCATxx member of SYS1.NUCLEUS (or the LOADxx member of SYS1.PARMLIB) is also created, and a table called the CRT keeps track of these service tasks. This number controls the multitask level within the CAS. The default value is X'3C'.

If you intend to use VLF as a cache for catalog records the general recommendation is to use VLF for Usercats and CAS for Mastercat.

Changes to the Mastercat are automatically reflected in the CAS tables. The information in the Mastercat is normally the same as the information in CAS. For shared catalogs, the CAS on all the sharing systems are updated, maintaining data integrity for your systems.

5.8.1 Restarting the catalog address space

Restarting the CAS should be considered only as a final option before IPLing a system. Never try restarting CAS unless an IPL is your only other option. A system failure caused by catalogs, or a CAS storage shortage due to FREEMAIN failures, might require you to use MODIFY CATALOG,RESTART to restart CAS in a new address space.

Never use RESTART to refresh catalog or VVDS control blocks or to change catalog characteristics. Restarting CAS is a drastic procedure, and if CAS cannot restart, you will have to IPL the system.

Note: It is possible that a catalog request, currently being processed, cannot be properly retried after being interrupted by these commands. Use these commands when all other means of correcting an ongoing catalog error have failed.

When you issue MODIFY CATALOG,RESTART, the CAS mother task is abended with abend code 81A, and any catalog requests in process at the time are redriven.

The restart of CAS in a new address space should be transparent to all users. However, even when all requests are redriven successfully and receive a return code of zero, the system might produce indicative dumps on the console, the system log, and on user job logs. There is no way to suppress these indicative dumps.

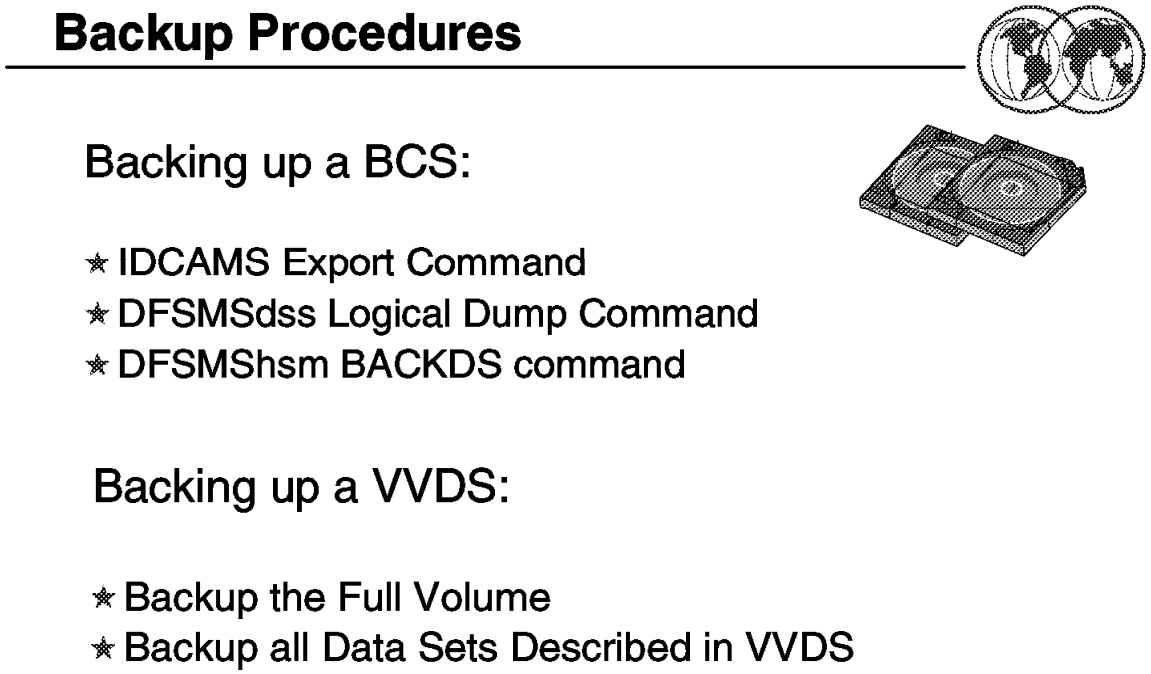


Figure 182. Backup procedures

5.9 Backup procedures

The two parts of an ICV catalog, the BCS and the VVDS, require different backup techniques: the BCS can be backed up like any other data set, whereas the VVDS can only be backed up as part of a volume dump. The entries in the VVDS and VTOC are backed up when the data sets they describe are exported with access method services, logically dumped with DFSMSdss, or backed up with DFSMShsm.

5.9.1 Backing up a BCS (Mastercat or Usercat)

You can use the access method services EXPORT command, the DFSMSdss logical DUMP command, or the DFSMShsm BACKDS command to back up a BCS. You can later recover the backup copies using the same utility used to create the backup; the access method services IMPORT command for exported copies; the DFSMSdss RESTORE command for logical dump copies; and the DFSMShsm RECOVER command for DFSMShsm backups.

The copy created by these utilities is a *portable* sequential data set that can be stored on a tape or direct access device, which can be of a different device type than the one containing the source catalog.

When these commands are used to back up a BCS, the aliases of the catalog are saved in the backup copy. The source catalog is not deleted, and remains as a fully functional catalog. The relationships between the BCS and VVDSs are unchanged.

You cannot permanently export a catalog by using the PERMANENT parameter of EXPORT. The TEMPORARY option is used even if you specify PERMANENT or allow it to default. PERMANENT specifies that the cluster or alternate index is to be deleted from the original system. Its storage space is freed. If its retention period has not yet expired, you must also code PURGE.

Note: You cannot use IDCAMS REPRO or other copying commands to create and recover BCS backups.

5.9.2 Backing up a VVDS

The VVDS should not be backed up as a data set to provide for recovery. To back up the VVDS, back up the volume containing the VVDS, or back up all data sets described in the VVDS (all VSAM and SMS-managed data sets). If the VVDS ever needs to be recovered, recover the entire volume, or all the data sets described in the VVDS.

You can use either DFSMSdss or DFSMShsm to back up and recover a volume or individual data sets on the volume.

5.9.3 Backing up a Mastercat

A Mastercat can be backed up as any other BCS. You should use EXPORT, DFSMSdss, or DFSMShsm for the backup. Another way to provide a backup for the Mastercat is to create an alternate Mastercat. For information on defining and using an alternate Mastercat, see *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

You should also make periodic volume dumps of the Mastercat′s volume. This dump can later be used by the stand-alone version of DFSMSdss to restore the Mastercat if you cannot access the volume from another system.

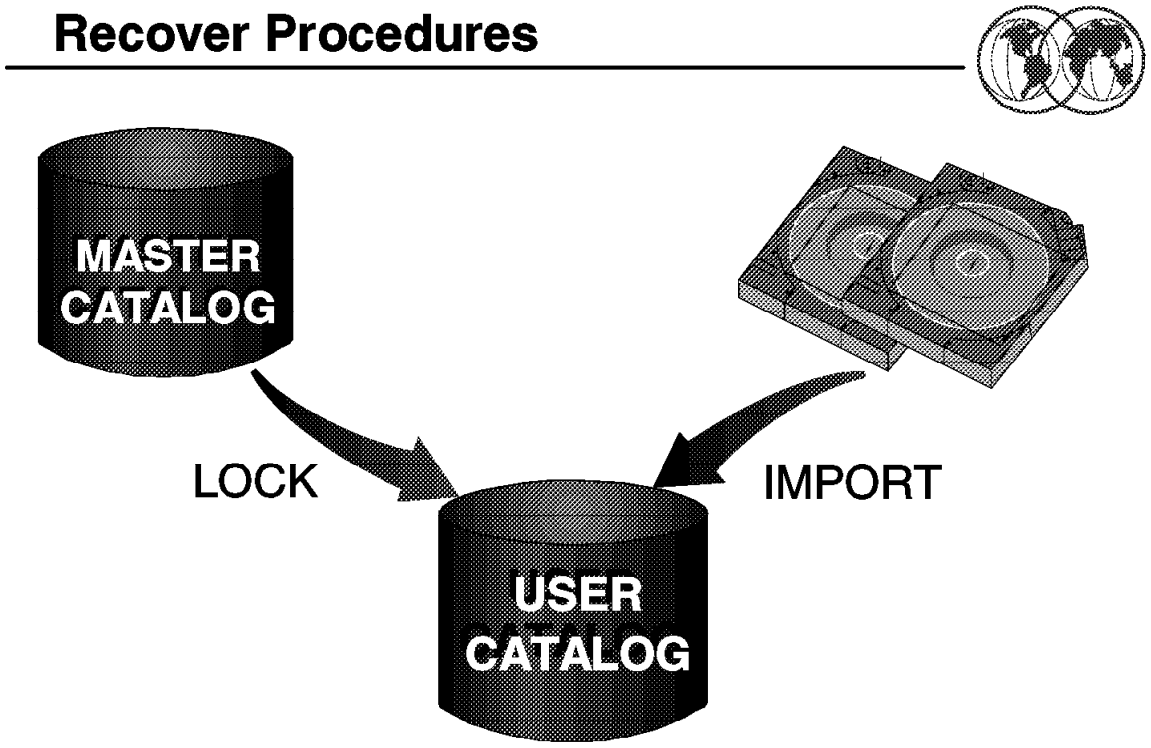


Figure 183. Recover procedures

5.10 Recovery procedures

Normally, a BCS is recovered separately from a VVDS. A VVDS usually does not need to be recovered, even if an associated BCS is recovered. However, if you need to recover a VVDS, and a BCS resides on the VVDS′s volume, you must recover the BCS as well. If possible, you should export the BCS before recovering the volume, and then recover the BCS from the exported copy. This ensures a current BCS.

Before recovering a BCS or VVDS, try to recover damaged records. If damaged records can be rebuilt, you can avoid a full recovery. BCS records can be recovered using the access method services DELETE and DEFINE commands with appropriate parameters. VVDS and VTOC records can be recovered using the DELETE command and by recovering the data sets on the volume.

You can recover a BCS that was backed up with the access method services EXPORT command, the DFSMSdss logical DUMP command, or the DFSMShsm BACKDS command or automatic backup. To recover the BCS, use the IDCAMS IMPORT command, the DFSMSdss RESTORE command, or the DFSMShsm RECOVER command.

When you recover a BCS using these commands, you do not need to delete and redefine the target catalog unless you want to change the catalog′s size or other characteristics, or unless the BCS is damaged in such a way as to prevent the usual recovery. The recovered catalog is reorganized when you use IMPORT or RECOVER, but not when you use RESTORE.

Aliases to the catalog can be defined if you use DFSMSdss, DFSMShsm, or if you specify ALIAS on the IMPORT command. If you have not deleted and redefined the catalog, all existing aliases are maintained, and any aliases defined in the backup copy are redefined if they are not already defined.

Lock the BCS before you try to recover a BCS. After you recover the catalog, update the BCS with any changes which have occurred since the last backup You can use the access method services DIAGNOSE command to identify certain unsynchronized entries.

When locking a catalog we recommend that you restrict access to a catalog when you are recovering it or when you are performing other maintenance procedures which involve redefining the catalog. If you do not restrict access to the catalog (by locking it, by terminating user sessions, or by another method), users might be able to update the catalog during recovery or maintenance and create a data integrity exposure. Locking the catalog eliminates the need to terminate user sessions during catalog recovery or maintenance.

In addition, you can only lock integrated user catalogs. You cannot lock a master catalog. While the catalog is locked, unauthorized requests to access the catalog fail with a return code indicating that the catalog is temporarily unavailable. Jobs entered with JCL fail with a JCL error. You cannot make JCL jobs wait until the catalog is unlocked. The catalog is also unavailable to any system that shares the catalog.

You can also use the Integrated Catalog Forward Recovery Utility (ICFRU) to recover a damaged catalog to a correct and current status. This utility uses system management facilities (SMF) records that record changes to the catalog, updating the catalog with changes made since the BCS was backed up. ICFRU can also be used with shared catalogs or master catalogs, if the master catalog is not being used as a master at the time. We recommend the use of ICFRU to avoid the loss of some catalog data even after the recovery.

For further information about recovery procedures, see *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

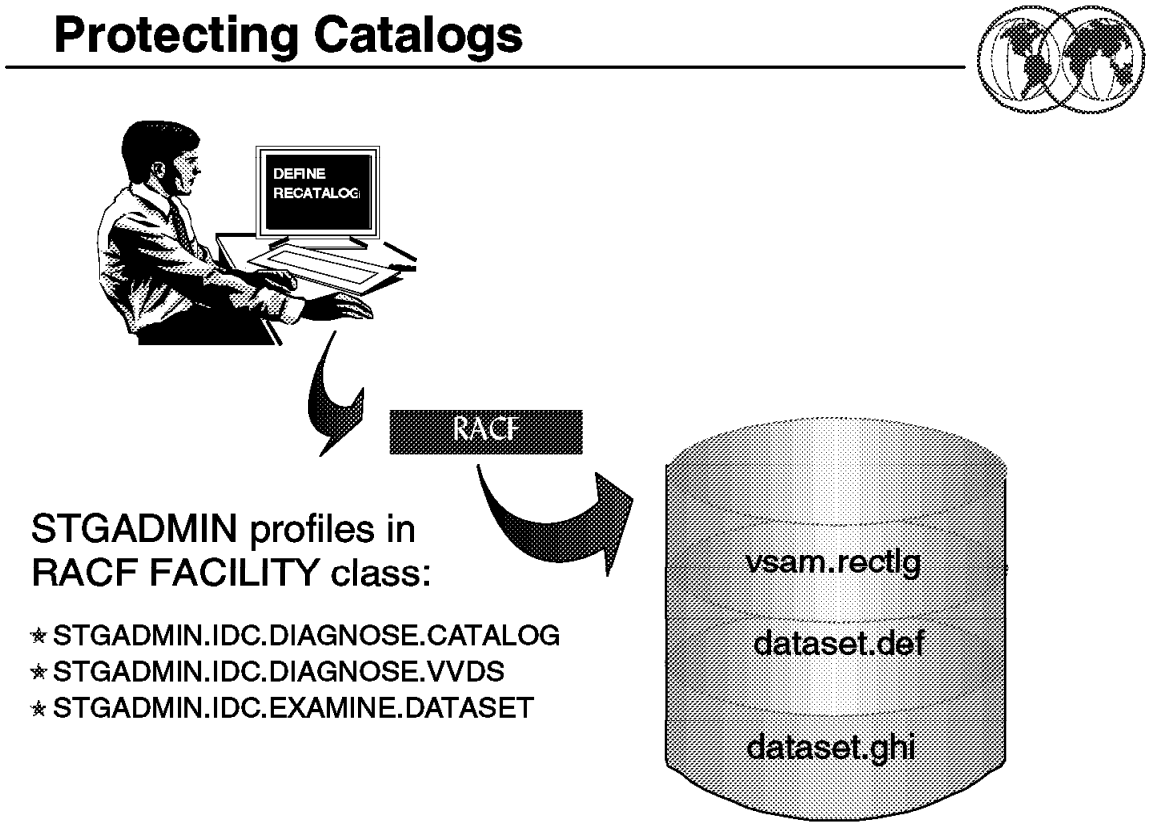


Figure 184. RACF

5.11 Protecting catalogs

The protection of data includes:

- Data security: the safety of data from theft or intentional destruction
- Data integrity: the safety of data from accidental loss or destruction

Data can be protected either indirectly, by preventing access to programs which can be used to modify data, or directly, by preventing access to the data itself. Catalogs and cataloged data sets can be protected in both ways.

To protect your catalogs and cataloged data, use the Resource Access Control Facility (RACF) or similar product.

5.11.1 RACF authorization checking

RACF provides a software access control measure you can use in addition to or instead of VSAM passwords. RACF protection and password protection can coexist for the same data set.

To open a catalog as a data set, you must have ALTER authority and APF authorization. When defining an SMS-managed data set, the system only checks to make sure the user has authority to the data set name and SMS classes and groups. The system selects the appropriate catalog, without checking the user's authority to the catalog. You can define a data set if you have ALTER or OPERATIONS authority to the applicable data set profile.

Deleting any type of RACF-protected entry from a RACF-protected catalog requires ALTER authorization to the catalog or to the data set profile protecting the entry being deleted. If a non-VSAM data set is SMS-managed, RACF does not check for DASDVOL authority. If a non-VSAM, non-SMS-managed data set is being scratched, DASDVOL authority is also checked.

For ALTER RENAME, the user is required to have the following two types of authority:

- ALTER authority to either the data set or the catalog
- ALTER authority to the new name (generic profile) or CREATE authority to the group

Be sure that RACF profiles are correct after you use REPRO MERGECAT or CNVTCAT on a catalog that uses RACF profiles. If the target and source catalogs are on the same volume, the RACF profiles remain unchanged.

Non-VSAM tape data sets defined in an integrated catalog facility catalog can be protected by:

- Controlling access to the tape volumes
- Controlling access to the individual data sets on the tape volumes

5.11.2 Profiles

To control the ability to perform functions associated with storage management, define profiles in the FACILITY class whose profile names begin with STGADMIN (storage administration). For a complete list of STGADMIN profiles, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920. Examples of some profiles are:

```
STGADMIN.IDC.DIAGNOSE.CATALOG
STGADMIN.IDC.DIAGNOSE.VVDS
STGADMIN.IDC.EXAMINE.DATASET
```

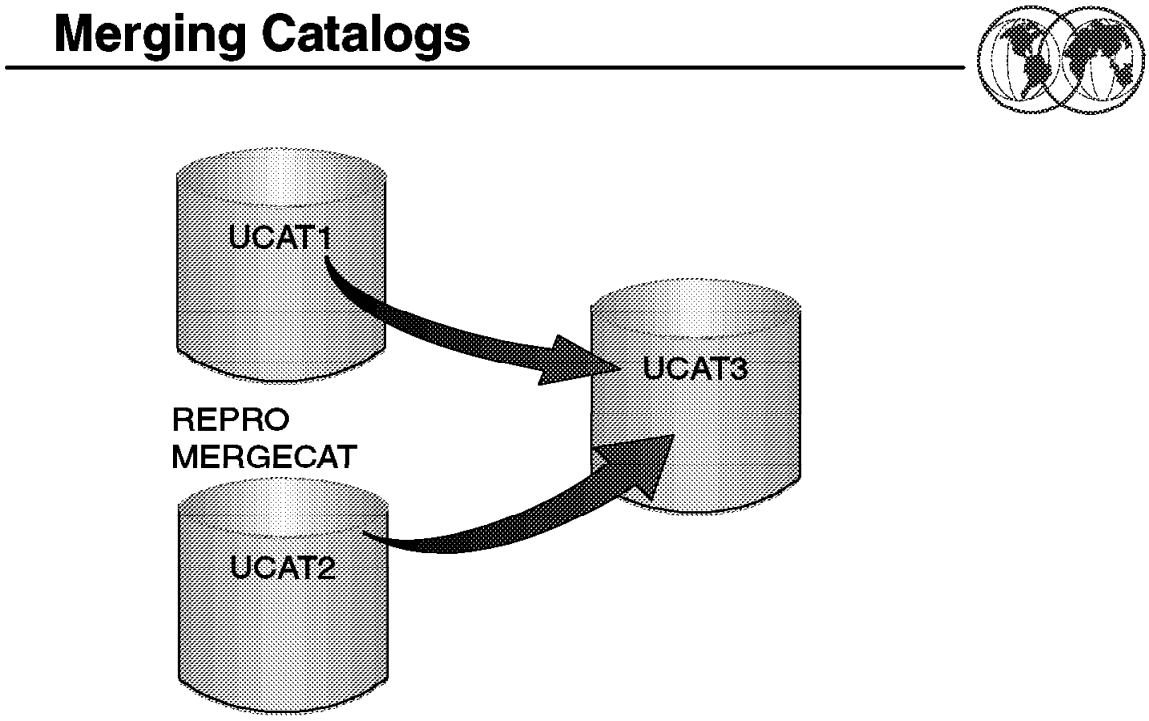


Figure 185. Merging catalogs

5.12 Merging catalogs

You might find it beneficial to merge catalogs, if you have many small or seldom-used catalogs. An excessive number of catalogs can complicate recovery procedures and waste resources such as CAS storage, tape mounts for backups, and system time performing backups.

Merging catalogs is accomplished in much the same way as splitting catalogs. The only difference between splitting catalogs and merging them is that in merging, you want all the entries in a catalog to be moved to different catalog, so that you can delete the obsolete catalog.

The following steps should be followed to merge two integrated catalog facility catalogs:

1. Use ALTER LOCK to lock both catalogs.
2. Use LISTCAT to list the aliases for the catalog you intend to delete after the merger. Use the OUTFILE parameter to delete a data set to contain the output listing.
3. Use EXAMINE and DIAGNOSE to ensure that the catalogs are error-free. Fix any errors indicated.
4. Use REPRO MERGECAT without specifying the ENTRIES or LEVEL parameter. The OUTDATASET parameter specifies the catalog that you are keeping after the two catalogs are merged. This step can take a long time to complete.
5. Use the listing created in step 2 to create a sequence of DELETE ALIAS and DEFINE ALIAS commands to delete the aliases of the obsolete catalog, and to redefine the aliases as aliases of the catalog you are keeping. If this step fails for any reason, see “Recovering from a REPRO

MERGECAT Failure" in topic 4.2.3 of *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

6. Use DELETE USERCATALOG to delete the obsolete catalog. Specify RECOVERY on the DELETE command.
7. If your catalog is shared, run the EXPORT DISCONNECT command on each shared system to remove unwanted user catalog connector entries.
8. Use ALTER UNLOCK to unlock the remaining catalog.

You can also merge entries from one tape volume catalog to another using REPRO MERGECAT. REPRO retrieves tape library or tape volume entries and redefines them in a target tape volume catalog. In this case, VOLUMEENTRIES needs to be used to correctly filter the appropriate entries. The LEVEL parameter is not allowed when merging tape volume catalogs.

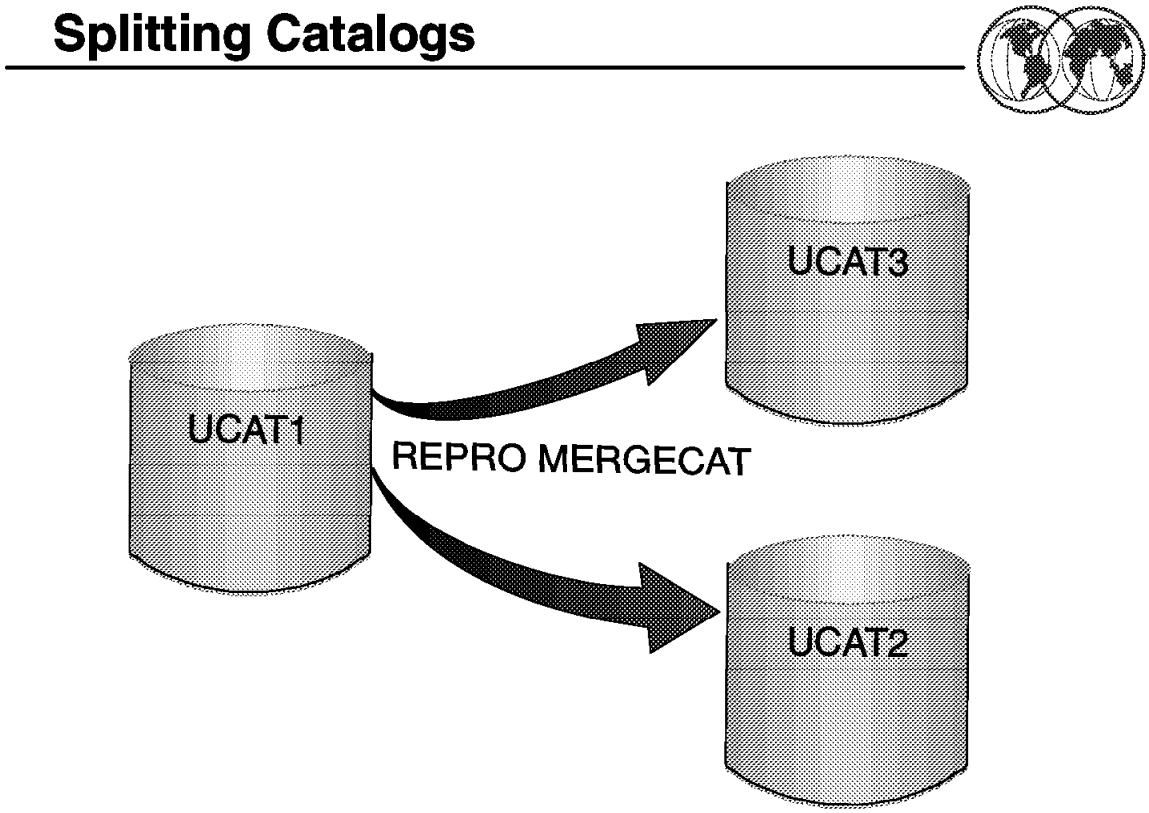


Figure 186. Splitting catalogs

5.13 Splitting a catalog

You can split a catalog to create two catalogs or to move a group of catalog entries, if you determine that a catalog is either unacceptably large or that it contains too many entries for critical data sets.

If the catalog is unacceptably large (a catalog failure would leave too many entries inaccessible), then you can split the catalog into two catalogs. If the catalog is of an acceptable size but contains entries for too many critical data sets, then you can simply move entries from one catalog to another.

To split a catalog or move a group of entries, use the access method services REPRO MERGECAT command. The following steps should be followed to split a catalog or to move a group of entries:

1. Use ALTER LOCK to lock the catalog. If you are moving entries to an existing catalog, lock it as well.
2. If you are splitting a catalog, define a new catalog with DEFINE USERCATALOG LOCK.
3. Use LISTCAT to obtain a listing of the catalog aliases which you are moving to the new catalog. Use the OUTFILE parameter to define a data set to contain the output listing.
4. Use EXAMINE and DIAGNOSE to ensure that the catalogs are error-free. Fix any errors indicated.
5. Use REPRO MERGECAT to split the catalog or move the group of entries. When splitting a catalog, the OUTDATASET parameter specifies the catalog created in step 2. When moving a group of entries, the OUTDATASET parameter specifies the catalog which is to receive the entries. This step can take a long time to complete.

Use the ENTRIES or LEVEL parameters to specify which catalog entries are to be removed from the source catalog and placed in the catalog specified in OUTDATASET.

If this step fails for any reason, see “Recovering from a MERGECAT Failure” in topic 4.2.3 of *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

6. Use the listing created in step 3 to create a sequence of DELETE ALIAS and DEFINE ALIAS commands for each alias. These commands delete the alias from the original catalog, and redefine them as aliases for the catalog which now contains entries belonging to that alias name.

   The DELETE ALIAS/DEFINE ALIAS sequence must be run on each system that shares the changed catalogs.

7. Unlock both catalogs using ALTER UNLOCK.

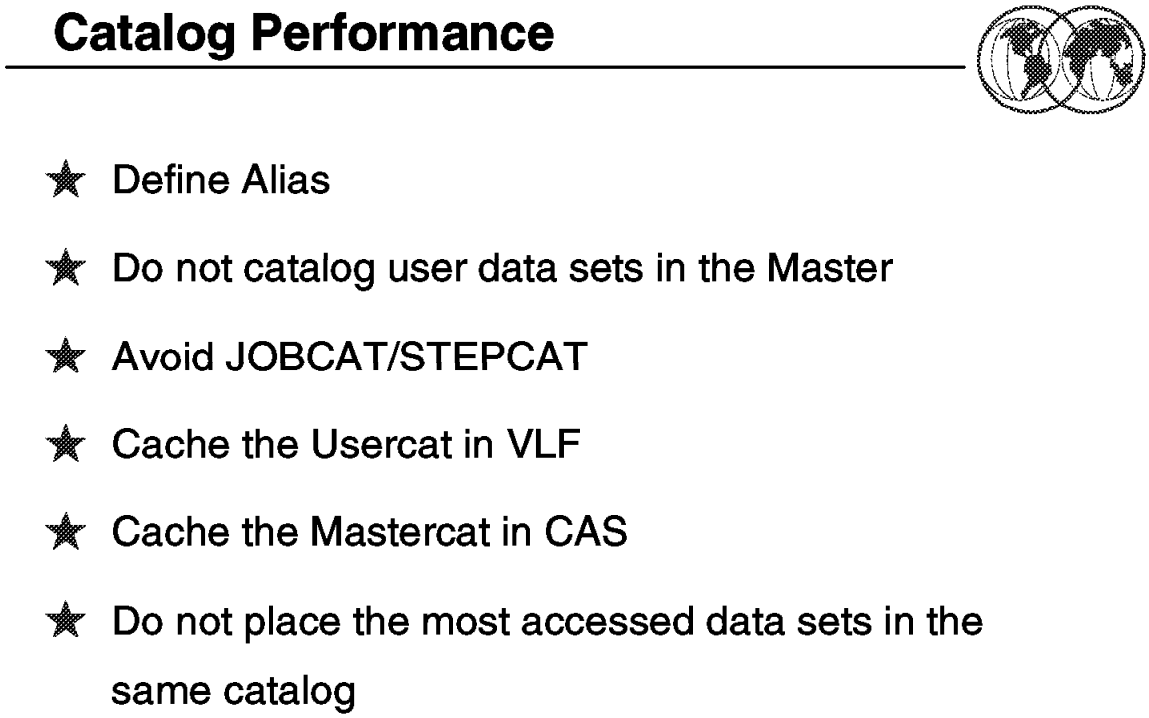


Figure 187. Catalog performance

5.14 Catalog performance

Performance should not be your main consideration when you define catalogs. It is more important to create a catalog configuration which allows easy recovery of damaged catalogs with the least amount of system disruption. However, there are several options you can choose to improve catalog performance without affecting the recoverability of a catalog.

5.14.1 Factors affecting catalog performance

The main factors affecting catalog performance are the amount of I/O required for the catalog and the subsequent amount of time it takes to perform the I/O. These factors can be reduced by caching catalogs in special caches used only by catalogs. If a catalog is cached, the reduced I/O to the catalog reduces the effect of other factors on catalog performance.

If the Mastercat only contains entries for catalogs, catalog aliases, and system data sets, the entire Mastercat is read into CAS virtual storage during system initialization. Because the Mastercat, if properly used, is rarely updated, the performance of the Mastercat is not appreciably affected by I/O requirements.

5.14.2 Eliminating JOBCAT and STEPCAT DD

Another good reason for not using STEPCAT and JOBCAT is because they have poor performance.

5.14.3 Caching catalogs

The simplest method of improving catalog performance is to use cache to maintain catalog records within CAS private area address space or VLF data space.

Two kinds of cache are available exclusively for catalogs. The in-storage catalog (ISC) cache is contained within the catalog address space (CAS). The catalog data space cache (CDSC) is separate from CAS and uses the MVS VLF component which stores the cached records in a data space. Both types of cache are optional, and each can be cancelled and restarted without an IPL.

Although you can use both types of catalog cache, you cannot cache a single catalog in both types of cache simultaneously. You must decide which catalogs benefit the most from each type of cache.

Catalog records are cached in the ISC or CDSC under the following conditions:

- For Mastercat, all records accessed sequentially or by key are cached except for alias records. Alias records are kept in a separate table in main storage.
- For Usercats, only records accessed by key are cached, that is by data set name—which is the normal type of access.
- For each catalog, the records are cached in the CDSC if you have indicated the catalog is to use CDSC. Otherwise, the records are cached in the ISC (Usercat included) unless you have stopped the ISC for the catalog. If you stop both the CDSC and the ISC for a catalog, then records are not cached.

In-storage cache (ISC)

The in-storage catalog cache resides in the private area of the CAS. It is the default catalog cache. Each Usercat cached in ISC is given a fixed amount of space for cached records. When a Usercat reaches its allotted space in the ISC, the least recently used record is removed from the ISC to make room for the new entry.

Catalogs which are not frequently updated use the ISC most effectively. The Mastercat is ideally suited for ISC because master catalog records are infrequently updated. The performance of the ISC is affected if the catalog is shared with another system.

Mastercat, unlike Usercats, are not limited to a set amount of CAS virtual storage. All eligible records in the Mastercat are cached in the ISC as they are read.

Thus, you should keep the number of entries in the master catalog to a minimum, so that the ISC for the master does not use an excessive amount of main storage.

Since ISC is the default catalog cache, catalogs are cached in the ISC unless you specify that the catalog is to use CDSC, or unless you use the `MODIFY CATALOG` operator command to remove the catalog from the ISC.

Catalog data space cache (CDSC)

CDSC resides in a data space which you define with the COFVLFxx member of SYS1.PARMLIB. The CDSC uses the virtual look-aside facility (VLF), which can be started by using the `START VLF` operator command. In the COFVLFxx you should declare:

- NAME(IGGCAS)
- EMAJ(catname)...: for the names of the catalogs to be cached
- MAXVIRT({ 4096|size}): specifying the maximum virtual storage that VLF can use to cache catalog records.

You can add catalogs to the CDSC only by changing the COFVLFxx member to specify the catalogs, stopping VLF, then starting VLF. Because this interrupts servicing on the existing CDSC, catalog performance might be degraded for a while.

The CDSC can concurrently be used for any catalog which is not using the ISC. A single catalog cannot use both the CDSC and the ISC. Unlike the ISC, catalogs cached in the CDSC are not limited to a specific amount of data space virtual storage. A catalog caches records until no space is left in the data space cache. Once the data space cache is full, the space occupied by the record least used is removed to make room for new records.

Monitoring the CAS

```
**CAS*************************************************************
**CAS**  CATALOG ADDRESS SPACE REPORT OUTPUT   **
**CAS*************************************************************
**CAS**  CATALOG COMPONENT LEVEL      = HDZ11C0   **
**CAS**  CATALOG ADDRESS SPACE ASN    = 0012       **
**CAS**  SERVICE TASK UPPER LIMIT     = 00C8       **
**CAS**  SERVICE TASK LOWER LIMIT     = 003C       **
**CAS**  HIGHEST # SERVICE TASKS      = 0006       **
**CAS**  CURRENT # SERVICE TASKS      = 0006       **
**CAS**  MAXIMUM # OPEN CATALOGS      = 0032       **
**CAS**  ALIAS TABLE AVAILABLE        = YES        **
**CAS**  ALIAS LEVELS SPECIFIED       = 1          **
**CAS**  CRT TABLE SLOT ROTATION      = OFF        **
**CAS**  SYS% TO SYS1 CONVERSION      = OFF        **
**CAS**  CAS MOTHER TASK              = 00A5BE88   **
**CAS**  CAS MODIFY TASK              = 00A5B5B0   **
**CAS**  CAS ANALYSIS TASK            = 00A5B950   **
**CAS**  CAS ALLOCATION TASK          = 00A5BCF0   **
**CAS**  VOLCAT HI-LEVEL QUALIFIER    = SYS1       **
**CAS**  DELETE UCAT/VVDS WARNING     = ON         **
**CAS*************************************************************
```

Figure 188. Monitoring the CAS

5.14.4 Monitoring the CAS

Using the MODIFY CATALOG,REPORT command, you can obtain general information about the catalog address space. This information can be used to evaluate your current catalog environment setup. If you determine that your current setup is inadequate, you can change it with another MODIFY CATALOG command, or by changing the SYSCATxx member of SYS1.NUCLEUS.

The visual shows an example of the output of an unqualified MODIFY CATALOG,REPORT command.

There is information about the service tasks available to process catalog requests. In this example, you can see that the service task lower limit is adequate for the current number of tasks. The service task upper limit defaults to X′B4′, and can be changed by the MODIFY CATALOG,TASKMAX command. The highest number of service tasks is equal to the highest value the current number of tasks field has reached.

Catalog management creates tasks as necessary as the current number of tasks exceeds the service task lower limit. As catalog requests subside, the number of tasks attached and available for processing requests is reduced until the lower limit is reached.

The maximum number of open catalogs is the value set by the MODIFY CATALOG,CATMAX command. This is the maximum number of catalogs, in hexadecimal, which might be open to CAS simultaneously. If the maximum is reached, catalog management closes the least recently used catalog before opening another catalog. Normally, a catalog remains open once it has be opened.

The alias table available entry indicates whether there is a problem with the catalog alias table. This field should always say YES. If it says NO try restarting the catalog address space. Performance is affected when catalog management does not have catalog aliases in the catalog alias table.

For more details on the F CATALOG,REPORT output, refer to *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914.

Monitoring the CAS Performance

```
IEC359I CATALOG PERFORMANCE REPORT
*CAS*************************************************
* -----CATALOG EVENT----  --COUNT--  ---AVERAGE---  *
* Entries to Catalog            409    18.617  SEC   *
* BCS ENQ Shr                   413     0.458 MSEC   *
* BCS ENQ Excl                    9     0.341 MSEC   *
* BCS DEQ                       442     0.459 MSEC   *
* VVDS RESERVE Cl                24   132.823 MSEC   *
* VVDS DEQ Cl                    24     1.141 MSEC   *
* VVDS RESERVE Shr               93   310.896 MSEC   *
* VVDS RESERVE Excl               8     0.970 MSEC   *
* VVDS DEQ                      101     1.193 MSEC   *
* SPHERE ENQ Excl                16     0.304 MSEC   *
* SPHERE DEQ                     16     0.303 MSEC   *
* RPL ENQ                         3     0.210 MSEC   *
* RPL DEQ                         3     0.244 MSEC   *
* BCS Get                     2,074     2.468 MSEC   *
* BCS Put                         5    21.764 MSEC   *
* BCS Erase                       3    34.630 MSEC   *
* VVDS I/O                      130    13.194 MSEC   *
* VLF Define Major                1    13.861 MSEC   *
```

Figure 189. Monitoring the CAS performance

5.14.5 Monitoring the CAS performance

This visual shows the partial report obtained by the execution of the MVS command:

```
MODIFY CATALOG,REPORT,PERFORMANCE
```

This command can be useful in identifying performance problems that you suspect are related to catalog processing. For example, if the average time for ENQs that is shown in the report seems excessive, it might indicate some problems in the GRS configuration, or parameter specifications. High I/O times might indicate problems with:

- Channel or device load
- Volumes that are suffering a high number of I/O errors
- Volumes that have excessively high RESERVE rates or long RESERVE durations

Monitoring the CDSC Performance

```
IEC359I CATALOG CACHE REPORT
*CAS*************************************************************************************
*   HIT% -RECORDS- -SEARCHES --FOUND-- -DELETES- -SHR UPD- --PURGE
*
* SYS1.MVSRES.MASTCAT                                    (ISC)
*   42%       1,578          679       291        13          0          3
* SYS1.PROD.UCAT                                         (VLF)
*   37%       2,304        1,014       372        27         42          0
*                                        *

*CAS*************************************************************************************
```

Figure 190. Monitoring the CDSC performance

5.14.6 Monitoring the CDSC performance

In order to evaluate the catalog data space cache, use the command:

```
MODIFY CATALOG,REPORT,CACHE
```

You can use this command to evaluate the cache performance for a specified catalog, or for all catalogs that are currently being cached. The numbers shown in this report are in decimal.

When you evaluate the cache performance for a catalog, you need to consider how long the catalog has been using the cache. If the cache has only been available for a catalog for an hour, the hit ratio will likely be low. However, if the catalog has been using the cache for a week, expect a good hit ratio. The hit ratio is an indication of cache usage while the cache is available. The values do not accumulate between performing IPLs, stopping and starting VLF, or restarting the catalog address space.

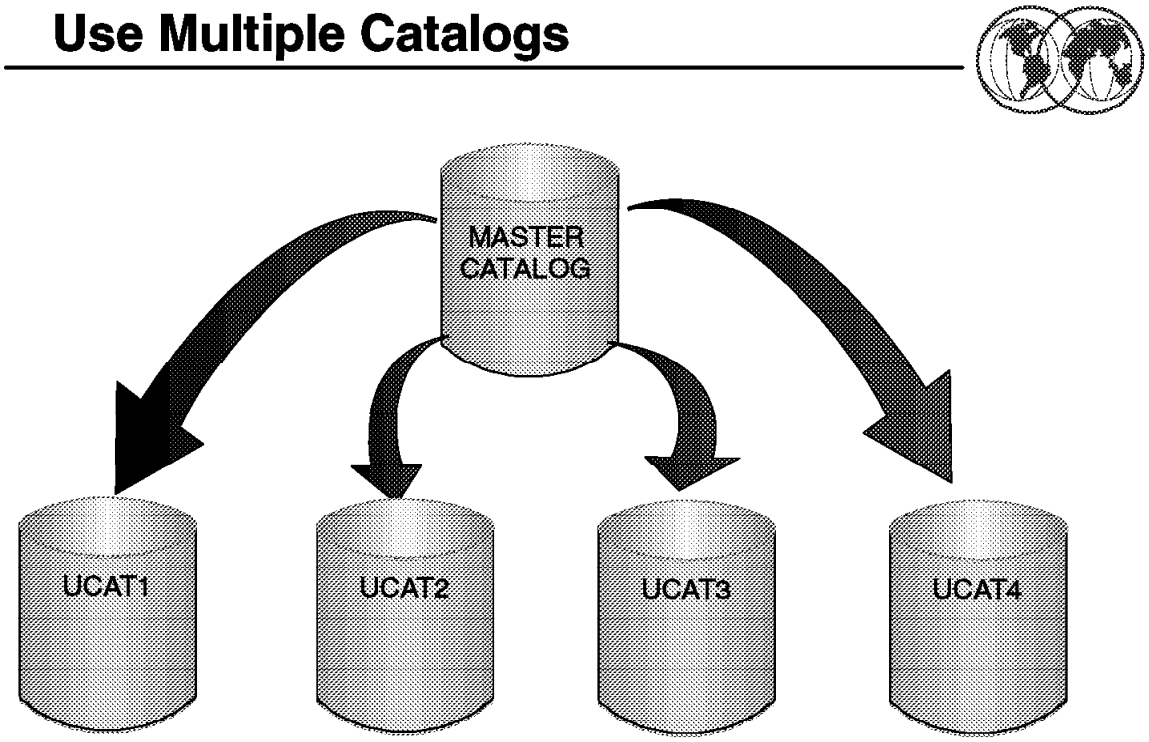


Figure 191. Use multiple catalogs

5.15 Using multiple catalogs

Multiple catalogs on multiple volumes may perform better than fewer catalogs on fewer volumes. This is because of the interference between requests to the same catalog; for example, a single shared catalog being locked out by another system in the sysplex. This situation can occur if some other application issues a RESERVE against the volume that has nothing to do with catalog processing. Another reason can be that there is more competition to use the available volumes, and thus more I/O can be in progress concurrently.

If you are not using VLF for caching, there is a limited amount of cache storage assigned to each catalog in CAS. Large catalogs may suffer from this fixed limit and receive less benefit from non-VLF caching than an environment with more smaller catalogs.

Multiple catalogs may reduce the impact of the loss of a catalog by:

- Reducing the time necessary to recreate any given catalog
- Allowing multiple catalog recovery jobs to be in process at the same time

Recovery from a pack failure is dependent on the total amount of catalog information on a volume—regardless of whether this information is stored in one or many catalogs

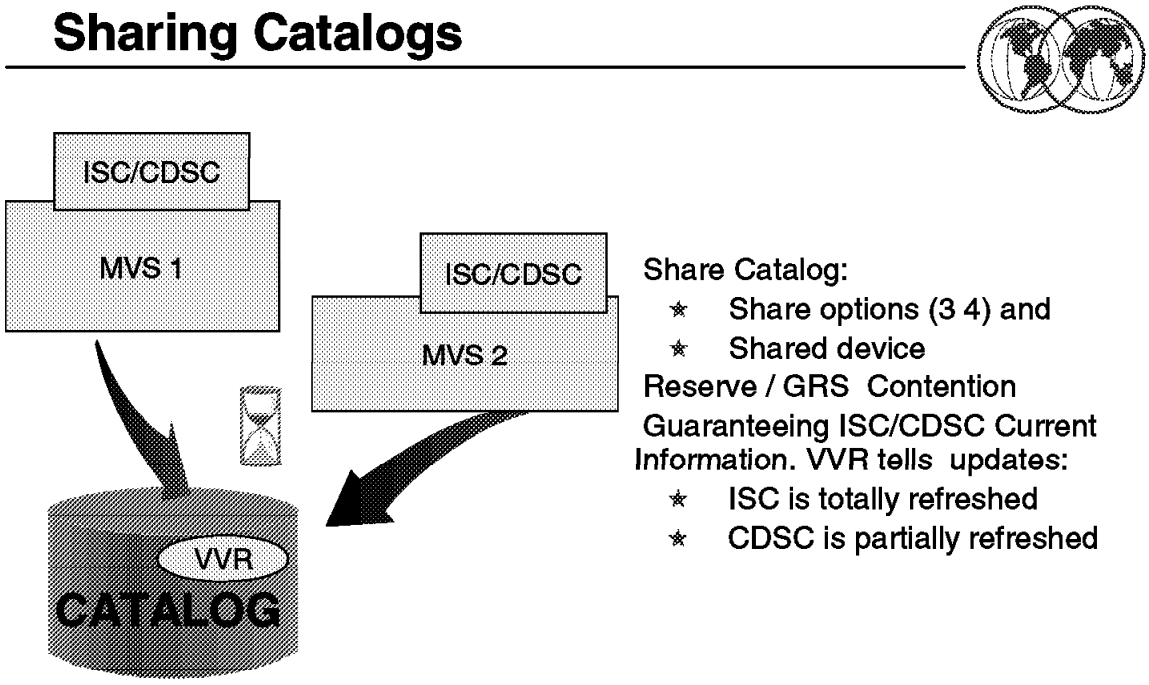


Figure 192. Sharing catalogs

5.15.1 Sharing catalogs

If a BCS catalog is defined with share options (3 4), and if it resides on a shared device, catalog management considers the catalog a shared catalog, including the correspondent VVDSs.

If a catalog is not really shared with another system, move the catalog an unshared device or alter its share options to (3 3). To prevent potential catalog damage, never place a catalog with share options (3 3) on a shared device.

The volume containing the VVDS is reserved before the I/O is performed.

5.15.1.1 Guaranteeing the ISC/CDSC current information

Before each physical access to a shared catalog, special I/O checking is performed to ensure that the ISC or CDSC contains current information. Checking also ensures that the access method control blocks for the catalog are updated in the event the catalog has been extended, or otherwise altered from another system. This checking maintains data integrity. It also affects performance because in order to keep this integrity, for every catalog access, a special VVR in the shared catalog must be read before using the ISC or CDSC version of the BCS record. This access implies a DASD reserve and I/O operations.

Changes to shared catalogs are handled differently depending on whether the catalog uses the ISC or the CDSC:

- If a catalog uses the ISC and a sharing system updates a record as indicated by the special VVR—any record, even if the record is not cached in this system′s ISC—catalog management releases the entire ISC for the catalog and creates a new ISC for the catalog. Individual records changed by a sharing system are not identified and updated for ISC catalogs.
- The CDSC, however, can identify individual records which a sharing system has updated by executing I/O operations. The special VVR contains a wrap table with the keys of the 90 most recent records updated/added/deleted. For every catalog record read, the catalog VVR must be read. If the BCS records cached in CDSC on that system are not changed, they are still valid. If the table of updated records in the VVR has wrapped since the last time the catalog was accessed by this system, the complete CDSC cache is purged for this catalog. If not, only those records indicated in VVR are discarded from CDSC cache. This requires I/O to VVDS for each BCS I/O. It also generates Reserves or GRS traffic.

  Thus, when a sharing system updates a record, the CDSC space used by the catalog is not necessarily totally released.

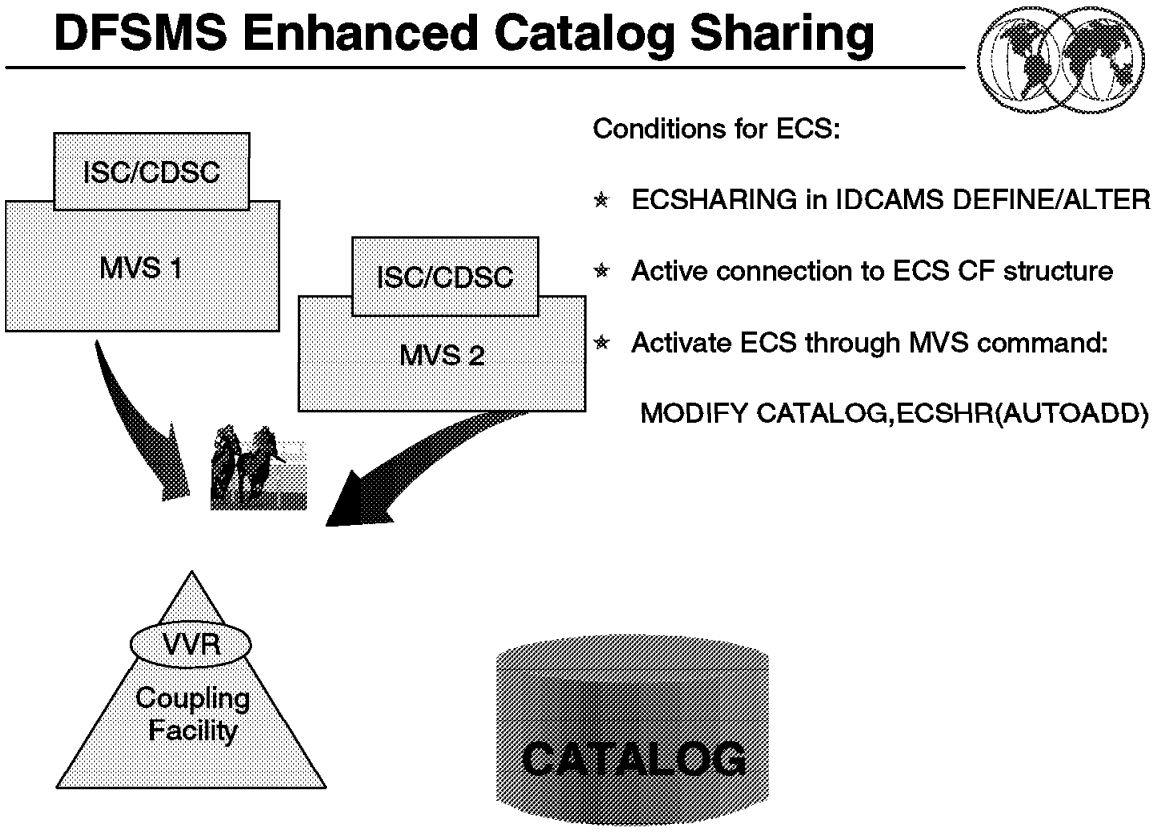


Figure 193. DFSMS enhanced catalog sharing

5.15.2 DFSMS enhanced catalog sharing

If the catalog has been defined to support DFSMS/MVS 1.5 Enhanced Catalog Sharing (ECS) by the ECSHARING attribute, most of the overhead associated with shared catalog is eliminated.

ECS uses a cache Coupling Facility structure to keep the special VVR. Also the Coupling Facility structure (as defined in CFRM) keeps a copy of the updated records. A new ENQ is used to indicate that a catalog is being used in ECS mode. As you can see, there is no more I/O to read the catalog VVR in order to verify the updates. In addition, the eventual modifications are also kept in the Coupling Facility structure, thus avoiding more I/O.

A new attribute, ECSHARING, is available on the IDCAMS DEFINE and ALTER commands. Setting this attribute on makes a catalog eligible for sharing using the ECS protocol, as opposed to the VVDS/VVR protocol. However, catalog management does not actually use the ECS protocol unless:

- There is an active connection to the ECS cache structure, and
- ECS mode has been activated by either the MODIFY CATALOG, ECSHR(AUTOADD) command or the MODIFY CATALOG, ECSHR(ADD...) command.

ECS provides saves about 50 percent in elapsed time and an enormous reduction in ENQ/Reserves.

Appendix A. OS/390 installation and customization

This appendix shows the SYS1.PARMLIB members and their functions.

A.1 SYS1.PARMLIB members

Table 5 (Page 1 of 3). Overview of parmlib members

Member	Description
ADYSETxx	Parameters that control dump analysis and elimination (DAE) processing.
ALLOCxx	Parameters that control installation defaults for allocation values.
APPCPMxx	Parameters that define or modify the APPC/MVS configuration.
ASCHPMxx	Parameters that define scheduling information for the ASCH transaction scheduler.
BLSCECT	Parameters that control dump formatting performed by IPCS, and by the system (through the ABEND and SNAP macros).
BLSCUSER	Parameters that specify IPCS customization.
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.
CLOCKxx	Parameters that control operator prompting to set the TOD clock, specifying the difference between the local time and GMT, and ETR usage.
CNGRPxx	Parameters that define console groups as candidates for switch selection if a console fails.
CNLcccxx	Defines how translated messages are to be displayed at your installation.
COFDLFxx	Allows a program to store DLF objects that can be shared by many jobs in virtual storage managed by Hiperbatch.
COFVLFxx	Allows an authorized program to store named objects in virtual storage managed by VLF.
COMMNDxx	Commands to be issued by the control program immediately after initialization. JES commands may not be included.
CONFIGxx	Allows the installation to define a standard configuration that is compared with the current configuration and to reconfigure processors, storage, and channel paths.
CONSOLxx	Parameters to define an installation's console configuration, initialization values for communications tasks, 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 and designate the alternate console group for hardcopy recovery.
COUPLExx	Describes the systems complex (sysplex) environment for the system.
CSVLLAxx	Allows an installation to list the entry point name or LNKLST libraries that can be refreshed by the 'MODIFY LLA, UPDATE=xx' command.
CSVRTLxx	Defines the run-time library services (RTLS) configuration. CSVRTLxx can be used to specify names of libraries to be managed, as well as storage limits for caching modules from the libraries.
CTncccxx	Specifies component trace options.
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.
DIAGxx	Contains diagnostic commands that control the common storage tracking and GETMAIN/FREEMAIN/STORAGE (GFS) trace functions.
EXITxx	Allows an installation to specify installation exits for allocation decisions.

Table 5 (Page 2 of 3). Overview of parmlib members	
Member	**Description**
EXSPATxx	Allows an installation to specify actions taken to recover from excessive spin conditions without operator involvement.
GRSCNFxx:	Configuration parameters for systems that are to share resources in a global resource serialization complex.
GRSRNLxx	Resource name lists (RNLs) that the system uses when a global resource serialization complex is active.
GTFPARM	Parameters to control GTF.
IEAABD00:	Default parameters for an ABEND dump when a SYSABEND DD statement has been specified.
IEAAPFxx	Names of authorized program libraries.
IEAAPPxx	Names of authorized installation-written I/O appendage routines.
IEACMD00	IBM-supplied commands that are processed during system initialization.
IEADMCxx	Parameters for DUMP command.
IEADMP00	Default parameters for an ABEND dump when SYSUDUMP DD statement has been specified.
IEADMR00	Default parameters for an ABEND dump when SYSMDUMP DD statement has been specified.
IEAFIXxx	Names of modules to be fixed in central storage for the duration of the IPL.
IEAICSxx	Parameters of an installation control specification that associate units of work (transactions) with performance groups. Performance groups are used by the system resources manager to control and report on transactions.
IEAIPSxx	Parameters of an installation performance specification that control the Workload Manager of the System Resources Manager.
IEALPAxx	Names of reenterable modules that are to be loaded as a temporary extension to the PLPA.
IEAOPTxx	Parameters that control resource and workload management algorithms in the System Resources Manager.
IEAPAKxx	“Pack List” names of groups of modules in the LPALST concatenation that the system will load between page boundaries to minimize page faults.
IEASLPxx	Contains valid SLIP commands.
IEASVCxx	Allows the installation to define its own SVCs in the SVC table.
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.
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.
IECIOSxx:	Parameters that control missing interrupt handler (MIH) intervals and update hot I/O detection table (HIDT) values.
IEFSSNxx	Parameters that identify what subsystems are to be initialized.
IFAPRDxx	Parameters that define a product enablement policy.
IGDDFPKG	One or more statements that specify which DFSMS/MVS functional components and separately orderable features are licensed for use on a particular MVS system or across a sysplex.
IGDSMSxx	Initialize the Storage Management Subsystem (SMS) and specify the names of the active control data set (ACDS) and the communications data set (COMMDS).
IKJPRMxx	TIOC parameters that control TSO/TCAM time sharing buffers.
IKJTSOxx	For TSO/E, specifies authorized commands and authorized programs, 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.

Table 5 (Page 3 of 3). Overview of parmlib members	
Member	**Description**
IPCSPRxx	Parameters that are used during an IPCS session.
IVTPRM00	Sets Communications Storage Manager (CSM) parameters.
LNKLSTxx	List of data sets to be concatenated to form the LNKLST concatenation. You can also use PROGxx to define the concatenation.
LOADxx	Specifies data sets MVS uses to configure your system.
LPALSTxx	List of data sets to be concatenated to SYS1.LPALIB from which the system builds the pageable LPA (PLPA).
MMSLSTxx	Specifies information that the MVS message service (MMS) uses to control the languages that are available in your installation.
MPFLSTxx	Parameters that the message processing facility uses to control message processing and display.
MSTJCLxx	Contains the master scheduler job control language (JCL) that controls system initialization and processing.
NUCLSTxx	Specifies members of SYS1.NUCLEUS to be included in, or excluded from, the nucleus region at IPL-time.
PFKTABxx	Parameters contain the definitions for program function key tables (PFK tables).
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, and SYS1.CSSLIB to appear at the beginning of the LNKLST concatenation and SYS1.LPALIB to appear at the beginning of the LPALST concatenation.
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.
SMFPRMxx	Parameters that define SMF options.
TSOKEYxx	VTIOC parameters that are used by TSO/VTAM time sharing.
VATLSTxx	Volume attribute list that defines the mount and use attributes of direct access volumes.
XCFPOLxx	Specifies the actions that a system in a sysplex on PR/SM is to take when another system in the sysplex becomes inactive.

Notes:

1. These parameters can be listed at IPL: APF, DUMP, FIX, ICS, MLPA, SYSP, and OPT.
2. The only mandatory parameter is PAGE. Other parameters have coded defaults.
3. Performance-oriented parameters in IEASYSxx: APG, CMD, FIX, IPS, MLPA, OPT, REAL, RSU, WTOBFRS, WTORPLY.
4. System symbols in COMMNDxx and MSTJCLxx are processed differently than system symbols in other parmlib members. See the descriptions of COMMNDxx and MSTJCLxx in this book for details.

A.2 IEASYSxx SYS1.PARMLIB parameters

Table 6 (Page 1 of 5). 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.

Table 6 (Page 2 of 5). Overview of IEASYSxx parameters	
Parameter	**Use of the parameter**
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.
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 Greenwich Mean Time (GMT).
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.
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 hard-copy 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.
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 of 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.
DUMP	Specifies whether SYS1.DUMP data sets for SVC dump are to be on direct access device(s). This parameter can also indicate that no SYS1.DUMP data sets are to be made available for SVC dumps.
DUPLEX	Specifies a duplex page data set name or overrides the existing duplex data set name. This parameter is ignored on quick starts and warm starts (non-CLPA IPLs).
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 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.
ICS	Completes the name of the parmlib member (IEAICSxx) that contains the installation control specification. The installation control specification is used by the system resources manager (SRM) to assign performance groups.

Table 6 (Page 3 of 5). Overview of IEASYSxx parameters	
Parameter	**Use of the parameter**
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).
IPS	Completes the name of the parmlib member (IEAIPSxx) from which the system resources manager (SRM) will obtain the installation performance specification. The absence of an IPS= indicates that the system is to be IPLed in workload management goal mode.
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.
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.
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 to use to locate the parmlib statements to configure the OS/390 UNIX System Services 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. 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.
PAGTOTL	Specifies the total number of page and swap 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.

Table 6 (Page 4 of 5). Overview of IEASYSxx parameters	
Parameter	**Use of the parameter**
PLEXCFG	Specifies the sysplex configuration into which the system is allowed to IPL.
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 (RDE) feature is included. For information on RDE, see *OS/390 MVS Diagnosis: Tools and Service Aids*, LY28-1085 (available to IBM licensed customers only).
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 (either physical partitioning, or 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 created in response to a START command.
RTLS	Specifies the following for the RTLS configuration: • Library names to be managed • Module name to be managed • Cache storage to hold libraries • Library Volume Serial name
SCH	Specifies a parmlib member (SCHEDxx) from which the master scheduler will obtain 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 will obtain its parameters.
SMS	Specifies a parmlib member (IDGSMSxx) from which SMS will obtain 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 an installation supplies to define its own SVCs. NIP processing places these SVCs in the SVC table.
SWAP	Specifies the names of new swap data sets to be used as additions to or replacements for existing swap data sets.

Table 6 (Page 5 of 5). Overview of IEASYSxx parameters	
Parameter	**Use of the parameter**
SYSNAME	Specifies the name of the system being initialized. The name specified here is used by the system in various ways. For example: • In a multisystem 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) hard-copy log and in the display created by the DISPLAY R command. • The value specified can be used in the IGDDFPKG member to identify a system for which a particular DFSMS/MVS offering level is licensed.
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.
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.

A.3 System data sets

Name	Description
Master Catalog	A required system data set that contains data sets and volume information necessary to locate data sets and users.
IODF	A VSAM data set that contains hardware and software configuration definitions.
SYSn.IPLPARM	An optional partitioned data set that contains LOADxx members that point to I/O definition files (IODFs) that reside on the same volume as SYSn.IPLPARM. **Note:** The suffix *n* ranges from 0 to 9.
SYS1.BRODCAST	A required basic direct access method (BDAM) data set that can contain two types of TSO/E messages: • **Notices**—messages available to all users of a system. • **Mail**—messages available to specific users of a system. This system data set also contains a notice directory to make it easier to access each type of message.
SYS1.CMDLIB	An optional partitioned data set that contains TSO/E command processor routines, service routines, and utility programs.
SYS1.CSSLIB	A required partitioned data set (PDS) or partitioned data set extended (PDSE) that contains IBM-supplied linkage-assist routines that programs use for accessing callable services.
SYS1.DUMPnn	SYS1.DUMPnn system data sets (SYS1.DUMP00 through SYS1.DUMP99) are optional sequential data sets that contains system dumps, which record areas of virtual storage in case of system task failures. For more information on the creation and usage of SYS1.DUMPnn.
SYS1.HELP	An optional partitioned data set that contains help information about the syntax, operands, and function of each TSO/E command.
SYS1.IMAGELIB	A required partitioned data set that contains data for printers. For example, it contains the universal character set (UCS) images for 1403, 3203-5, and 3211; forms control buffer (FCB) modules for 3203-5, 3211, and 3800; and so forth.

Name	Description
SYS1.LINKLIB	A required partitioned data set (PDS) or partitioned data set extended (PDSE) that contains programs and routines referred to by the XCTL, ATTCH, LINK, and LOAD macro instructions, as well as nonresident system routines. SYS1.LINKLIB also contains the assembler program, the linkage editor, the utility programs, and some other service aids. You can also include user-written routines, in load module form, in SYS1.LINKLIB.
LOGREC	An optional data set that contains statistical data about machine failures, for example processor failures, I/O device errors, and channel errors. It also contains records for program error recording, missing-interrupt information, and dynamic device reconfiguration (DDR) routines. **Note:** The LOGREC data set cannot be shared between systems.
SYS1.LPALIB	A required partitioned data set that contains all the modules loaded into the pageable link pack area (PLPA). Those modules include system routines, SVC routines, data management access methods, nonresident machine-check handler modules, the program management binder, authorization and accounting exit routines, logon mode tables, and some TSO/E modules. You can also include user-written routines in load module form in SYS1.LPALIB.
SYS1.MACLIB	A required partitioned data set that contains macro definitions. You can also include user-written system macros in SYS1.MACLIB.
SYS1.MIGLIB	A required partitioned data set (PDS) or partitioned data set extended (PDSE) that is the system load library for the interactive problem control system (IPCS) and all component and subsystem dump exit modules. The component and subsystem dump exits that must function during SNAP processing must also reside in SYS1.LPALIB.
SYS1.MODGEN	A required partitioned data set that contains macro definitions. You can also include user-written system macros in SYS1.MODGEN.
SYS1.NUCLEUS	A required partitioned data set that contains: • The resident portion of the control program in two members, IEAVEDAT (DAT-off) and IEANUC0x (DAT-on). • The nucleus initialization program (NIP), and programs for hardware configuration definition (HCD) used by IPL. • Possibly a pointer (in member SYSCATxx) to an alternate master catalog.
SYS1.PARMLIB	A required partition data set that contains IBM-supplied and installation-created members, which contain lists of system parameter values. You can also include user-written system parameter members in SYS1.PARMLIB.
SYS1.PROCLIB	A required partitioned data set that contains the source JCL used to perform certain system functions. The source JCL can be for system tasks or processing program tasks invoked by the operator or the system programmer. You can also include user-written procedures in SYS1.PROCLIB.
SYS1.SAMPLIB	A required partitioned data set that contains the installation verification procedure (IVP), the independent utilities, and the IPL text. It also contains sample exit routines.
SYS1.SVCLIB	A required partitioned data set that contains some online text executive program (OLTEP) and appendage modules. You can also include user-written routines, in load module form in SYS1.SVCLIB.
SYS1.UADS	An optional partitioned data set that contains a list of authorized TSO/E users, as well as information about those users.
SYS1.STGINDEX	The VIO journaling data set is an optional VSAM data set that contains auxiliary storage management records for virtual I/O (VIO) data sets to enable them to be saved across IPLs for checkpoint/restart. **Note:** VIO data sets cannot be preserved if CVIO or CLPA is specified at IPL time. The VIO journaling data set is required only if you will use checkpoint/restart to restart jobs automatically, and want to preserve VIO data sets used by such jobs across IPLs. For more information about Checkpoint/Restart, see *DFSMS/MVS Version 1 Release 5 Checkpoint/Restart*, SC26-4907.

Name	Description
SYS1.VTAMLIB	An optional partitioned data set that contains the ACF/VTAM load modules, installation-coded logon exit routines, authorization and accounting exit routines, and unformatted system services (USS) definition tables. You can also include user-written routines, in load module form in SYS1.VTAMLIB. **Note:** SYS1.VTAMLIB is only required if you have ACF/VTAM installed in your system.

Appendix B. Job Management

This appendix includes references from the Chapter on the Introduction to Job Management.

B.1 Example of a JES2 Initialization Data Set

```
OPTSDEF   LIST=YES
/*********************************************************************/
/*                                                                   */
/*              SAMPLE JES2 PARAMETER LIBRARY LISTING                */
/*                                                                   */
/*********************************************************************/
CKPTDEF   CKPT1=(DSN=SYS1.JESCKPT1,VOL=CHECK1,INUSE=YES),
          CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2,INUSE=YES),
          MODE=DUPLEX,DUPLEX=ON
/*********************************************************************/
/*                                                                   */
/*               LOCAL DEVICES                                       */
/*                                                                   */
/*********************************************************************/
RDR(1)    UNIT=00C
RDR(2)    UNIT=011,PRIOLIM=9,CLASS=X,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO),
          PRTDEST=R3
PRT(1)    UNIT=002,CLASS=AJH,UCS=PN
PRT(2)    UNIT=00E,CLASS=AJH,UCS=PN
PRT(3)    UNIT=00F,CLASS=A,ROUTECDE=22,UCS=PN
PRT(4)    UNIT=018,CLASS=NI,MARK=YES,BURST=NO,TRKCELL=YES
PUN(1)    UNIT=00D,PAUSE=YES
INTRDR    PRIOLIM=9,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO)
INITDEF   PARTNUM=8
INIT(1)   CLASS=AFJKE                   /*INITIATOR 1*/
INIT(2)   CLASS=BCDEF                   /*INITIATOR 2*/
INIT(3)   CLASS=DEFGH                   /*INITIATOR 3*/
INIT(4)   CLASS=XKH                     /*INITIATOR 4*/
INIT(5)   CLASS=JKEBF                   /*INITIATOR 5*/
INIT(6)   DRAIN                         /*SPARE INITIATOR*/
INIT(7)   DRAIN                         /*SPARE INITIATOR*/
INIT(8)   DRAIN                         /*SPARE INITIATOR*/
JOBCLASS(STC) LOG=NO,OUTPUT=NO,CONDPURG=YES  /*STARTED TASK DEFINITION*/
JOBCLASS(TSU) REGION=50K,COMMAND=IGNORE,MSGLEVEL=(1,1),CONDPURG=YES
JOBCLASS(S)  PROCLIB=03,HOLD=YES        /*SYSTEM PROGRAMMER CLASS */
OUTCLASS(H)  OUTDISP=(HOLD,HOLD),TRKCEL=NO     /*SYSOUT CLASS HELD */
                                               /*FOR OUTPUT        */
OUTCLASS(N)  TRKCELL=YES
OUTCLASS(X)  OUTPUT=DUMMY,TRKCELL=NO     /*THROWAWAY CLASS*/
/*********************************************************************/
/*                                                                   */
/*               NJE/RJE INITIALIZATION PARAMETERS                   */
/*                                                                   */
/*********************************************************************/
NJEDEF    OWNNODE=1,NODENUM=10,LINENUM=15,RESTNODE=10
LINE(1)   UNIT=040,DUPLEX=FULL,TRANSPAR=YES,REST=10
LINE(2)   UNIT=041,TRANSPAR=YES,PASSWORD=SECRET,REST=20
LINE(3)   UNIT=042,TRANSPAR=YES,PASSWORD=SECRET,REST=15
LINE(4)   UNIT=043,TRANSPAR=YES,PASSWORD=SECRET,REST=50
LINE(5)   UNIT=044,TRANSPAR=YES,PASSWORD=SECRET,REST=10
```

```
LINE(6)   UNIT=SNA
LINE(7)   UNIT=SNA,PASSWORD=LINE4PW
RMT(1)    DEVTYPE=3780,LINE=1,NUMPUN=1,TRANSPAR=YES,BUFEXPEN=1,
          COMPRESS=YES
R1.PR1    PRWIDTH=144
RMT(2)    DEVTYPE=2922,NUMPU=1,CONS=YES,MULTILV=YES,TRANSPAR=YES
R2.PR1    PRWIDTH=132
RMT(3)    DEVTYPE=S/370,NUMPRT=2,CONS=YES,MULTILV=YES,TRANSPAR=YES
R3.PR1    PRWIDTH=150,FCBLOAD=YES
R3.PR2    PRWIDTH=132
RMT(4)    DEVTYPE=1130,CONS=YES,MULTILV=YES,NUMPUN=1
R4.PU1    START=NO
RMT(5)    DEVTYPE=SYSTEM/3,NUMRDR=3,NUMPUN=2,CONS=YES,MULTILV=YES
R5.PR1    PRWIDTH=132
RMT(6)    DEVTYPE=2780,NUMPUN=1,TRANSPAR=YES,MRF2780=YES,HTABS=YES
R6.PR1    PRWIDTH=144
RMT(7)    DEVTYPE=LUTYPE1,ROUTECDE=10,BUFSIZE=512,DISCINTV=8000,
          NUMPUN=1,COMPRESS=YES
RMT(8)    DEVTYPE=LUTYPE1,LINE=6,BUFSIZE=256,NUMPUN=1
OPTSDEF LIST=NO
/*********************************************************************/
/*                                                                   */
/*                  REMOTE PASSWORDS                                 */
/*                                                                   */
/*********************************************************************/
RMT(1)           PASSWORD=CHICAGO
RMT(2)           PASSWORD=BOARDWLK
RMT(3)           PASSWORD=ALBANY
RMT(4)           PASSWORD=LACKLSTR
RMT(5)           PASSWORD=KALAMAZO
RMT(6)           PASSWORD=UNIQUE
OPTSDEF LIST=YES
/*********************************************************************/
/*                                                                   */
/*                JES2 PARAMETER OVERRIDES                           */
/*                                                                   */
/*********************************************************************/
BUFDEF    BELOWBUF=(LIMIT=10),EXTBUF=(LIMIT=30)
SPOOLDEF  TRKCELL=6
PRINTDEF  NIUCS=GF12
/*********************************************************************/
/*                                                                   */
/*         MULTI-ACCESS SPOOL CONFIGURATION PARAMETERS               */
/*                                                                   */
/*********************************************************************/
MASDEF   OWNMEMB=SYSA,SHARED=NOCHECK
MEMBER(1) NAME=SYSA
MEMBER(2) NAME=SYSB
D SPOOLDEF VOLUME
D CKPTDEF CKPT1                      /* DISPLAY CURRENT */
D CONDEF CONCHAR                     /* PARAMETER VALUES */
/*********************************************************************/
/*                                                                   */
/*                       OPERATOR OVERRIDES                          */
/*                                                                   */
/*********************************************************************/
CONSOLE                              /* ALLOW OPERATOR TO OVERRIDE  */
/*********************************************************************/
/*                                                                   */
```

```
/*                      EVENT TRACE OPTIONS                       */
/*                                                                */
/******************************************************************/
TRACEDEF   ACTIVE=YES,                    /* START EVENT TRACING   */
           LOG=(CLASS=A,SIZE=200,START=YES),  /* LOG TRACE OUTPUT   */
           PAGES=4,TABLES=20,TABWARN=50    /* SET TABLE SIZE/NUMBER */
TRACE(1-*) START=YES                      /* START ALL TRACE IDs)  */
/******************************************************************/
/*                                                                */
/*                         OPERATOR COMMANDS                      */
/*                                                                */
/******************************************************************/
$S LINE(1-5)
$TLINE(1),TR=YES                   /*TRACE I/O COMPLETIONS ON LINE 1*/
$VS,'V (234,235,236,237),OFFLINE'
/******************************************************************/
/*                                                                */
/*               END OF JES2 PARAMETER LIBRARY LISTING            */
/*                                                                */
/******************************************************************/
```

Appendix C. Special Notices

This publication is intended to help new system programmers who need to understand S/390 and the OS/390 operating system. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 Versions. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 Version 2 Release 8, Program Number 5647-A01 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

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 IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

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

ACF/VTAM
AFP
CICS
CICS/MVS
Advanced Function Printing
AnyNet
CICS/ESA
DB2

DFSMS
DFSMSdfp
DFSMShsm
DFSORT
FICON
IMS
Intelligent Printer Data Stream
IP PrintWay
MVS (block letters)
NetSpool
OpenEdition
Parallel Sysplex
RACF
RMF
S/390 Parallel Enterprise Server
VTAM
DFSMS/MVS
DFSMSdss
DFSMSrmm
ESCON
IBM
InfoPrint
IPDS
Language Environment
MVS/DFP
NetView
OS/390
PrintWay
RAMAC
S/390
Sysplex Timer

The following terms are trademarks of other companies:

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjobenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

PC Direct is a trademark of Ziff Communications Company in the United States and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

SET, SET Secure Electronic Transaction, and the SET logo are trademarks owned by Secure Electronic Transaction LLC.

UNIX is a registered trademark in the United States and other countries licensed exclusively through the Open Group.

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

Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these ITSO publications see “How to get IBM Redbooks” on page 305.

- *OS/390 Release 5 Implementation*, SG24-5151
- *OS/390 Release 4 Implementation*, SG24-2089
- *OS/390 Release 3 Implementation*, SG24-2067
- *OS/390 Release 2 Implementation*, SG24-4834
- cit.OS/390 Security Server 1999 Updates Technical Presentation Guide, SG24-5627
- *Security in OS/390-based TCP/IP Networks*, SG24-5383
- *Hierarchical File System Usage Guide*, SG24-5482
- *Enhanced Catalog Sharing and Management*, SG245594
- *Integrated Catalog Facility Backup and Recovery*, SG24-5644
- *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customiztion*, SG24-5178
- *IBM S/390 FICON Implementation Guide*, SG24-5169
- *Exploiting S/390 Hardware Cryptography with Trusted Key Entry*, SG24-5455
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Introduction to Storage Area Network SAN*, SG2-45470
- *TCP/IP in a Sysplex*, SG24-5235
- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *ADSM Server-to-Server Implementation and Operation*, SG24-5244
- *Stay Cool on OS/390: Installing Firewall Technology*, SG24-2046
- *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268
- *TCP/IP OpenEdition Implementation Guide*, SG24-2141
- *IMS/ESA Version 5 Performance Guide*, SG24-4637
- *Parallel Sysplex Configuration: Overview*, SG24-2075
- *Parallel Sysplex Configuation: Cookbook*, SG24-2076
- *Parallel Sysplex Config.: Connectivity*, SG24-2077
- *DFSMS Optimizer Presentation Guide Update*, SG24-4477
- *MVS Parallel Sysplex Capacity Planning*, SG24-4680

- *Getting the Most Out of a Parallel Sysplex*, SG24-2073
- *OS/390 eNetwork Communication Server TCP/IP Implementation Guide Volume 2*, SG24-5228

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 Initialization and Tuning Guide*, SC28-1751
- *OS/390 Initialization and Tuning Reference*, SC28-1752
- *OS/390 Introduction and Release Guide*, GC28-1725
- *OS/390 MVS JCL User's Guide*, SC28-1758
- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS System Diagnosis: Tools and Service Aids*, LY28-1085, (available to IBM licensed customers only)
- *Interactive System Productivity Facility Getting Started*, SC34-4440
- *OS/390 Security Server (RACF) System Programmer's Guide*, SC28-1913
- *OS/390 TSO/E Customization*, SC28-1965
- *OS/390 TSO/E Primer*, GC28-1967
- *OS/390 TSO/E User's Guide*, SC28-1968
- *OS/390 SMP/E Reference*, SC28-1806
- *OS/390 SMP/E User's Guide*, SC28-1740
- *OS/390 SMP/E Commands*, SC28-1805
- *Standard Packaging Rules for MVS-Based Products*, SC23-3695
- *OS/390 MVS System Commands*, GC28-1781
- *OS/390 MVS IPCS Commands*, GC28-1754
- *OS/390 MVS IPCS User's Guide*, GC28-1756
- *DFSMS/MVS Using Data Sets*, SC26-4922
- *OS/390 Planning for Installation*, GC28-1726
- *OS/390 MVS System Data Sets Definition*, GC28-1782

- *ICKDSF R16 Refresh, User's Guide*, GC35-0033
- *OS/390 MVS System Management Facilities (SMF)*, GC28-1783
- *EREP V3R5 Reference*, GC35-0152
- *OS/390 JES2 Commands*, GC28-1790
- *OS/390 Hardware Configuration Definition User's Guide*, SC28-1848
- *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929
- *IBM ServerPac for OS/390 Using the Installation Dialog*, SC28-1244
- *OS/390 Hardware Configuration Definition Planning*, GC28-1750
- *OS/390 MVS Using the Subsystem Interface*, SC28-1502
- *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914
- *DFSMS/MVS Version 1 Release4: Access Method Services for Integrated Catalog Facility*, SC26-4906
- *DFSMS/MVS: DFSMShsm Implementation and Customization Guide*, SH21-1078
- *DFSMS/MVS Access Method Services for ICF Catalogs*, SC26-4500
- *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920
- *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*, SC31-8565
- *OS/390 eNetwork Communications Server SNA Resource Definition Samples*, SC31-8566
- *OS/390 eNetwork Communications Server: SNA Operation*, SC31-8567
- *OS/390 V2R7.0 eNetwork CS IP Configuration*, SC31-8513
- *eNetwork Communications Server: IP User's Guide* GC31-8514
- *OS/390 UNIX System Services Planning*, SC28-1890
- *OS/390 TCP/IP OpenEdition: Configuration Guide* SC31-8304
- *OS/390 Open Systems Adapter Support Facility User's Guide*, SC28-1855.
- *OS/390 V2R6.0 MVS Planning: APPC/MVS Management*, GC28-1807
- *Print Services Facility for OS/390: Customization*, S544-5622
- *DFSMS/MVS Planning for Installation*, SC26-4919
- *DFSMS/MVS Implementing System-Managed Storage*, SC26-3123
- *DFSMS/MVS Remote Copy Administrator's Guide and Reference*, SC35-0169
- *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913
- *DFSMS/MVS DFSMSdfp Diagnosis Guide*, SY27-9605
- *DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921
- *DFSMS/MVS Using Magnetic Tapes*, SC26-4923
- *DFSMS/MVS Utilities*, SC26-4926
- *Service Level Reporter User's Guide: Reporting*, SH19-6530
- *DFSMS/MVS Object Access Method Application Programmer's Reference*, SC26-4917
- *DFSMS/MVS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC26-4918
- *DFSORT Installation and Customization*, SC26-7041
- *DFSORT Getting Started with DFSORT R14*, SC26-4109
- *DFSMS/MVS Network File System Customization and Operation*, SC26-7029

- *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930
- *DFSMShsm Storage Administration Guide*, SH21-1076
- *DFSMShsm Storage Administration Reference*, SH21-1075
- *DFSMS/MVS Network File System User's Guide*, SC26-7028
- *DFSMS/MVS DFSMSrmm Guide and Reference*, SC26-4931
- *DFSMS/MVS DFSMSrmm Implementation and Customization Guide*, SC26-4932
- *MVS/ESA Storage Management Library Managing Data*, SC26-3124
- *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125
- *MVS/ESA Storage Management Library Leading a Storage Administration Group*, SC26-3126.
- *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911
- *ADSTAR Distributed Storage Manager for MVS Administrator's Guide*, GC35-0277
- *OS/390 MVS Programming: Assembler Services Guide*, GC28-1762
- *OS/390 MVS Programming: Resource Recovery*, GC28-1739
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779
- *OS/390 MVS Sysplex Services Guide*, GC28-1771
- *OS/390 Parallel Sysplex Systems Management*, GC28-1861
- *OS/390 MVS Systems Codes*, GC28-1780
- *OS/390 MVS System Messages Volume 1*, GC28-1784
- *OS/390 MVS System Messages Volume 2*, GC28-1785
- *OS/390 MVS System Messages Volume 3*, GC28-1786
- *OS/390 MVS System Messages Volume 4*, GC28-1787
- *OS/390 MVS System Messages Volume 5*, GC28-1788
- *OS/390 MVS Installation Exits*, SC28-1753
- *OS/390 MVS Diagnosis Reference*, SY28-1084
- *CICS User's Handbook*, SX33-1188
- *CICS Diagnosis Guide*, LX33-6093
- *MQSeries for MVS/ESA Messages and Codes*, GC33-0819

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbook fax order form to:

In United States:	e-mail address: usib6fpl@ibmmail.com
Outside North America:	Contact information is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

abend. Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task executing.

ACB. Access method control block.

access. A specific type of interaction between a subject and an object that results in the flow of information from one to the other.

access authority. An authority that relates to a request for a type of access to protected resources. In RACF, the access authorities are NONE, READ, UPDATE, ALTER, and EXECUTE.

access list. A list within a profile of all authorized users and their access authorities.

access method control block (ACB).. A control block that links an application program to VTAM.

ACDS. Active control data set.

ACF/VTAM. An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. VTAM runs under MVS (OS/VS1 and OS/VS2), VSE, and VM/SP and supports direct control application programs and subsystems such as VSE/POWER.

ACIF. (1) AFP conversion and indexing facility. (2) A PSF utility program that converts a print file into AFP, MO:DCA-P, creates an index file for input data, and collects resources used by an AFP document into separate file.

action message retention facility (AMRF). A facility that, when active, retains all action messages except those specified by the installation in the MPFLSTxx member in effect.

action message sequence number. A decimal number assigned to action messages.

Advanced Function Presentation (AFP). A set of licensed programs, together with user applications, that use the all-points-addressable concept to print on presentation devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

Advanced Program-to-Program Communications (APPC). A set of inter-program communication services that support cooperative transaction processing in a SNA network.

AFP. Advanced Function Presentation.

AFP Printer Driver for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and creates output in AFP format, for printing on AFP printers.

AFP Viewer plug-in for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and allows you to view files in AFP format.

AIX operating system. IBM′s implementation of the UNIX operating system. The RS/6000 system, among others, runs the AIX operating system.

allocate. To assign a resource for use in performing a specific task.

alphanumeric character. A letter or a number.

amode. Addressing mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. AMODE states the addressing mode that is expected to be in effect when the program is entered.

AMRF. action message retention facility

AOR. Application-owning region

APPC. Advanced Program-to-Program Communications

APPN. Advanced Peer-to-Peer Networking.

ASCII (American Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bit including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

audit. To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data accuracy.

authority. The right to access objects, resources, or functions.

authorization checking. The action of determining whether a user is permitted access to a RACF-protected resource.

Authorized Program Analysis Report (APAR). A request for correction of problem caused by a defect in a current unaltered release of a program.

authorized program facility (APF). A facility that permits identification of programs authorized to use restricted functions.

automated operations. Automated procedures to replace or simplify actions of operators in both systems and network operations.

AVR. Automatic volume recognition.

B

banner page. A page printed before the data set is printed.

basic mode. A central processor mode that does not use logical partitioning. Contrast with logically partitioned (LPAR) mode.

batch message processing (BMP) program. An IMS batch processing program that has access to online databases and message queues. BMPs run online, but like programs in a batch environment, they are started with job control language (JCL).

batch-oriented BMP program. A BMP program that has access to online databases and message queues while performing batch-type processing. A batch-oriented BMP does not access the IMS message queues for input or output. It can access online databases, GSAM databases, and MVS files for both input and output.

BMP. Batch message processing (BMP) program.

broadcast. (1) Transmission of the same data to all destinations. (2) Simultaneous transmission of data to more than one destination.

binary data. (1) Any data not intended for direct human reading. Binary data may contain unprintable characters, outside the range of text characters. (2) A type of data consisting of numeric values stored in bit patterns of 0s and 1s. Binary data can cause a large number to be placed in a smaller space of storage.

BIND. In SNA, a request to activate a session between two logical units (LUs).

buffer. A portion of storage used to hold input or output data temporarily.

buffered device. A device where the data is written to a hardware buffer in the device before it is placed on the paper (for example, IBM 3820).

burst. To separate continuous-forms paper into single sheets.

C

cache structure. A coupling facility structure that enables high-performance sharing of cached data by multisystem applications in a sysplex. Applications can use a cache structure to implement several different types of caching systems, including a store-through or a store-in cache.

carriage control character. An optional character in an input data record that specifies a write, space, or skip operation.

carriage return (CR). (1) A keystroke generally indicating the end of a command line. (2) In text data, the action that indicates to continue printing at the left margin of the next line. (3) A character that will cause printing to start at the beginning of the same physical line in which the carriage return occurred.

CART. Command and response token.

case-sensitive. Pertaining to the ability to distinguish between uppercase and lowercase letters.

catalog. (1) A directory of files and libraries, with reference to their locations. (2) To enter information about a file or a library into a (3) The collection of all data set indexes that are used by the control program to locate a volume containing a specific data set.

CBPDO. Custom Built Product Delivery Offering.

CEC. Synonym for central processor complex (CPC).

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

central processor complex (CPC). A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

CFRM. Coupling facility resource management.

channel-to-channel (CTC). Refers to the communication (transfer of data between programs on opposite sides of a channel-to-channel adapter (CTCA

channel-to-channel adapter (CTCA). An input/output device that is used a program in one system to communicate with a program in another system.

checkpoint. (1) A place in a routine where a check, or a recording of data for restart purposes, is performed. (2) A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.

checkpoint write. Any write to the checkpoint data set. A general term for the primary, intermediate, and final writes that update any checkpoint data set.

CICS. Customer Information Control System.

CICSplex. A group of connected CICS regions.

CICSPlex SM. CICSPlex System Manager

client. A functional unit that receives shared services from a server. See also client-server.

client-server. In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

CMOS. Complementary metal-oxide semiconductor.

CNGRPxx. The SYS1.PARMLIB member that defines console groups for the system or sysplex.

code page. (1) A table showing codes assigned to character sets. (2) An assignment of graphic characters and control function meanings to all code points. (3) Arrays of code points representing characters that establish ordinal sequence (numeric order) of characters. (4) A particular assignment of hexadecimal identifiers to graphic elements.

code point. A 1-byte code representing one of 256 potential characters.

coexistence. Two or more systems at different levels (for example, software, service or operational levels) that share resources. Coexistence includes the ability of a system to respond in the following ways to a new function that was introduced on another system with which it shares resources: ignore a new function, terminate gracefully, support a new function.

command and response token (CART). A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

command prefix facility (CPF). An MVS facility that allows you to define and control subsystem and other command prefixes for use in a sysplex.

COMMDS. Communications data set.

complementary metal-oxide semiconductor (CMOS). A technology that combines the electrical properties of positive and negative voltage requirements to use considerably less power than other types of semiconductors.

connection. In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In Internet communications, a connection extends from a TCP application on one syste system to a TCP application on another system.

console. That part of a computer used for communication between the operator or user and the computer.

console group. In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

CONSOLxx. The SYS1.PARMLIB member used to define message handling, command processing, and MCS consoles.

control unit. Synonymous with device control unit.

conversation. A logical connection between two programs over an LU type 6.2 session that allows them to communicate with each other while processing a transaction.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

copy group. One or more copies of a page of paper. Each copy can have modifications, such as text suppression, page position, forms flash, and overlays.

couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. See also sysplex couple data set.

coupling facility. A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

coupling facility channel.. A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

coupling services. In a sysplex, the functions of XCF that transfer data and status between members of a group residing on one or more MVS systems in the sysplex.

CP. Central processor.

CPC. Central processor complex.

CPF. Command prefix facility.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

cryptography. The transformation of data to conceal its meaning.

cryptographic key. A parameter that determines cryptographic transformations between plaintext and ciphertext.

CTC. Channel-to-channel.

Customer Information Control System (CICS). An IBM licensed program tha that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

D

DAE. Dump analysis and elimination.

daemon. A program that runs unattended to perform a standard service.

DASD. Direct access storage device.

data definition name. The name of a data definition (DD) statement, which corresponds to a data control block that contains the same name. Abbreviated as ddname.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data integrity. The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set label. (1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

data set separator pages. Those pages of printed output that delimit data sets.

data sharing. The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data while maintaining data integrity.

data stream. (1) All information (data and control commands) sent over a data link usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

DBCS. Double-byte character set.

DBCTL. IMS Database Control.

DBRC. Database Recovery Control.

DB2. DATABASE 2 for MVS/ESA.

DB2 data sharing group. A collection of one or more concurrent DB2 subsystems that directly access and change the same data while maintaining data integrity.

DB2 PM. DB2 Performance Monitor.

deallocate. To release a resource that is assigned to a specific task.

default. A value, attribute, or option that is assumed when no alternative is specified by the user.

destination node. The node that provides application services to an authorized external user.

device control unit. A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices or terminals.

device number. The unique number assigned to an external device.

device type. The general name for a kind of device; for example, 3330.

DFSMS. Data Facility Storage Management Subsystem.

direct access storage device (DASD). A device in which the access time effectively independent of the location of the data.

directory. (1) A type of file containing the names and controlling information for other files or other directories. Directories can also contain subdirectories, which can contain subdirectories of their own. (2) A file that contains directory entries. No two directory entries in the same directory can have the same name. (POSIX.1). (3) A file that points to files and to other directories. (4) An index used by a control program to locate blocks of data that are stored in separate areas of a data set in direct access storage.

display console. In MVS, an MCS console whose input/output function you can control.

DLL filter. A filter that provides one or more of these functions in a dynamic load library - init(), prolog(), process(), epilog(), and term(). See cfilter.h and cfilter.c in the /usr/lpp/Printsrv/samples/ directory for more information. See also filter. Contrast with DLL filter.

DOM. An MVS macro that removes outstanding WTORs or action messages that have been queued to a console end-of-tape-marker. A marker on a

magnetic tape used to indicate the end of the permissible recording area, for example, a photo-reflective strip a transparent section of tape, or a particular bit pattern.

dotted decimal notation. The syntactical representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. It is used to represent IP addresses.

double-byte character set (DBCS). A set of characters in which each character is represented by a two-bytes code. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

drain. Allowing a printer to complete its current work before stopping the device.

E

entry area. In MVS, the part of a console screen where operators can enter commands or command responses.

EMIF. ESCON Multiple Image Facility.

Enterprise Systems Connection (ESCON). A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

EPDM. IBM SystemView Enterprise Performance Data Manager/MVS.

ESCD. ESCON Director.

ESCM. ESCON Manager. The licensed program System Automation for OS/390 includes all of the function previosuly provided by ESCM.

ESCON. Enterprise Systems Connection.

ETR. External Time Reference. See also Sysplex Timer.

extended MCS console. In MVS, a console other than an MCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

F

FMID. Function modification identifier. The IBM release-specific product identifier such as HJE6610 for OS/390 Release 1 JES2.

FOR. File-owning region.

frame. For a System/390 microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

FSS. functional subsystem. An address space uniquely identified as performing a specific function related to the JES. An example of an FSS is the program Print Services Facility that operates the 3800 Model 3 an 38xx printers.

functional subsystem (FSS). An address space uniquely identified as performing a specific function related to the JES.

functional subsystem application (FSA). The functional application program managed by the functional subsystem.

functional subsystem interface (FSI). The interface through which JES2 JES3 communicate with the functional subsystem.

G

gateway node. A node that is an interface between networks.

generalized trace facility (GTF). Like system trace, gathers information used to determine and diagnose problems that occur during system operation. Unlike system trace, however, GTF can be tailored to record very specific system and user program events.

global access checking. The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources.

global resource serialization. A function that provides an MVS serialization mechanism for resources (typically data sets) across multiple MVS images.

global resource serialization complex. One or more MVS systems that use global resource serialization to serialize access to shared resources (such as data sets on shared DASD volumes).

group. A collection of RACF users who can share access authorities for protected resources.

GTF. Generalized trace facility.

H

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

hardware. Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

hardware configuration dialog. In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

Hardware Management Console. A console used to monitor and control hardware such as the System/390 microprocessors.

HCD. Hardware Configuration Definition.

highly parallel. Refers to multiple systems operating in parallel, each of which can have multiple processors. See also n-way.

I

ICMF. Integrated Coupling Migration Facility.

IMS. Information Management System.

IMS DB. Information Management System Database Manager.

IMS DB data sharing group. A collection of one or more concurrent IMS DB subsystems that directly access and change the same data while maintaining data integrity.

IMS TM. Information Management System Transaction Manager.

initial program load (IPL). The initialization procedure that causes an operating system to begin operation.

instruction line. In MVS, the part of the console screen that contains messages about console control and input errors.

internal reader. A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

IOCDS. Input/output configuration data set.

IOCP. Input/output configuration program.

IODF. Input/output definition file.

IPL. Initial program load.

IRLM. Internal resource lock manager.

ISPF. Interactive System Productivity Facility.

J

JES common coupling services. A set of macro-driven services that provide the communication interface between JES members of a sysplex. Synonymous with JES XCF.

JESXCF. JES cross-system coupling services. The MVS component, common to both JES2 and JES3, that provides the cross-system coupling services to either JES2 multi-access spool members or JES3 complex members, respectively.

JES2. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

JES2 multi-access spool configuration. A multiple MVS system environment that consists of two or more JES2 processors sharing the same job queue and spool

JES3. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In complexes that have several loosely-coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

JES3 complex. A multiple MVS system environment that allows JES3 subsystem consoles and MCS consoles with a logical association to JES3 to receive messages and send commands across systems.

job entry subsystem (JES). A system facility for spooling, job queuing, and managing the scheduler work area.

job separator page data area (JSPA). A data area that contains job-level information for a data set. This information is used to generate job header, job trailer or data set header pages. The JSPA can be used by an installation-defined JES2 exit routine to duplicate the information currently in the JES2 separator page exit routine.

job separator pages. Those pages of printed output that delimit jobs.

K

keyword. A part of a command operand or SYS1.PARMLIB statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

L

LIC. Licensed Internal Code.

list structure. A coupling facility structure that enables multisystem applications in a sysplex to share information organized as a set of lists or queues. A list structure consists of a set of lists and an optional lock table, which can be used for serializing resources in the list structure. Each list consists of a queue of list entries.

lock structure. A coupling facility structure that enables applications in a sysplex to implement customized locking protocols for serialization of application-defined resources. The lock structure supports shared, exclusive, and application-defined lock states, as well as generalized contention management and recovery protocols.

logical partition (LP). A subset of the processor hardware that is defined to support an operating system. See also logically partitioned (LPAR) mode.

logically partitioned (LPAR) mode. A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, central storage, expanded storage, and channel paths) among logical partitions. Contrast with basic mode.

logical unit (LU). In SNA, a port through which an end user accesses th SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs).

logical unit type 6.2. The SNA logical unit type that supports general communication between programs in a cooperative processing environment.

loosely coupled. A multisystem structure that requires a low degree of interaction and cooperation between multiple MVS images to process a workload. See also tightly coupled.

LP. Logical partition.

LPAR. Logically partitioned (mode).

M

MAS. Multi-access spool.

master console. In an MVS system or sysplex, the main console used for communication between the operator and the system from which all MVS commands can be entered. The first active console with AUTH(MASTER) defined becomes the master console in a system or sysplex.

master console authority. In a system or sysplex, a console defined with AUTH(MASTER) other than the master console from which all MVS commands can be entered.

master trace. A centralized data tracing facility of the master scheduler, used in servicing the message processing portions of MVS.

MCS. Multiple console support.

MCS console. A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

member. A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

message processing facility (MPF). A facility used to control message retention, suppression, and presentation.

message queue. A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message text. The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

microprocessor. A processor implemented on one or a small number of chips.

mixed complex. A global resource serialization complex in which one or more of the systems in the global resource serialization complex are not part of a multisystem sysplex.

MP. Multiprocessor.

MPF. Message processing facility.

MPFLSTxx. The SYS1.PARMLIB member that controls the message processing facility for the system.

MRO. Multiregion operation.

multiple console support (MCS). The operator interface in an MVS system.

multi-access spool (MAS). A complex of multiple processors running MVS/JES2 that share a common JES2 spool and JES2 checkpoint data set.

multiprocessing. The simultaneous execution of two or more computer programs or sequences of instructions. See also parallel processing.

multiprocessor (MP). A CPC that can be physically partitioned to form two operating processor complexes.

multisystem application. An application program that has various functions distributed across MVS images in a multisystem environment.

multisystem console support. Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

multisystem environment. An environment in which two or more MVS images reside in one or more processors, and programs on one image can communicate with programs on the other images.

multisystem sysplex. A sysplex in which two or more MVS images are allowed to be initialized as part of the sysplex.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

MVS router. The MVS router is a system service that provides an installation with centralized control over system security processing.

MVS system. An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.

MVS/ESA. Multiple Virtual Storage/ESA.

MVSCP. MVS configuration program.

N

n-way. The number (n) of CPs in a CPC. For example, a 6-way CPC contains six CPs.

NIP. Nucleus initialization program.

NJE. Network job entry.

no-consoles condition. A condition in which the system is unable to access any full-capability console device.

nonstandard labels. Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

nucleus initialization program (NIP). The stage of MVS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

O

offline. Pertaining to equipment or devices not under control of the processor.

OLTP. Online transaction processing.

online. Pertaining to equipment or devices under control of the processor.

OPC/ESA. Operations Planning and Control.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

operations log. In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

OPERLOG. The operations log.

OPERPARM. In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

OS/390. OS/390 is a network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

OS/390 Network File System. A base element of OS/390, that allows remote access to MVS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

OS/390 UNIX System Services (OS/390 UNIX). The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the OS/390 operating system that allow users to write and run application programs that conform to UNIX standards.

P

parallel processing. The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch). See also highly parallel.

Parallel Sysplex. A sysplex that uses one or more coupling facilities.

partitionable CPC. A CPC that can be divided into 2 independent CPCs. See also physical partition, single-image mode, MP, side.

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it.

permanent data set. A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with temporary data set.

PFK. Program function key.

PFK capability. On a display console, indicates that program function keys are supported and were specified at system generation.

PFKTABxx. The SYS1.PARMLIB member that controls the PFK table settings for MCS consoles in a system.

physical partition. Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

physically partitioned (PP) configuration. A system configuration that allows the processor controller to use both central processor complex (CPC) sides as individual CPCs. The A-side of the processor controller controls side 0; the B-side of the processor controller controls side 1. Contrast with single-image (SI) configuration.

PR/SM. Processor Resource/Systems Manager.

Print Services Facility (PSF). The access method that supports the 3800 Printing Subsystem Models 3 and 8. PSF can interface either directly to a user′s application program or indirectly through the Job Entry Subsystem (JES) of MVS.

printer. (1) A device that writes output data from a system on paper or other media.

processor controller. Hardware that provides support and diagnostic functions for the central processors.

Processor Resource/Systems Manager (PR/SM). The feature that allows the processor to use several MVS images simultaneously and provides logical partitioning capability. See also LPAR.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program function key (PFK). A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

program status word (PSW). A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

pseudo-master console. A subsystem-allocatable console that has system command authority like that of an MCS master console.

PSW. Program status word.

R

RACF. See Resource Access Control Facility.

RAID. See redundant array of independent disk.

RAMAC Virtual Array (RVA) system. An online, random access disk array storage system composed of disk storage and control unit combined into a single frame.

read access. Permission to read information.

recording format. For a tape volume, the format of the data on the tape, for example, 18, 36, 128, or 256 tracks.

recovery. The process of rebuilding data after it has been damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a log.

redundant array of independent disk (RAID). A disk subsystem architecture that combines two or more physical disk storage devices into a single logical device to achieve data redundancy.

remote operations. Operation of remote sites from a host system.

Resource Access Control Facility (RACF). An IBM-licensed program or a base element of OS/390, that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system and logging the detected accesses to protected resources.

restructured extended executor (REXX). A general-purpose, procedural language for end-user personal programming, designed for ease by both casual general users and computer professionals. It is also useful for application macros. REXX includes the capability of issuing commands to the underlying operating system from these macros and procedures. Features include powerful character-string manipulation, automatic data typing, manipulation of objects familiar to people, such as words, numbers, and names, and built-in interactive debugging.

REXX. See restructured extended executor.

RMF. Resource Measurement Facility.

rmode. Residency mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. RMODE states the virtual storage location (either above 16 megabytes or anywhere in virtual storage) where the program should reside.

roll mode. The MCS console display mode that allows messages to roll of off the screen when a specified time interval elapses.

roll-deletable mode. The console display mode that allows messages to roll off the screen when a specified time interval elapses. Action messages remain at the top of the screen where operators can delete them.

routing. The assignment of the communications path by which a message will reach its destination.

routing code. A code assigned to an operator message and used to route the message to the proper console.

RVA. See RAMAC Virtual Array system.

S

SCDS. Source control data set.

SDSF. System Display and Search Facility.

shared DASD option. An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

side. A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

single point of control. The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

single system image. The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

single-image (SI) mode. A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with physically partitioned (PP) configuration.

single-system sysplex. A sysplex in which only one MVS system is allowed to be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system but does not provide signalling services between MVS systems. See also multisystem sysplex, XCF-local mode.

SLR. Service Level Reporter.

small computer system interface (SCSI). A standard hardware interface that enables a variety of peripheral devices to communicate with one another.

SMF. System management facilities.

SMP/E. System Modification Program Extended.

SMS. Storage Management Subsystem.

SMS communication data set. The primary means of communication among systems governed by a single SMS configuration. The SMS communication data set (COMMDS) is a VSAM linear data set that contains the current utilization statistics for each system-managed volume, which SMS uses to help balance space usage among systems.

SMS configuration. The SMS definitions and routines that the Storage Management Subsystem uses to manage storage.

SMS system group. All systems in a sysplex that share the same SMS configuration and communications data sets, minus any systems in the sysplex that are defined individually in the SMS configuration.

software. (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) Contrast with hardware. A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library

routines, manuals, circuit diagrams. Contrast with hardware.

spanned record. A logical record contained in more than one block.

status-display console. An MCS console that can receive displays of system status but from which an operator cannot enter commands.

storage administrator. A person in the data processing center who is responsible for defining, implementing, and maintaining storage manageme policies.

storage class. A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage group. A collection of storage volumes and attributes, defined the storage administrator. The collections can be a group of DASD volume or tape volumes, or a group of DASD, optical, or tape volumes treated as single object storage hierarchy. See tape storage group.

storage management. The activities of data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These can be done either manually or by using automated processes. cThe Storage Management Subsystem automates these processes for you, while optimizing storage resources. See also Storage Management Subsystem.

Storage Management Subsystem (SMS). A DFSMS/MVS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

storage subsystem. A storage control and its attached storage devices. See also tape subsystem.

structure. A construct used by MVS to map and manage storage on a coupling facility. See cache structure, list structure, and lock structure.

subsystem-allocatable console. A console managed by a subsystem like JES3 or NetView used to communicate with an MVS system.

subsystem interface (SSI). An MVS component that provides communication between MVS and JES.

supervisor call instruction (SVC). An instruction that interrupts a program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

support element. A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

SVC routine. A control program routine that performs or begins a contro program service specified by a supervisor call instruction.

symmetry. The characteristic of a sysplex where all systems, or certain subsets of the systems, have the same hardware and software configurations and share the same resources.

synchronous messages. WTO or WTOR messages issued by an MVS system during certain recovery situations.

SYSLOG. The system log data set.

sysplex. A set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads. See also MVS system, Parallel Sysplex.

sysplex couple data set. A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All MVS systems in a sysplex must have connectivity to the sysplex couple data set. See also couple data set.

Sysplex Timer. An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the MVS generic name for the IBM Sysplex Timer (9037).

system control element (SCE). Hardware that handles the transfer of data and control information associated with storage requests between the elements of the processor.

system console. In MVS, a console attached to the processor controller used to initialize an MVS system.

system log (SYSLOG). In MVS, the system log data set that includes all entries made by the WTL (write-to-log) macro as well as the hardcopy log. SYSLOG is maintained by JES in JES SPOOL space.

system management facilities (SMF). An optional control program feature of OS/390 and MVS that provides the means for gathering and recording information that can be used to evaluate system usage.

System Modification Program Extended (SMP/E). In addition to providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog

interface, and supports dynamic allocation of data sets.

Systems Network Architecture (SNA). A description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of networks.

system trace. A chronological record of specific operating system events. The record is usually produced for debugging purposes.

T

temporary data set. A data set that is created and deleted in the same job.

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

terminal user. In systems with time-sharing, anyone who is eligible to log on.

tightly coupled. Multiple CPs that share storage and are controlled by a single copy of MVS. See also loosely coupled, tightly coupled multiprocessor.

tightly coupled multiprocessor. Any CPU with multiple CPs.

Time Sharing Option (TSO). An option on the operating system; for OS/390 the option provides interactive time sharing from remote terminals.

TOR. Terminal-owning region.

transaction. In APPC/MVS, a unit of work performed by one or more transaction programs, involving a specific set of input data and initiating a specific process or job.

transaction program (TP). For APPC/MVS, any program on MVS that issues APPC/MVS or CPI Communication calls, or is scheduled by the APPC/MVS transaction scheduler.

U

undelivered message. An action message or WTOR that cannot be queued for delivery to the expected console. MVS delivers these messages to any console with the UD console attribute in a system or sysplex.

uniprocessor (UP). A CPC that contains one CP and is not partitionable.

UP. Uniprocessor.

V

VM. Virtual Machine.

virtual telecommunications access method (VTAM). A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2 operating systems.

volume. (1) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

volume serial number. A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents (VTOC). A table on a direct access volume that describes each data set on the volume.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method.

VTOC. Volume table of contents.

W

wait state. Synonymous with waiting time.

waiting time. (1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

WLM. MVS workload management.

wrap mode. The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

write-to-log (WTL) message. A message sent to SYSLOG or the hardcopy log.

write-to-operator (WTO) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

write-to-operator-with-reply (WTOR) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

WTL message. Write-to-log message

WTO message. Write-to-operator message

WTOR message. Write-to-operator-with-reply message.

X

XCF. Cross-system coupling facility.

XCF PR/SM policy. In a multisystem sysplex on PR/SM, the actions that XCF takes when one MVS system in the sysplex fails. This policy provides high availability for multisystem applications in the sysplex.

XCF-local mode. The state of a system in which XCF provides limited services on one system and does not provide signalling services between MVS systems. See also single-system sysplex.

XRF. Extended recovery facility.

IBM Redbooks evaluation

ABCs of OS/390 System Programming Volume 2
SG24-5652-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-5652-00
Printed in the U.S.A.

SG24-5652-00