What is a Main Frame

Mainframe is a high end server. The largest type of server in use today will be referred as Mainframe by IBM. Mainframe can support dozens of applications and input/output devices to simultaneously serve thousands of users. ”A Mainframe is what businesses use to host the commercial databases, transaction servers, and applications that require a greater degree of security and availability than is commonly found on smaller-scale machines”

Businesses today relay on the mainframe to:

- Perform large-scale transaction processing (thousands of transactions per second
- Support thousands of users and application programs concurrently accessing many resources.
- Mange terabytes of information in databases
- High large-bandwidth communications

Factors contributing to mainframe use

Reliability, Availability, Serviceability, Security, Scalability, Compatibility

Z/OS Overview

Let’s have an overview on one of the latest Operating system

What is an Operating System?

Simplest term “OS” is a collection of program that manage the internal workings of a computer system.

What is Z/OS?

- Z – Stands for Zero Down Time!
- Z/OS is designed to offer a stable, secure, and continuously available environment for application s running on the mainframe.
- An Operating system is a collection of programs that manage the internal workings of a computer system.
- Z/OS, the most widely used mainframe operating system.

- Z/OS use of multiprogramming and multiprocessing, and its ability to access and manage enormous amounts of storage and I/O operations, makes it ideally suited for running mainframe workloads.

Role of a system programmer

The role of the system programmer is to install, customize, and maintain the operating system. The OS/390 operating system runs on various hardware configurations.

A system programmer must also define the hardware I/O configuration resources that are to be available to the OS/390 operating system.

As an OS/390 system programmer, you must be aware of the following:

- Storage concepts
- Device I/O configurations
- Processor configurations
- Console definitions
- System libraries where the software is placed
- System data sets and their placement
- Customization parameters that are used to define your OS/390 configuration

An overview of Hardware

An IBM System /390 consists of hardware and software products. The Hardware has a central processor complex (CPC) that includes the central processor, storage, channels etc. The Software consists of system application programs, end-user application tools etc. The primary program executing on the system is an operating system such as OS/390.

The OS/390 operating system manages the instructions to be processed and the resources required to process them. A single copy of OS/390 running on a single processor system is called a uni-processor.

By adding more processors to a CPC, multiple program instructions can be processed simultaneously. A multiprocessor system running under a single OS/390 image is referred to as non-partitionable multiprocessor. Some S/390 multiprocessors have a physical partitioning capability which can support different operating system on each partition. With the introduction of Processor Resources/System Management (PR/SM) System /390 was enabled to run logical partitions (LPARs).

The PR/SM feature allows multiple logical partitions such as:

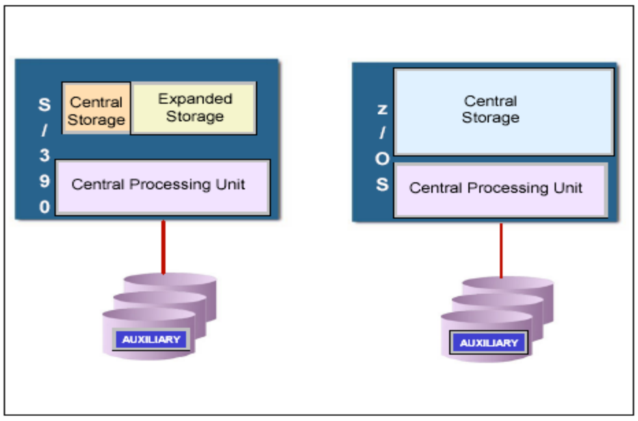
- Processors
- Storage
- Channel paths

A sysplex (systems complex) consists of two or more MVS systems residing on one or more CPCs. It is the basis for a simplified multi-system management. In a sysplex, an installation can view multiple MVS systems as a single entity. The enhanced parallel-sysplex supports communication and data sharing among more systems, thus dynamically balancing workloads.

Storage Concepts

Processor storage overview

From 370-XA until ESA/390 architecture, processor storage consisted of central plus expanded storage. In z/OS architecture there is no expanded storage. The system uses a portion of both central storage and virtual storage.



Central storage

Central storage often referred to as main storage, provides the system with directly addressable fast-access storage of data. Both data and programs must be loaded into central storage (from input devices) before they can be processed.

Main storage may include one or more smaller faster-access buffer storages, sometimes called caches. A cache is usually physically associated with a CPU or an I/O processor. The effects, except on performance, of the physical construction and use of distinct storage media are not observable by the program.

Expanded storage

Expanded storage may be available on some models. Expanded storage, when available, can be accessed by all CPUs in the configuration by means of instructions that transfer 4 KB blocks of data from expanded storage to main storage or from main storage to expanded storage.

Each 4 KB block in expanded storage is addressed by means of a 32-bit unsigned binary integer called an expanded-storage block number.

CPU

The central processing unit (CPU) is the controlling center of the system. It contains the sequencing and processing facilities for instruction execution, interruption action, timing functions, initial program loading and other machine-related functions.

The physical implementation of the CPU may differ among models, but the logical function remains the same. The result of executing an instruction is the same for each model, providing that the program complies with the compatibility rules.

The CPU, in executing instructions, can process binary integers and floating-point numbers of fixed length, decimal integers of variable length, and logical information of either fixed or variable length. Processing may be in parallel or in series; the width of the processing elements, the multiplicity of the shifting paths, and the degree of simultaneity in performing the different types of arithmetic differ from one CPU to another without affecting the logical results.

Auxiliary storage

An installation needs auxiliary direct access storage devices (DASD) for placement of all system data sets. Enough auxiliary storage must be available for the programs and data that comprise the system. Auxiliary storage used to support basic system requirements has three logical areas as follows:

- System data set storage area
- Paging data sets for backup of all pageable address spaces
- Swap data sets used for LSQA pages and private area pages that are swapped in with the address space (also called the working set)

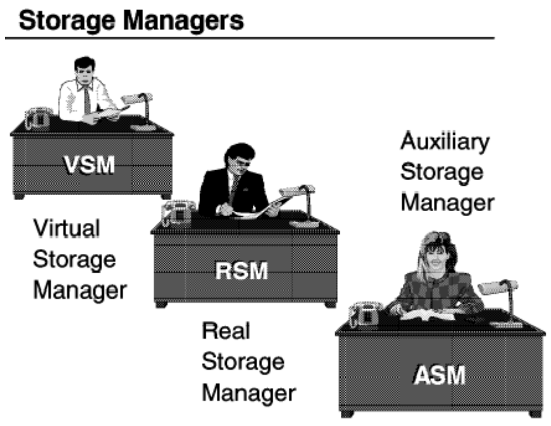
Storage managers

In an OS/390 system, storage is managed by the following storage components managers:

Real The real storage manager (RSM) controls the allocation of central storage frames during initialization and execution of user and system function

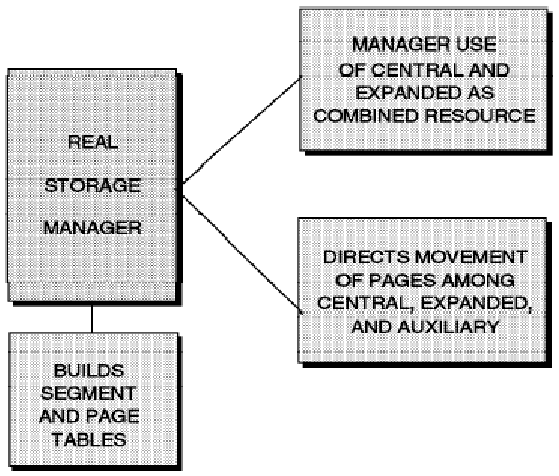
Some RSM functions:

· Allocate central storage to satisfy GETMAIN requests for SQA and LSQA
· Allocate central storage for page fixing
· Allocate central storage for an address space that is to be swapped in
· Allocate and initialize control blocks and queues related to expanded storage



RSM acts together with the auxiliary storage manager to support the virtual storage concept, and with VSM to ensure that a GETMAINed page will be backed up with a real storage frame. Furthermore, RSM establishes many services to other components and application programs to manipulate the status of pages and frames.

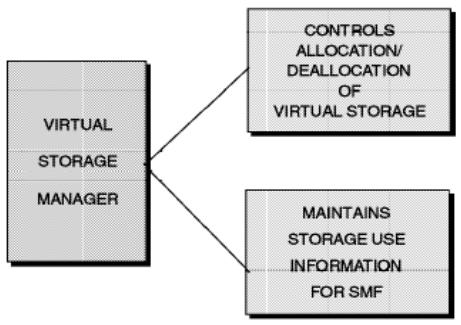
Real Storage Manager


REAL
STORAGE
MANAGER
MANAGER USE OF CENTRAL AND EXPANDED AS COMBINED RESOURCE
DIRECTS MOVEMENT OF PAGES AMONG CENTRAL, EXPANDED, AND AUXILIARY
BUILDS SEGMENT AND PAGE TABLES

If there is storage above 16 megabytes, RSM allocates central storage locations above 16 megabytes for SQA, LSQA, and the pageable requirement of the system. When non-fixed pages are fixed for the first time, RSM: Ensures that the pages occupy the appropriate type of frame. Fixes the pages and records the type of frame used

Virtual Each installation can use virtual storage parameters to specify how certain virtual storage areas are to be allocated. These parameters have an impact on central storage use and overall system performance.

Virtual Storage Manager


VIRTUAL
STORAGE
MANAGER
CONTROLS ALLOCATION/ DEALLOCATION OF VIRTUAL STORAGE
MAINTAINS STORAGE USE INFORMATION FOR SMF

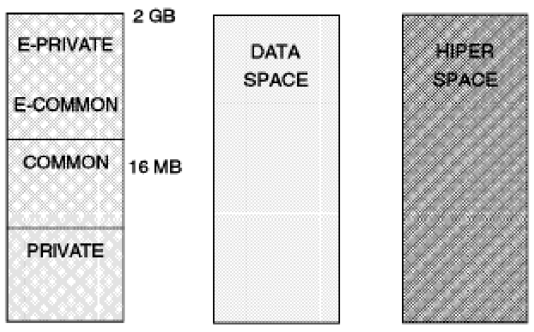
Virtual storage is managed by the virtual storage manager (VSM). Its main function is to distribute the virtual storage among all requests. Virtual storage is requested with the GETMAIN or STORAGE OBTAIN macro and returned to the virtual storage manager with the FREEMAIN or STORAGE RELEASE macro.

Auxiliary The auxiliary storage manager code controls the use of page and swap data sets. As a system programmer, you are responsible for:
· Page and swap operations
· Page and swap data set sizes
· Space calculation
· Performance of page and swap data sets
· Estimating the total size of the paging data sets

Concept of Virtual Storage

Virtual storage is normally larger than main storage (called real storage in OS/390). The size of real storage depends on the CPU type. In a computing system without virtual storage, a program cannot be executed unless there is enough storage to hold it. In addition the complete storage used is allocated until it is finished.

Virtual Storage



An OS/390 program resides in virtual storage and only parts of the program currently active need to be in real storage at processing time. The inactive parts are held in auxiliary storage, DASD devices, called page data sets. An active virtual storage page resides in a real storage frame. An inactive virtual storage page resides in a auxiliary storage slot. Moving pages between frames and slots is called paging.

Estimating Virtual Storage

Estimating the virtual storage allocated at an installation is important primarily because this storage must be backed up by central storage in some ratio (for example, 25%). This backup storage contributes significantly to an installation's total central storage requirements.

Virtual storage must also be backed up by expanded storage or auxiliary storage. Each installation can use virtual storage parameters to specify how certain virtual storage areas are to be allocated. These parameters have an impact on central storage use and overall system performance.

Virtual Storage Address Space

A two-gigabyte virtual storage address space is provided for:

- The master scheduler address space
- JES
- Other system component address spaces, such as allocation, system trace, system management facilities (SMF), and dumping services
- Each user (batch or TSO/E).

Address space storage

Estimating the virtual storage allocated at an installation is important primarily because this storage must be backed up by central storage in some ratio (for example, 25%). This backup storage contributes significantly to an installations total central storage requirements.

Virtual storage must also be backed up by expanded storage or auxiliary storage. Each installation can use virtual storage parameters to specify how certain virtual storage areas are to be allocated. These parameters have an impact on central storage use and overall system performance. The following overview describes the function of each Virtual storage area.

The system uses a portion of each virtual address space. Each virtual address space consists of:

- The common area below 16 MB
- The private area below 16 MB
- The extended common area above 16 MB
- The extended private area above 16 MB

The common area contains system control programs and control blocks. The following storage areas are located in the common area:

- Prefixed storage area (PSA)
- Common service area (CSA)
- Pageable link pack area (PLPA)
- Fixed link pack area (FLPA)
- Modified link pack area (MLPA)
- System queue area (SQA)
- Nucleus, which is fixed and nonswappable

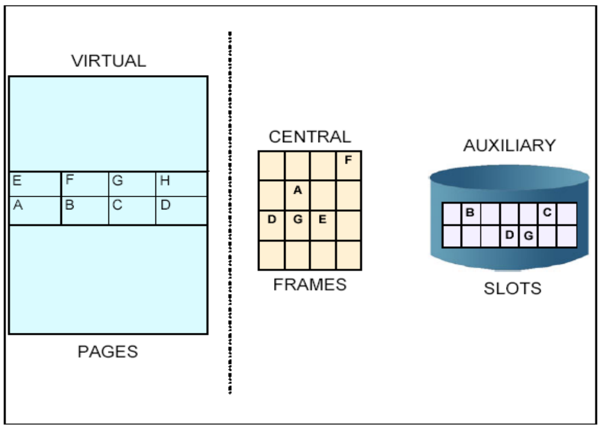
Each storage area in the common area (below 16 MB) has a counterpart in the extended common area (above 16 MB) with the exception of the PSA.
Each address space uses the same common area. Portions of the common area are paged in and out as the demands of the system change and as new user jobs (batch or time-shared) start and old ones terminate.

The private area contains:

- A local system queue area (LSQA)
- A scheduler work area (SWA)
- Subpools 229, 230, and 249 (the authorized user key area)
- A 16 KB system region area
- Either a V=V (virtual = virtual) or V=R (virtual = real) private user region for running programs and storing data

Except for the 16 KB system region area and V=R user regions, each storage area in the private area below 16 MB has a counterpart in the extended private area above 16 MB. Each address space has its own unique private area allocation. The private area (except LSQA) is pageable unless a user specifies a V=R region. If assigned as V=R, the actual V=R region area (excluding SWA, the 16 KB system region area, and subpools 229, 230, and 249) is fixed and nonswappable.

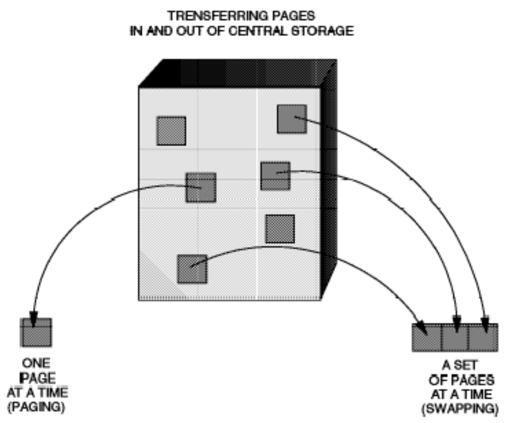
Paging and Swapping



The paging and swapping controllers of the auxiliary storage manager (ASM) attempt to maximize I/O efficiency by incorporating a set of algorithms to distribute the I/O load as evenly as is practical. In addition, every effort is made to keep the system operable in situations where a shortage of a specific type of page (swap) space exists.

RSM uses expanded storage as an extension of central storage. When a page is to be removed from central storage, RSM first considers moving it to expanded storage instead of auxiliary storage. When a page that is needed is not in central storage, RSM first checks expanded storage for the page. If the page is in expanded storage, RSM synchronously retrieves the page. If the page is not in expanded storage, RSM calls ASM to schedule asynchronously the paging I/O to retrieve the page from auxiliary storage.

Paging and Swapping


TRENSFERRING PAGES
IN AND OUT OF CENTRAL STORAGE
ONE PAGE AT A TIME (PAGING)
A SET OF PAGES AT A TIME (SWAPPING)

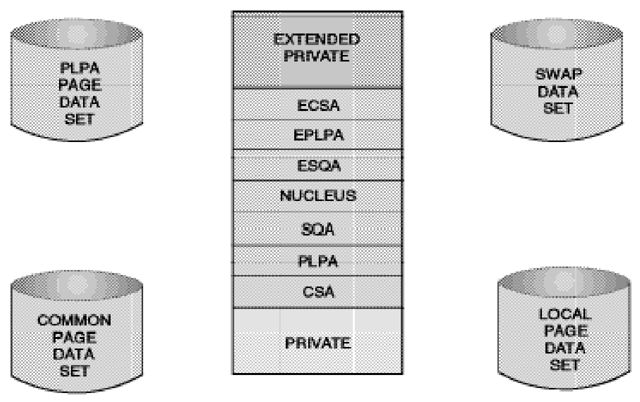
When contention for expanded storage increases, the system removes pages from expanded storage to free expanded storage frames. RSM first moves the pages from expanded storage to central storage. RSM then calls ASM to schedule the paging I/O necessary to send these pages to auxiliary storage. This process is called migration. Migration completes when the pages are actually sent to auxiliary storage.

RSM is responsible for reclaiming the central storage allocated to an address space when the address space is to be swapped out of central storage. RSM is also responsible for building the control structures necessary to efficiently swap the address space back into central storage. When an address space is swapped out of central storage, RSM works with SRM to identify the working set pages that will be swapped back into central storage.

Paging

To page efficiently and expediently, ASM divides the pages of the system into classes, namely PLPA, common, and local. Contention is reduced when these classes of pages are placed on different physical devices. Although the system requires only one local page data set, performance improvement can be obtained when local page data sets are distributed on more than one device, even though some devices may be large enough to hold the entire amount of necessary page space. The PLPA and common page data sets are both required data sets, and there can be only one of each. Spillage back and forth between the PLPA and common page data sets is permissible, but, in the interest of performance, only spilling from PLPA to common should be tolerated.

Auxiliary Data Sets



Paging operation

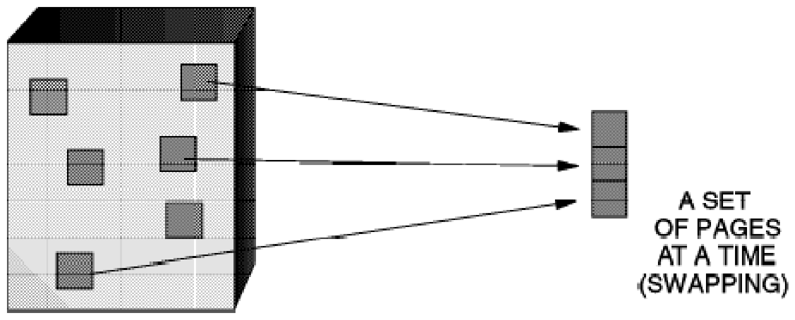
The pages data sets are:

- **PLPA page data set**: this contains pageable link pack area. (There is only one.)
- **Common page data set**: this contains the non-PLPA virtual pages of the system common area.
- **Local page data set:** this contains the private area (address space) pages, and space for any VIO data sets. Local page data sets can be dynamically added to and deleted from the paging configuration without re-IPLing the system.(There are one or more.)
- **Duplex page data set**: this is an optional data set, used when duplexing of the common area is requested.

Swapping

Swapping is the primary function used by SRM to exercise control over distribution of resources and system throughput. Using information specified by the installation through IPS and OPT parameters, and system status information that is periodically monitored, SRM determines which address spaces should have access to system resources.

Swapping Operations



In addition to the swapping controls described in the following text, SRM also provides an optional swap-in delay to limit the response time of TSO/E transactions.
There are several reasons for swapping. Some swaps are used for control of domains and the competition for resources between individual address spaces within a domain, while others provide control over system-wide performance and help increase the throughput.
ASM sends LSQA and private area working set pages to swap data sets as long as the data sets are defined and contain free space. A swap data set consists of groups of 4096-byte slots called swap sets. Each swap set consists of 12 contiguous slots. Swap data sets use only one seek per swap set. To ensure this seek efficiency, ASM prevents the swap set from crossing cylinder boundaries and uses the direct access device multi-track feature.

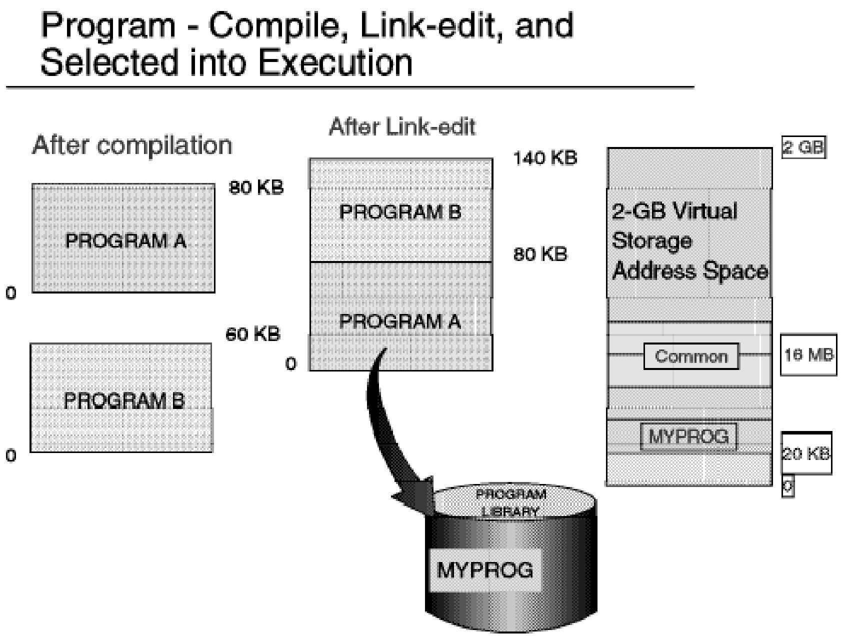
If ASM finds no swap sets, it uses a local page data set.

ASM frees swap sets immediately upon swap-in; that is, swap pages are valid on the swap data set only for that period between swap-out and swap-in.

Program execution

An OS/390 system may appear to be one big block of code that drives your CPU. Actually, OS/390 is a complex system comprised of many different smaller blocks of code. Each of those smaller blocks of code performs a specific function within the system.

Each system function is composed of one or more load modules. In an OS/390 environment, a load module represents the basic unit of machine-readable executable code. Load modules are created by combining one or more object modules and processing them with a link-edit utility. The link-editing of modules is a process that resolves external references and addresses. The functions on your system, therefore, are one or more object modules that have been combined and link-edited.



Processor Storage

- Storage Addressing
- Paging and Swapping
- Data Spaces and Hiperspaces

CENTRAL STORAGE

- Central Storage (also called main storage) is where programs, or portions of programs, must reside while they are executing.
- Central Storage has a continuous range of address from zero to the maximum address available on a particular system. The size of central storage is variable from 16 MB, on similar systems to 1 GB (1024 MB) on the largest systems.
- Central Storage is managed by both the hardware and the software in 4K (4096 Byte) pieces called frames (hardware) or pages (software). As we'll see later, many system functions have been implemented based on the principle that storage is divided into 4K pieces.
  1. Dynamic Address Translation (DAT)
  2. Storage Protection
  3. Expanded Storage
  4. Paging and Swapping

A PROGRAM'S ADDRESS RANGE AFTER COMPILATION

- The output from a language translator (compiler) is an object module.
- The addresses in an object module are related to a zero origin.
- An object module is a relocatable format which means, that it contains address constants that can be altered later to make up for a change in origin.

A PROGRAM'S ADDRESS RANGE AFTER LINK EDIT

- Before an object module can be executed, it must be processed by the linkage editor. The output of the linkage editor is a load module.
- The addresses in a load module are related to a zero origin.
- Multiple object modules or load modules can be combined into a single load module by the linkage editor. In our example, Program B has been relocated to an origin at the end of Program A.
- A load module is in a format that can be relocated and mapped into virtual storage by an MVS system program called Program Fetch.

PROGRAM IN CENTRAL STORAGE (PRIOR TO VIRTUAL STORAGE)

- For the processor to execute a program instruction, the program instruction and the data it references must be in central storage.
- Prior to the implementation of virtual storage, the entire program had to be in central storage while its instructions were executing.
- The program was read into central storage wherever contiguous space was available. The addresses in the program were relocated (adjusted) to correspond to the area of central storage being occupied by the program.

PROGRAM IN CENTRAL STORAGE (WITH VIRTUAL STORAGE)

- With the implementation of virtual storage, the entire program to which an instruction belongs does not need to be in central storage. Only the instruction to be executed, along with the data areas that instruction references, need to be in central storage.
- However, the addressing scheme used by the program must be maintained. The address space preserves the addressing scheme of the program using contiguous virtual addresses while providing the means for dividing the program into 4K pieces that will occupy non-contiguous frames of central storage.
- The Dynamic Address Translation (DAT) hardware converts the virtual addresses to real addresses when the instruction executes.

ADDRESS SPACES AND CENTRAL STORAGE

- A 2 gigabyte range of virtual addresses is called an address space. An address space provides the capability for a program to address upto 2 gigabytes of virtual storage. An address space represents a user in the system.
- Frames of central storage will be occupied by pages from many different address grams mapped into that address space. The address space addresses are virtual addresses.
- As a program executes, the virtual addresses will be converted to real addresses.
- The system area, called the common area, maps the executable MVS code and the control blocks and work areas needed by all address spaces in the system. The common area is mapped around the 16 megabyte line.

REGISTERS

REGISTERS AND CENTRAL STORAGE

- If two processors are sharing central storage. Each processor will be executing instructions independently of the other. The instruction that either one is executing at any point in time will be associated with a program represented by one address space.
- The association of central processor with a particular program is accomplished by a set of registers including a special kind of register known as the Program Status Word (PSW). There is a set of these registers and a PSW for each processor in the system.
- While a program is executing on a processor, the program makes use of these registers and the PSW to control the execution of that program. Whenever the program is interrupted, the contents of these registers and the PSW are saved in central storage so that they can be restored when the program later regains control of the processor.

REGISTERS

- There are 16 general registers (sometimes called general purpose registers) that may be used by programs as accumulators in general arithmetic operations or as base registers or index registers when referencing data in virtual storage.
- There are 16 access registers (on ESA capable processors) that are used when accessing data spaces or data in other address spaces.
- There are 16 control registers that are used by the system software to provide special control information to various system facilities such as DAT.
- The Program Status Word (PSW) is a special hardware register that works in conjunction with the control registers to govern exactly how each instruction is executed.

In addition to the registers, there are four floating-point registers associated with each CPU that are available for floating-point operations.

PSW FUNCTION

- Each central processor will be governed by its own PSW. The PSW contains information required for the execution of the currently active program.
- In general, the PSW is used to control instruction sequencing and to indicate the status of a central processor in relationship to the program currently being executed.
- The contents of the PSW are manipulated by the system software on behalf of application programs.
- When an interrupt occurs, the current contents of the PSW are stored and new values are loaded into the PSW. This gives control of the central processor to another program.

INSTRUCTION ADDRESS

- The PSW is 64 bits in length.
- Bits 33 to 63 of the PSW form the instruction address. This address designates the location of the leftmost byte of the next instruction to be executed.
- The address in the PSW is typically a virtual address. It will be converted to a real address in central storage by DAT (Dynamic Address Translation).

ADDRESSING MODE

- Bit 32 of the PSW controls the size of the addresses for the next sequential instruction.
- When the bit is zero, 24-bit addressing is specified.
- When the bit is one, 31-bit addressing is specified.
- This is what provides the compatibility between older programs written prior to the availability of MVS/XA and newer programs that take advantage of the 2 gigabyte address range.
- Typically, this bit has a constant value for the execution of an entire program or program module. MVS sets this bit depending on whether the program was identified as a 24-bit or a 31-bit program when it was compiled or link edited.

SUPERVISOR / PROBLEM STATE

- Many of the hardware instruction that a program can execute are very specialized ones that are primarily used by MVS or by a system type program that functions as an extension to MVS.
- The specialized instructions are designated as privileged instructions and their execution is controlled by bit 15 of the PSW.
- When bit 15 is zero, the CPU is in the supervisor state. In the supervisor state, all System/370 instructions, including the privileged ones, may be executed on the processor.
- When bit 15 is one, the CPU is in the problem state. In the problem state, only those instructions that provide meaningful information to application programs and that cannot affect gysystem integrity are valid; such instructions are called unprivileged instructions.
- Application programs usually run in problem state. Supervisor state is intended for use only by the MVS control program and other system routines.

IBM-Mainframe Computers based on Models-discrete IC CPUs(1964-2009)

• IBM 1130	High-precision scientific computer, 1965
• IBM 2020	System/360 Model 20 Central Processing Unit, almost a 360: 1966
• IBM 2067	System/360 Model 67 Central Processing Unit, mid range 360, multi- Processor with virtual memory (DAT)
• IBM 2095	System/360 Model 95 Central Processing Unit, high range 360
• IBM 3033	System/370 multiprocessor complex, high range, 1977
• IBM 3090	System/370 mainframe; high range; J series supersedes series. Models: 150, 150E, 180, 200 (1985), 400 2-way (1985), 400 4-way (1985), 600E (1987), 600S (1988). A 400 actually consists of two 200s mounted together in a single frame. Although it provides an enormous computing power, some limits, like CSA size, are still fixed by the 16MB line in MVS.
• IBM 3165	System/370 Model 165 Central Processing Unit; mid range; without virtual memory [DAT] unless upgraded to 165-II
• IBM ES/9370	System/370 mainframe partly replaced IBM 8100 low range 1986
• IBM ES/9000	Family of System/390 mainframes; 1990
• IBM 2064	zSeries z900 number collision with earlier System/360-64 2000
• IBM 2066	zSeries z800; less powerful variant of the z900
• IBM 2084	zSeries z990; successor of larger z900 models

- IBM 2086 zSeries z890; successor of the z800 and smaller z900 models; 2004
- IBM 2094 System z9 Enterprise Class (z9 EC); initially known as z9-109; 2005
- IBM 2096 System z9 Business Class (z9 BC); successor to z890; 2006
- IBM 2097 System z10 Enterprise Class (z10 EC) introduce on february 26,2008.
- IBM 2098 System z10 Business Class (z10 EC) introduce on October 21,2008.

Mainframe Operating systems

- AIX, IBM's proprietary UNIX OS (Advanced Interactive eXecutive)
- BPS/360 (Basic Programming Support/360)
- BOS/360 (Basic Operating System/360)
- DOS/360 (Tape Operating System/360)
- DOS/360 (Disk Operating System/360)
- DOS/VS (Disk Operating System/Virtual Storage - 370)
- DOS/VSE (Virtual Storage Extended - 370, 4300)
- VSE/ESA (Virtual Storage Extended/Enterprise System Architecture)
- DPCX (Distributed Processing Control eXecutive)
- DPPX (Distributed Processing Programming eXecutive)
- TSS/360 (Time Sharing System, a failed predecessor to VM/CMS, intended for the IBM System/360 Model 67)
- OS/360 (Operating System/360)
- OS/MVT (Operating System - Multiprogramming w. Variable Tasks/360)
- OS/MVS (Operating System - Multiple Virtual Systems for IBM/360)
- OS/VS1 (Operating System - Virtual Storage 1) for the IBM/370.
- MVS/370 (Multiple Virtual Systems/370 - OS/MVS + Virtual Storage)
- MVS/XA (Multiple Virtual Systems - Extended Architecture)
- MVS/ESA (Multiple Virtual Systems - Enterprise System Architecture))
- OS/390, now z/OS (Zero down time/OS)
- VM/CMS, now z/VM (Virtual Machine/Conversational Monitor System)
- VM/ESA (Virtual Machine/Enterprise System Architecture)
- 4690 OS
- Z/OS Operating System

Introduction to Z/Os

Z/OS architecture includes all of ESA/390 and the following extensions:

- IBM’s high end server OS.
- Robust IBM e-server Z series mainframe OS for e-business.

- Highly secure scalable high performance base to deploy Internet + Java technology-enabled applications.
- Z/OS takes advantage of latest open s/w technologies like EJB /XML /HTML /UNICODE/IP network etc.
- Cryptographic services and distributed print services, storage management, and Parallel Sysplex availability.
- 64 bit General Registers and Control Registers.
- 64 bit addressing mode in addition to 24 bit and 31 bit addressing modes of ESA/390 which are carried forward to Z/OS architecture.
- The Program Status Word (PSW) is expanded to 16 bytes to contain larger instruction address.
- Upto three levels of DAT tables called Region Tables for translating 64 bit virtual address.
- Trimodal addressing as against bimodal addressing.
- Ability to switch between 24, 31 and 64 bit addressing.

Logically a system consists of the main storage, one or more CPUs and channel subsystems.

Addressing : Absolute, Real and Virtual

Address Translation converts Virtual Addressing to Real Addressing and Prefixes converts Real Addressing to Absolute Addressing.
A 24 bit or 31 bit virtual address is expanded to 64 bits by appending 40 or 33 zeroes on the left before it is translated by means of DAT process and 24 and 31 bit absolute address is expanded to 64 bits before it is transformed by prefixing. A 24 or 31 bit absolute address is expanded to 64 bits before main storage is accessed.

Major features of the eServer zSeries family

- Based on z/Architecture (64-bit real and virtual addresses), as opposed to earlier ESA/390 (31-bit) used in S/390 systems
- ESA/390 applications are fully compatible with z/Architecture
- Offers up to 32 central processors (CPs) per frame (rack)
- Frames can be coupled in up to a 32-frame Sysplex, with each frame physically separated up to 100 kilometers
- Supports the z/OS, Linux on zSeries, z/VM, z/VSE, z/TPF, and MUSIC/SP operating systems
- Some models introduced multiple I/O channel subsystems (exceeding the previous 256 channel limit) and zAAPs
- In July 2005, IBM announced a new brand name System z9 using it to announce System z9-109 servers

SYSTEM Z/9

The System z9-109 Model S54, with up to 54 processing units (PUs), is reportedly capable of performing approximately 18,660,000,000 core instructions per second. A single S54 can typically process one billion or more business transactions per day double the throughput of its predecessor. The 54 PUs can be configured, or "characterized", for a variety of purposes including general purpose processing (CPs), zAAPs, zIIPs, IFLs, and ICFs.

The System z9 servers add on top of that:

- Up to 54 central processors (CPs) per frame
- zIIP engines.
- MIDAW
- Advanced Encryption Standard (AES) cryptography implemented in hardware

SYSTEM Z/10

Z/10-Enterprise Class:

Released on February 26, 2008, the System z10 Enterprise Class is available in five hardware models: E12, E26, E40, E56, and E64. Each are of the machine type 2097[2]. The Enterprise Class PU cores (four per chip) operate at speeds of 4.4 GHz, still (December, 2008) the highest clock speed of any processor with more than two cores per chip. The processors are stored in one to four compartments referred to as "books". Each book is comprised of a multi-chip module (MCM) of processing units (PUs) and memory cards (including multi-level cache memory). The number of PUs in each book is based upon the model number.

- A minimum of one CP, IFL, or ICF must ordered with every model.
- For each CP ordered, one zAAP and one zIIP may also be ordered.
- Optional SAPs are required only in some situations when using TPF/ESA or z/TPF.
- Memory figures refer to user-accessible memory. The z10 EC reserves 16GB for HSA .
- Sub-capacity (fractional) CP configurations are also available.

IBM System z10 BC Mainframe:

The IBM System z10 servers have many similarities to z9 servers but support more memory and can have up to 64 central processors (CPs) per frame. The full speed z10 processor's uniprocessor performance is up to 62% faster than that of the z9 server, according to IBM's z10 announcement.

Released on October 21, 2008, the z10 Business Class has only a single model: E10. Machine type is 2098. It has the same processor chip design and instruction set as the z10 EC but with higher manufacturing yields (3.5 GHz clock speed, one core per chip disabled) and lower cost processor packaging due to reduced cooling and reduced multi-chip shared cache needs. The z10 BC also introduced new, more efficient I/O packaging options. It is possible to configure a z10 BC without spare cores if desired, although such maximally configured z10s still fail gracefully in the unlikely event there's a core failure: the system will move any work from the failed core to surviving cores automatically, without operating system or software involvement, keeping all applications running, albeit at slightly reduced capacity if there are no spares remaining. The baseline model of the z10 EC has a reported price starting at $1,000,000 for a new system.

- For each CP ordered, one zAAP and one zIIP may also be ordered.
- Memory figures refer to user-accessible memory. The z10 BC reserves 8GB for HSA
- Sub-capacity (fractional) CP configurations are also available.

The Total Cost of Ownership is considerably reduced when transitioning to z/OS.e. At only a fraction of the cost of z/OS, z/OS.e makes it easier to run new workloads on the mainframe due to its exceptional robustness and functionality. z/OS.e and z800 together reduce the total cost of ownership of hardware, software, people, and environmentals making the combination very cost-effective for new applications.

Intelligent Resource Director

Intelligent Resource Director (IRD) is a new feature of the z/Architecture which extends the z/OS Workload Manager to work with PR/SM on zSeries servers to dynamically manage resources across an LPAR cluster. An LPAR cluster is the subset of the systems that are running as LPARs on the same CEC.
Based on business goals, WLM can adjust processor capacity, channel paths, and I/O requests across LPARs without human intervention. IRD assigns resources to the application; the application is not assigned to the resource. This capability of a system to dynamically direct resources to respond to the needs of individual components within the system is an evolutionary step. It enables the system to continuously allocate resources for different applications, and this helps to reduce the total cost of ownership of the system. IRD is made up of three parts that work together to respond to the demands of e-business:

- LPAR CPU Management
- Dynamic Channel Path Management
- Channel Subsystem Priority Queuing

WLM enhancements

- Initiator balancing - It’s a new function provided in WLM to rebalance the batch initiators across the system in a sysplex.
- Provides multi-level security.

- When operating in goal mode WLM has two sets of routines.
- Achieve transaction goals and
- Maximize the utilization of resources in the environment.

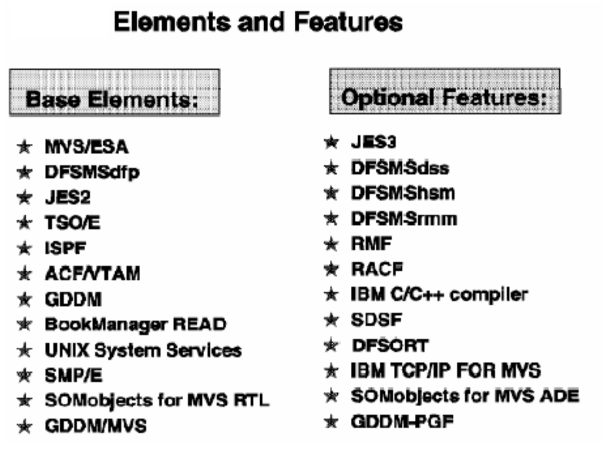
ELEMENTS AND FEATURES

Basic two types of elements are: Base elements & Optional elements

Base Elements

The z/OS system consists of base elements that deliver essential operating functions—in addition to the services provided by the BCP functions such as communications support, online access, host graphics, and online viewing of publications.

When we order z/OS or z/OS.e, we receive all of the base elements. However, with z/OS.e, some base elements are not functional or not licensed for use, or both.



Elements and Features

Base Elements:	Optional Features:
★ MVS/ESA	★ JES3
★ DFSMSdfp	★ DFSMSdss
★ JES2	★ DFSMShsm
★ TSO/E	★ DFSMSrmm
★ ISPF	★ RMF
★ ACF/VTAM	★ RACF
★ GDDM	★ IBM C/C++ compiler
★ BookManager READ	★ SDSF
★ UNIX System Services	★ DFSORT
★ SMP/E	★ IBM TCP/IP FOR MVS
★ SOMobjects for MVS RTL	★ SOMobjects for MVS ADE
★ GDDM/MVS	★ GDDM-PGF

In addition to the base elements, z/OS has optional features that are closely related to the base features. The optional features are orderable with z/OS or z/OS.e and provide additional operating system functions.

/4Optional features are unpriced or priced:

- **Unpriced features** are shipped only if we specifically order them.
- **Priced features** are always shipped.

IBM enables the priced features you ordered and disables the priced features you did not order. Later on, if you decide to use them, you can notify IBM, and then you enable them dynamically (which is known as dynamic enablement). Dynamic enablement is done by updating SYS1.PARMLIB member IFAPRDxx.

Some optional features that support dynamic enablement are always shipped. Examples are JES3, DFSMSdss, and DFSMShsm. If these features are ordered as part of the z/OS system order, they are shipped as enabled in the system. If they are not ordered, they are shipped as disabled.
There are two classifications of elements in the OS/390 system: exclusive and nonexclusive.

- **Exclusive elements:**
  The functional level of an element or feature that can be ordered only as part of the OS/390 package, and is not available as an independent element or feature anywhere else.
- **Nonexclusive elements:**
  Those elements or features included in the OS/390 package that are also orderable as independent products, at the same functional level, from the MVS product set.

Command to display the registered products in the system

```
D PROD,REGISTERED
```

JES (JOB ENTRY SUBSYSTEM)

JES2 or **JES3**: BCP uses a job entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by BCP, and control their output processing. In a simple view, z/OS divides the management of jobs and resources between the JES and the base control program of z/OS. JES2 manages jobs before and after running the program; the base control program manages them during processing of their output, then JES2 purges them from the system. For an installation that has more than one processor in a configuration, there are noticeable differences in how JES2 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. JES2 is an exclusive, base element and JES3 is an exclusive, optional element.

SMS (STORAGE MMANAGEMENT SUBSYSTEM)

DFSMS (Data facility storage management subsystem) provides a range of automated data and space management functions that eliminate,simplify, and automate tasks normally done by users or a storage administrator, improve storage space use; control external storage centrally; and let the storage administrator manage storage growth. DFSMS is a family of products, each one having specific functions.

1. DFSMSdfp. Data Facility Product (dfp) is in charge of:

   - Space allocation
   - Access methods
   - Support for the storage hardware to balance performance throughout the system
   - Program management tools
   - Applying the storage management policy throughout the SMS constructs

2. DFSMSdss, the Data Set Services (dss), is a high speed and high capacity utility used to move data from disk to disk (copy) or disk to tape (dump) very quickly and efficiently. It performs both logical dumps (to copy data) and physical dumps(to copy track images). DFSMSdss is a external interface to Concurrent Copy.

3. Hierarchical Storage Manager, DFSMShsm, provides the following functions:
   - Full volume dump and restore
   - Policy-based space management
   - Automatic and periodic data backup data set and volume levels
   - Data set backup (full and incremental)
   - Data set recovery
   - Aggregate backup and recovery support (ABARS)
   - Automatic or user-initiated migration and recall data sets from HSM database (disk or tape)

4. DFSMSrmm (removable media manager) :

   A removable media library, • System-managed tape libraries. Enterprise Automated Tape Library (3494) and IBM TotalStorage Virtual Tap Servers (VTS). Non-system-managed tape libraries or traditional tape libraries Storage locations that are on-site and off-site. Storage locations defined as home locations.
5. DFSMStvs (Transactional VSAM Services) enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently. Except for DFSMSdfp, which is a base element, all others features are exclusive and optional.

RACF (Resource access control Facility)

The RACF product is a component of the z/OS Security Server and works together with the existing system features of z/OS to provide improved data security for an installation.

RACF helps meet the need for security by providing:

- Flexible control of access to protected resources
- Protection of installation-defined resources
- Ability to store information for other products
- Choice of centralized or decentralized control of profiles
- An ISPF panel interface
- Transparency to end users
- Exits for installation-written routines

Resource Management Facility (RMF)

Many different activities are required to keep your z/OS running smoothly, and to provide the best service on the basis of the available resources and workload requirements. The console operator, the service administrator, the system programmer, or the performance analyst does these tasks.
RMF is the tool (online and batch reports) that helps each of these people do the job effectively.

RMF gathers data using three monitors:

- Short-term data collection with Monitor III
- Snapshot monitoring with Monitor II
- Long-term data gathering with Monitor I

Data is gathered for a specific cycle time, and consolidated data records are written at a specific interval time. The default value for data gathering is one second and for data recording 30 minutes. You can select these options according to your requirements and change them whenever the need arises.

Monitor I collects long-term data about system workload and resource utilization, and coversall hardware and software components of your system: processor, I/O device and storage activities and utilization, as well as resource consumption, activity, and performance of groups of address spaces.

System Management Facility (SMF)

System management facility (SMF) collects and records system and job-related information that your installation can use in:

- Billing users
- Reporting reliability
- Analyzing the configuration
- Scheduling jobs
- Summarizing direct access volume activity
- Evaluating data set activity
- Profiling system resource use
- Maintaining and auditing system security

SMF formats the information that it gathers into system-related or job-related records. System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information on the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.
An installation can provide its own routines as part of SMF. These routines will receive control either at a particular point as a job moves through the system, or when a specific event occurs.
For example, an installation-written routine can receive control when the CPU time limit for a job expires or when an initiator selects the job for processing. The routine can collect additional information, or enforce installation standards.

Virtual Lookaside Facility (VLF)

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 a data space 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. The data is known as a data object. Data objects should be small to moderate in size, named according to the VLF naming convention, 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 each installation must weigh when planning for the resources required by VLF.
When you define the LLA class to VLF, and start VLF, the most active modules from LLA-managed libraries are staged into the DCSVLLA data space managed by VLF. You obtain the most benefit from LLA when you have both LLA and VLF functioning, so you should plan to use both.

Real Storage Manager (RSM)
Controls the allocation of central storage during system.initialization, and pages in user

or system functions during execution. Some specific RSM functions are:

- Allocate central storage to satisfy GETMAIN requests for SQA and LSQA.
- Allocate central storage for page fixing.
- Allocate central storage for an address space that is to be swapped in.
- Allocate and initialize control blocks and queues related to expanded storage.

Virtual Storage Manager (VSM)

Each installation can use virtual storage parameters to specify how certain virtual storage areas are to be allocated. These parameters have an impact on central storage use and overall system performance.

Auxiliary Storage Manager (ASM)

The auxiliary storage manager code controls the use of page and swap data sets. As a system programmer, you are responsible for:

- Page and swap operations
- Page and swap data set sizes
- Space calculation
- Performance of page and swap data sets
- Estimating the total size of the paging data sets

Transmission Control Protocol/Internet Protocol (TCP/IP)

Formatted: Portuguese (Brazil)

Its a set of protocols and applications that allow you to perform certain computer functions in a similar manner independent of the types of computers or networks being used. When you use TCP/IP, you are using a network of computers to communicate with other users, share data with each other, and share the processing resources of the computers connected to the TCP/IP network.

A computer network is a group of computer nodes electronically connected by some communication medium. Each node has the hardware and the programs necessary to communicate with other computer nodes across this communication medium. The node can be a PC, workstation, microcomputer, departmental computer, or large computer system.

UNIX System Services

z/OS UNIX interacts with the following elements and features of z/OS:

- C/C++ Compiler, to compile programs
- Language Environment, to execute the shell and utilities
- Data Facility Storage Management Subsystem
- z/OS Security Server
- Resource Measurement Facility (RMF)
- System Display and Search Facility (SDSF)

- Time Sharing Option Extensions (TSO/E)
- TCP/IP Services

BPXOINIT is the started procedure that runs the initialization process. The OMVS address space is now started automatically at IPL by means of the OMVS statement in the IEASYSxx SYS1.PARMLIB member.

Introduction to TSO

Time-Sharing Option (TSO) is an option of the MVS operating system that allows users to interactively share computer time and resources.

It is a tool with which you can
- Develop and maintain programs in languages such as COBOL, REXX, PL/I, etc.
- Create an office environment
- Process data
- Communicate with other TSO users
- Access the MVS operating system

You can use TSO in
- ISPF/PDF : Interactive System Productivity Facility and its Program Development Facility. This provides interactive panels for maintaining libraries of information in TSO and allows call, browse, etc.
- Line Mode : Commands can be entered and executed after the Ready message is displayed.

Logon and Logoff Procedure

Logon

Type LOGON and press the ENTER key. The following message is displayed on your terminal. ENTER USERID : Type your "user-id" and press the ENTER key. A LOGON panel appears on the screen. (As an alternative, you can also type LOGON followed by your user-id).
Type in your "password". If you are logging on to the screen for the first time, your password will be same as your "user-id", and the following message will be displayed.
CURRENT PASSWORD EXPIRED, ENTER NEW PASSWORD

Rules for framing your password:
The length of your password can be between 6 and 8 characters.
Your password is valid for 30 days, after which you will be prompted automatically to enter a new password. The password should be in the format of ALLLLLLL/AAAAAALL where A is any Alphabetic character and L is any Alphanumeric character.

A maximum of three attempts for typing a password is permitted, after which the user-id will be revoked. When changing the password, you cannot use the previous five passwords as your new password. Type in a new password (conforming to the rules mentioned above) in the field “NEW PASSWORD” and press the ENTER key. (Use the “TAB (->|)” key on the left side of your keyboard to switch across fields). You are prompted with the following message:

PLEAS REENTER NEW PASSWORD FOR VERIFICATION

Now, type in your new password in the “NEW PASSWORD” field, and press the ENTER key to validate your password. If you have typed in your new password correctly, and if it conforms to the rules, then the system will accept the password. The next time you logon, you can use the newly entered password.

After this, the system broadcast messages are displayed on the screen. Please read them carefully. After the broadcast is over, (***) is displayed at the bottom of the screen. Press the ENTER key and wait. After this, a panel with the header as “ISPF/PDF MAIN MENU” is displayed. You are now into ISPF.
Every screen you see in ISPF is called a “PANEL”
The F3 key on the keyboard takes you to the previous panel (if available)

Logoff
From the panel you are in, type ‘=X’ in the OPTION field on the top and press the ENTER key to take you back to the first panel (i.e. ISPF/PDF MAIN MENU). Type ‘X’ at the OPTION field of this panel and press the ENTER key.

There are two possibilities now

A blank screen, with a “READY” prompt is displayed. Now, type “LOGOFF” and press the ENTER key to terminate your TSO session. Or

A panel asking you to enter LIST and LOG data disposition options is displayed. Type ‘D’ against ‘PROCESS OPTIONS’ and press the ENTER key. This takes you to the ‘READY’ prompt and you can type ‘LOGOFF’ to terminate your TSO session.

Introduction to ISPF

Interactive System Productivity Facility provides user-friendly menus to interactively process your tasks. It provides dialogs using which, you can interact with the computer. It runs on TSO.
ISPF/PDF (Program Development Facility) is a dialog application that provides application development facilities. It provides interfaces to many system facilities through easy-to-use menus, relieving you of the need to know the specific command syntax of the interactive system you are using.

Keyboard Layout
The IBM Keyboard connected to your terminal is a 122 keys keyboard. It has 24 function keys (F1-F24), number, pad, navigation keys, special function keys, Reset key, Attention keys, etc. Each key has a unique function.

Function Keys
The Function Keys F1-F12 are called the Primary Keys and the Function Keys F13-F24 are called the Alternative Keys.

The PFSHOW Command
The PFSHOW <ON> command displays the functions assigned to all Function Keys. This command is very useful to a new user, since it displays the function assigned to each command key till the end of the session. The display can be toggled off by issuing the command PFSHOW OFF.

The KEYS Command

This command displays the current definitions for the function keys and also allows you to change the key definitions if necessary. To change the function of any function key, type the new function against the appropriate key. The changes are valid till the next change is made. When you logoff, the effect of the changes is lost.

Miscellaneous Keys
The RESET Key: Under certain circumstances, you may find that what you type may not be displayed. This happens if you try to type on inappropriate column positions like protected fields, or for many other reasons. You are then prompted with the ‘X<-o->’ Symbol on the bottom of the screen. Press the RESET key to proceed in such cases.

The ATTENTION Key: If your program runs in a loop, or if you deliberately want to stop any processing, press the Attention Key – PA1

Introduction to Data Sets
Storage of any information under the MVS platform is in the form of DATASET. A dataset may be of the following types:

Partitioned Data Set
Physical Sequential Data Set
VSAM Data Set

A partitioned Dataset can be compared to a directory in DOS, which may contain many files (‘members’ as it is called in MVS terms). However, it cannot have sub-directories like DOS. A Physical Sequential Data Set can be compared to a simple file (in DOS).
Information is saved in a Dataset in the form of a record.

A record may have any of the formats mentioned below.
F – Fixed Length
V – Variable Length
U - Undefined Format
B – Blocked Records
A – ASA Printer Control Characters

F, V, or U is required. A record has a record length, in bytes.

Data Set Naming Conventions

The maximum length of a data set name is 44 characters. A data set name is divided into several levels of qualifiers viz. First level qualifier, second level qualifier and so on.

For ex let us consider the data set name as,
'trgi01.trg1.entl'
Here, the first level qualifier is 'trgi01'
Second level qualifier is 'trg1'
Third level qualifier is 'entl'

Each qualifier is separated by a period
Each qualifier should start with a letter of the alphabet
Each qualifier can have a maximum length of eight alphanumeric characters
There can be up to 22 levels of qualifiers
There should not be any blanks in the data set name
The first level qualifier must be your 'user-id' only.

Standards
The third level qualifier must be named properly to describe the members of the data set.
For ex.

'trgi01.trg.end'	–	For JCL Members
'trgi01.trg.cobol'	–	For COBOL Source Programs
'trgi01.trg.load'	–	For Executable Load Modules
'trgi01.trg.exe'	–	For REXX Programs

DASD

The Direct Access Storage Device is a high capacity, high-speed auxiliary storage unit. It has a storage capacity of 40 GB. The total space is spread across 28 volumes. Each volume is identified by a unique volume serial name.

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 or by user-id, without knowing where the data set resides.

VTOC

Volume Table of Contents has a group of records that contain the data set names and attributes of each data set on the DASD Volume. Every volume has a VTOC.

Allocation of Data Sets

Allocation of data sets can be done in the foreground using the 3.2 option from the ISPF/PDF Primary Menu. Both Partitioned and Physical Sequential Data Sets can be allocated. Another panel is now displayed, where you have to provide the dataset information.

Information about the dataset name, the volume in which the dataset has to be created, the type of data set (Partitioned (PO) or Physical Sequence (PS)), space required for data set (in terms of number of cylinders or tracks or Bytes), maximum length of record, blocksize and the record format must be provided. The user is informed about the successful or unsuccessful attempt for creation.

If you mention SYSDA, the system allocates the data set in any volume where sufficient space is available. You are now authorized to create a data set in that volume. If you mention the number of directory blocks as 0, you can allocate a physical sequential data set. If the number of directory blocks is greater than 0, a PDS is created.
Help Message: For help on any panel press the F1 function key.

Editing Data Sets

Creating/Editing a Member of a PDS

Editing an existing or new member of a PDS or editing a PS can be done by choosing the option ‘2’ (Edit). The dataset name has to be specified by the user.

If the member name is not specified in the case of a PDS, you can choose the member you want to edit from a member list. You can do this by giving an ‘S’ against the member to select it. Alternatively, editing can also be done by choosing the ‘3.4’ option (DSLIST) and then giving a ‘c’ option in the command field for the appropriate data set.
A Member name can be a maximum of 8 characters in length. The first character should be alphabetic or any other national character. Your edit screen will appear and now you can edit the member/data set:

If you are not able to insert, give the command NULLS ON. After editing, save the changes by giving the SAVE command or press the F3 key to save and exit (END).

Primary Editing Commands
Some of the frequently used primary editing commands are listed below.

SAVE	- To save a data set/member and resume editing
FIND	- To find a specified string
CHANGE	- To find a specified string and replace it by some other string
SUBMIT	- To submit edit data as a job stream for background execution
END	- To save a data set/member and exit
CANCEL	- To abandon any changes done to the data set/member and exit
RESET	- To reset edit display
NUMBER	- To set number mode on or off
RENUM	- To set number mode on and renumber the data
UNNUM	- To set number mode off and blank out sequence numbers
LOCATE	- To display a particular line in the data
PROFILE	- To display the edit profile
RECOVERY	- To set recovery mode on or off
UNDO	- To undo the changes made after the previous SAVE
NULLS	- To set insert mode on or off

Line Commands
Three of the most commonly used line commands are I (insert), D (delete) and R (repeat). Together they provide the most basic line editing functions.

I – Insert line (to insert one or more lines of new data)
D – Delete line (to delete one line, several lines, or a block of lines)
R – Repeat line (to repeat a single line one or more times, or to repeat a block of lines one or more times)

Move/Copy Commands

The commands listed below are used to perform a move or copy operation.
1. **To move or copy lines within the data that is being edited:**

1. Use M or C line commands to specify the source
2. Use A, B, or O line commands to specify the destination

2. To move or copy data in, from a data set or member:

1. Use COPY or MOVE primary commands to specify the source
2. Use A or B line commands to specify the destination

3. To move or copy data out, into a data set or member:

1. Use M or C line commands to specify the source

2. Use CREATE or REPLACE primary commands to specify the destination

Copy Operation
Use C or CC to copy one or more lines. After a line has been copied it will exist in both its origin and new location.

1. C – identifies a single line that is to be copied
2. C3 – identifies the first of 3 (or any number) of lines to be copied
3. CC – identifies the first and last lines of a block of lines to be copied.

The destination of the line(s) to be copied can be specified using:

B or O line commands if the line(s) are to be copied to another place in the data being edited. CREATE or REPLACE primary commands if the lines are to be copied to a sequential data set (REPLACE) or to a member of a PDS.

MOVE Operation
Use M or MM to move one or more lines. After a line has been moved it will exist only in its new location.

1. M – identifies a single line that is to be moved
2. M3 – identifies the first 3 (or any number) lines to be moved
3. MM – identifies the first and last lines of a block of lines to be moved.

The destination of the line(s) to be moved can be specified using:
B or O line commands if the line(s) are to be moved to another place in the data being edited. CREATE or REPLACE primary commands if the lines are to be moved to a sequential data set (REPLACE) or to a member of PDS

Destination Specification (AFTER, BEFORE, OVERLAY)

The ‘A’ (after) line command identifies the destination where data is to be moved or copied after a line in the data.
If a line or lines are to be moved or copied from another part of the data, you can specify a number of the ‘A’ command and the line or lines will be repeated ‘n’ times.
For EX A4 results in the source line being repeated four times.

The ‘B’ before line command identifies the destination where data is to be moved or copied before a line in the data.
If a line or lines are to be moved or copied from another part of the data, you can specify a number of the B command and the line or lines will be repeated ‘n’ times.

For EX B4 results in the source line being repeated four times.

Use the ‘O’ (overlay) command to indicate the target when you want to merge lines. Overlay is used in conjunction with the M or C line command.

O – identifies a single line that is to be overlaid
O3 – identifies the first of 3 (or any number of) lines to be overlaid
OO – identifies the first and last lines of a block of lines to be overlaid.

Move or copy overlay can be used to merge data from one or more source lines onto one or more destination lines.

The following rules apply:
Data is processed character by character
Only blank characters are overlaid
Source lines will be used repeatedly until all destination lines are processed.

Shifting Commands
Two kinds of shifting are provided. Column shifting shifts either to the left or the right without regard to the data contained in a line. Data shifting shifts data within the line, and is designed primarily for use with indented program code.

)	–	Shift right one or more columns
(	-	Shift left one or more columns
>	-	Shift data right one or more characters
<	-	Shift data left one or more characters

Use) or)) to shift columns right on one or more lines.

) 2	–	Identifies a line on which columns are to be shifted right by 2
) 5	-	Identifies a line on which columns are to be shifted right by 5
)) 2	-	Identifies the first and last lines of a block of lines on which columns are to be shifted right by 2
)) 3	-	Identifies the first and last lines of a block of lines on which columns are to be shifted right by 3

Text Handling Commands
Use the text handling commands when you are entering or modifying textual data. They are especially useful when used together. For EX use text split, enter a word or phrase, and then use text flow to reformat the paragraph.

TS – text split, to split a text line at the cursor position to allow insertion
TF – text flow, to flow text to the end of a paragraph
TE – text enter, to format the screen for power typing one or more text paragraphs
LC – lowercase, to change text from uppercase to lowercase
UC – uppercase, to change text from lowercase to uppercase

Miscellaneous Commands

COLS Command
Use COLS to display the column numbers

For EX

```
cols    -----+-----1-----+-----2-----+-----3-----+-----4-----+-----5----- etc.
```

A digit is displayed in the columns 10, 20, 30, etc. (i.e. 3 is in column 30). A ‘+’ is displayed between the digits to indicate columns 5, 15, 25, etc. To remove the column line from the display, use the D line command or the RESET primary command.

TABS Command
Use TABS to display the tab line. To change the tab positions, simply overtype on it when it is displayed.

For EX

```
tabs   ----------------------------- *                    *
```

The tabs line is used to define software, logical and hardware tab fields.

“*” is used to define a hardware or logical tab field
“-“ or “_” is used to define a software tab field
Any other character entered on the tabs line will be blanked out.

Edit Macros
An edit macro is a routine that uses ISREDIT services to ease the editing process in the ISPF environment. Some of the edit macros available at our installation are as follows:

CURSEDIT	-	A cursor driven edit macro
CUT	-	To cut lines from a data set/member
PASTE	-	To paste lines into a data set/member
GO	-	To invoke a Clist or Rexx that is being edited
SHOW	-	To display the panel whose definition is being edited
JOBCRD	-	To include a job card for the JCL
ONLY	-	To search for specified strings and display those lines
COLRPT	-	To copy, move or repeat contents of columns specified

Cut and Paste Macros

Cut Macro
‘CUT” is an ISPF/PDF edit macro to write lines from a data set/member to the user PROFILE pool for an inclusion later by the PASTE macro.

To run: Enter CUT on the COMMAND line and use the C or M line commands to select the lines to be cut. If the M line command is used, the lines will be deleted. A parameter of R or REPLACE can be specified to replace any previously CUT lines (that have not yet been pasted) with the newly selected lines. Otherwise, the selected lines will be added to any previously CUT lines.

An arbitrary limit of 1000 lines is set in the macro, but, this could be changed, by providing a new limit when the macro is called. For EX to process up to 2000 lines, enter CUT 2000 on the COMMAND line. Remember that each line is stored in your PROFILE.

Paste Macro
Paste is an ISPF/PDF edit macro to write lines from the user PROFILE pool into the current data set/member. This macro is used in conjunction with the CUT macro.
To run: Enter PASTE on the COMMAND line and use the A or B line command to specify where the lines are to be pasted. A parameter of K or KEP can be specified to prevent the macro from setting the profile variables to null after the lines have been pasted into the data set/member.

If line truncation occurs, the profile variables will not be set to null. When this occurs, edit a data set with a large enough record length or use CUT REPLACE to replace the lines in the PROFILE pool.

Labels and Ranges
It is possible for symbolic labels to be assigned to lines on the edit data display. These labels can be referenced by several of the edit primary commands. The most common among the uses of labels is specifying a range of lines within the data being edited to be processed by a particular command.
A symbolic label may be assigned to a data line by the end user. This label is a character string that must begin with a period (.) followed by one to five alphabetic characters (no numeric or special characters are allowed and labels starting with “Z” are reserved for system use.)

Labels may only be assigned to data lines. A symbolic label may be removed from a data line by blanking out the label characters (or overtyping with a new label) and pressing ENTER. A label can be unassigned by deleting line containing that label, or by using RESET LABEL command.
There are several special labels that are automatically assigned and maintained by the editor. They all begin with the letter “Z”. Labels beginning with the letter “Z” are reserved for editor use and may not be assigned by the end user.

ZCSR	The Data Line On Which The Cursor Is Currently Positioned
ZFIRST	The First Data Line (Relative Line Number 1). This May Be Abbreviated To .Zf

ZLAST The Last Data Line. This May Be Abbreviated As .Zl.

You can limit the range of lines within the data being edited, which will be processed by certain primary commands. Do this by entering a pair of “labels” indicating the first and last lines to be processed, and naming those labels in the “range” operand of the appropriate command. The data will then be processed by the command if it is contained within the designated range.

The range operand consists of two labels that must be separated by a blank or comma. A single label is invalid. The labels may be any combination of “system labels” (.ZFIRST, .ZCRS, .ZLAST) or user labels.

Library Options

The following operations can be performed on a data set:
Compressing, Printing Index Listing, Printing

The following operations can be performed on members:
Browsing, Renaming, Deleting, Printing

Compressing Data Sets
The compress function recovers wasted space occupied by deleted or updated members. Use the option “C” for compressing data sets.

Printing Index Listing
An index listing shows general data set information and member names for partitioned data sets. For ISPF source libraries, activity statistics are listed for each member. For load libraries, load module information is listed for each member. The index listing is recorded in the ISPF list data set. Use the option ‘X’ for printing index listing.

Primary Data Sets
The print data set function formats the contents of a source data set for printing, and records the output in the ISPF list data set. This function also produces an index listing, which appears at the beginning of the output. Use the option ‘L’ for printing the entire data set.

Printing a Member
The print member function is used to print the contents of a member of a partitioned data set. The listing is recorded in the ISPF list data set. (Use the member list option if you wish to print several members). Use the option ‘P’ for printing a member.

Renaming a Member
The rename member function is used to rename a member of a partitioned data set. (Use the member list option if you wish to rename several members). Use the option ‘R’ for renaming a member.

Deleting a Member
The delete member function is used to delete a member of a partitioned data set. (Use the member list option if you wish to delete several members). Use the option ‘D’ for deleting a member.

Browsing a Member
The browse member function is used to browse a member of a partitioned data set (Use the member list option if you wish to browse several members). To browse a single member, fill in the following fields in the library utility panel. Use the option ‘B’ for browsing a member.

Information Display
Used to display information about a data set on a direct access device. Use the option ‘S’ for data set information.

Data Set Utilities

Renaming Data Sets
Used to rename a data set on a direct access device. Use the option ‘R’ to rename data sets. You cannot rename VSAM data sets & password protected data sets with this utility.

Deleting Data Sets
To delete a data set on a direct access device, fill in the following fields in the data set utility panel. Use the option ‘D’ for deleting data sets. You cannot delete VSAM data sets and password protected data sets with this utility.

Data Set Information
Used to display information about a data set on a direct access device. Use the option ‘ ‘ for data set information.

Cataloging Data Sets
Used to catalog a data set on a direct access device. Use the option ‘C’ for cataloging a data set.

Uncataloging Data Sets
Used to uncatalog a data set on a direct access device. Use the option ‘U’ for uncataloging a data set.

Move / Copy Data Set Option
The Move/Copy utility is used to move or copy data from one data set to another, or from one member of a partitioned data set to another.

Data may be moved/copied to and from:

- A sequential data set
- A partitioned data set

In the Move/Copy Utility Panel, Enter the "From" and "To" library information in the appropriate fields. If the "From" data set is partitioned, enter a member name as follows:

- To move or copy a single member, enter the member name
- To move or copy all members, enter '*' (asterisk)
- To request a member selection list, leave member name blank or specify a pattern

When you press the ENTER key you will get a 'TO' data set screen.

Data Set List Option

The DSLIST panel options are:

The display data set list function displays a list of data sets in scrollable format. This list is created by entering blank as the selection code on the DATA SET LIST UTILITY panel and specifying a data set name level and a volume serial or by just specifying a volume serial.
When VOLUME is specified, the list is created through a search of the VTOC (Voume Table Of Contents contains a group of records that contain the data set names and attributes of each data set on the DASD). When DSNAME LEVEL is specified but VOLUME is not specified, the list is created through a catalog search. Other functions that can be performed with the data set list are: Browse, Edit, Delete, Rename, Catalog, Uncatalog, Print, Memberlist, Compress and Free.

COMMAND

The TSO command processor option allows TSO commands, CLISTs, and REXX execs to be executed under ISPF. The TSO command processor panel is displayed when option 6 is entered on the primary option menu.
You may enter a long command that wraps to the next line. The maximum number of characters that may be entered is 234.

List and Log Data Sets

The LIST and LOG command panels shown the default processing values for the respective data set. You can use the panels to print, delete, or keep the data set by selecting a process option and pressing ENTER. If you are printing the data set, you can also modify the SYSOUT class, local printer ID, and job statement fields.

The process option must be specified in order for the list data set or log data set to be processed. You can specify a SYSOUT class for the ISPF list data set or log data set. It is used if the process option “PD” (print and delete) is selected. The SYSOUT class is a 15 character field. It is used in printing the list or log data set. If you change the SYSOUT class on the LIST or LOG command panels, the new value is saved in your user profile and is used as the default, the next time you process the data set.
You can specify a local printer id for the ISPF list data set or the log data set. It is used if process option “PD” (print and delete) is specified.

Basic TSO Commands

ALLOCATE	alloc	To Allocate a dataset
CALL		Loading and executing programs
DELETE	del	To delete one or more data set(s)/member(s)
EDIT	e	To edit a data set/member
EXEC	ex	To execute a REXX/CLIST utility
FREE	f	To free the unused space allocated for data set
HELP	h	To invoke the TSO help
LISTALC	lista	To list data sets currently allocated to TSO/E
LISTBC	listb	To list mails and notices for your installation
LOGON		To logon to a TSO session
LOGOFF		To logoff from a TSO session
RENAME	ren	To rename a data set/member
SEND	se	To send messages to others
SUBMIT	sub	To submit a Batch Job for execution

To execute the TSO commands from ISPF, type TSO followed by the Command name, at the command prompt. Alternatively, you may choose option 6 from the primary menu and then type the Command without the TSO prefix, or you may go to the READY prompt and type the TSO command.

Common Abends

B37 Abend

This abend occurs because, a data set opened for output has used all the primary space, and no secondary space was requested.

To rectify: Compress the dataset, if the problem is still unresolved, reallocate the data set increasing its space quantity.

D37 Abend

This abend occurs if there is no free space in the directory.

To rectify: This abend can be rectified by deleting the statistics for the members in the PDS by typing STATS OFF primary edit command. This is a temporary solution for reclaiming space. The permanent solution is to type TSO%REALLOC and to increase the number of directory blocks.

E37 Abend

This abend occurs if there is no more free space in the allocated data set. You will not be able to save the changes done to the data set or member.

To rectify: Compress the data set. If the problem is still unresolved, reallocate the dataset and increase its space quantity.

TSO Commands

The ALLOCATE Command

The allocate command is used to allocate data sets and also to associate filenames (ddnames) to dataset names.

```
ALLOCATE DATASET (‘DSNAME’/’LIST OF DSNAMES’/*)
        OR DUMMY
        FILE (‘DDNAME’) ALTFILE (‘DDNAME’)
        NEW/OLD/MOD/SHR/SYSOUT (‘CLASS’)
        VOLUME (‘SERIAL’/SERIAL LIST)
        SPACE (‘QUANTITY’,’INCREMENT’) DIR (‘INTEGER’)
        BLOCK (‘VALUE’)/TRACKS/CYLINDERS
        UNIT (‘UNIT-TYPE’)
                RELEASE REUSE
        KEEP/DELETE/CATALOG/UNCATALOG
        BLKSIZE (‘BLOCK SIZE’)
        DSORG (‘DATA SET ORGANIZATION’)
        LRECL (‘LOGICAL RECORD LENGTH’)
        RECFM (‘RECORD FORMAT’)
```

Ex:

```
TSO ALLOCATE DATASET (‘TRGE01, TRG.NEW’) NEW VOLUME (USER01)
SPACE(1,1) DIR(3) TRACK BLKSIZE(8000) DSORG(PO) LRECL(80) RECFM(F,B)
TSO ALLOCATE FI(SYSIN) DS(*)  TSO ALLOCATE FI(SYSOUT) DS(*)
```

The CALL Command

The call command is used to execute a load module in the foreground.

CALL ‘dsname (member)’ “parm” CAPS/ASIS	
OPERANDS:	If DSNAME is not coded, PREFIX.LOAD is used as the dsname
	If MEMBER is not coded, TEMPNAME is used as member.
	CAPS option is used to translate PARM string to uppercase

Ex
CALL ‘TRG101.TRG.LOAD (COBPRG)’

The DELETE Command

The delete command is used to delete either VSAM objects or NONVSAM data sets from a VSAM or ICF catalog and to free the space occupied by the objects or data sets. In addition, space occupied by datasets can be overwritten with zeros. The delete command also deletes GDG Bases and VSAM User Catalogs.

The EDIT Command

The edit command is used to create or modify sequential data sets or members of partitioned data sets. This is the TSO line editor, and not the ISPF editor.
EDIT ‘DSNAME’ NEW/OLD COBOL CAPS/ASIS

Operands:

‘DSNAME’	Name of the data set to be created or edited.
NEW	Data set named did not exist before the command was issued.
OLD	Data set already existed when the edit command was issued.
COBOL	Data consists of COBOL source statements.
TEXT	Data is prose. Character conversion default is ASIS.
CAPS	Data lower case letters are to be converted to uppercase letters, the default for all data set types except text.
ASIS	Input lower case letters are not to be converted to upper case. This is the default for text data sets.

The EXEC Command
The Exec command initiates execution of a command procedure or a REXX routine.

EXEC ‘DSNAME’ “PARAMETER LIST” LIST/NOLIST PROMPT/NOPROMPT EXEC/CLIST

Operands:

'DSNAME'	The name of the data set which contains the command procedure to be executed.
"PARAMETER LIST"	The list of values that are to be substituted for the symbolic parameters Defined when the command procedure was created if the procedure is a CLIST.
PROMPT	Prompting to the terminal for input is to be allowed during the execution of the command procedure.
NO PROMPT	Prompting to the terminal for input is not to be allowed.
CLIST	The command procedure to be run in a CLIST
EXEC	The command procedure to be run in an EXEC

Ex

```
EXEC 'TRG101.TRG.REXX (REXXPRG)' EXEC
```

```
DELETE      ('ENTRYNAME/PASSWORD'…)
            FILE('DNAME')
      PURGE | NO PURGE
      ERASE | NO ERASE
      SCRATCH | NO SCRATCH
      FORCE | NO FORCE
      RECOVERY | NO RECOVERY
      CLUSTER | GENERATION DATAGROUP
      NONVSAM | PATH | ALTERNATE INDEX
```

Operands:

'ENTRYNAME/PASSWORD'	Specifies the name of the entry to be deleted.
FILE('DNAME')	Specifies the location of the object to be deleted.
FORCE	The object is to be deleted even though it is not empty.
NOFORCE	The object is not to be deleted if it is not empty.
PURGE	The object is to be deleted regardless of expiration date.
NOPURGE	The object is not to be deleted before expiration date.
ERASE	Dataset space is to be overwritten with binary zeros.
NOERASE	Dataset space is not to be overwritten with binary zeros.
SCRATCH	The nonvsam dataset being deleted is to be removed from the VTOC of the volume on which it resides.
NOSCRATCH	The nonvsam data set being deleted from the catalog is to remain in the VTOC of the volume on which it resides.
CLUSTER	The entry to be deleted is a cluster.

GENERATION DATAGROUP	The entry to be deleted is a GDG.
NONVSAM	The entry to be deleted is a nonvsam data set.
PATH	The entry to be deleted is a path.
ALTERNATE INDEX	The entry to be deleted in an alternate index.

Ex

DELETE ‘TRGE01.TRG.NEW’

Introduction to JCL

What is JCL?

Job Control Language or JCL is a set of control statements that provide the specifications necessary to process a job. Communication with the OS is by typing Commands or Job Control Language statements. The OS coordinates and manages resources; the command language provides guidance and direction.
Job stream is a series of commands prepared and submitted before the first program is loaded.

The Role of JCL

We do not use JCL to write computer programs. Instead, it consists of control statements that introduce a computer job to the operating system, provide accounting information, direct the operating system on what is to be done, request hardware devices, and execute the job. JCL tells the operating system everything it needs to know about a job's (I/O) requirements. We code the JECL (Job Entry Control Language) to specify on which network computer to run the job, when to run the job, where to send the resulting output. IBM provides two job entry systems for z/OS: JES2 for decentralized control, & JES3 for highly centralized control of several computers.
The initiators are programs that have a single purpose: to find a job to start running and then go backstage.
When a JCL is submitted to MVS a series of things happen. The JCL is checked for syntax errors and interpreted. Any procedure library JCL that is called for is found and incorporated into the JCL that was submitted. JCL statements referring to files are processed. Files are found or created and connections to the program established.
When it is time to close, the initiators are closed and the system is shut down.

Purpose of Using JCL

Job Control Language functions:

- Identify users – essential for security
- Identity programs
- Specify device requirements
- Run-time intervention

Concept of Job and Job Steps

A JOB is an execution of one more related program in sequence. Each program to be executed by a job is a JOB STEP. A job begins with the execution of the first job step and continues until the last program has finished executing, unless an error occurs.

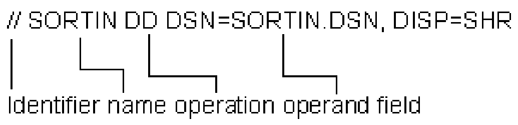
There are three basic JCL statements in every job – JOB, EXEC, and DD. The first statement in every job is a JOB statement to furnish necessary data to identify the job, such as programmer’s name and accounting information. A job consists of one or more job steps. For each job step within the job an EXEC statement has to be coded to indicate the name of the program to be executed. Within each job step, DD statements have to be coded for every file that is to be processed by the program.

JCL Statement Format

JCL statements are coded in 80-byte records in the card-image format. Thus editing is easy on a 3270 terminal, since most of these terminals have an 80-character per line format.
Only the first 72 of the 80 characters are available to code JCL. The last eight columns of each record are reserved for an optional sequence number.

The basic format is:

Identifier [name] [operation] [Operand] [comments]



```
// SORTIN DD DSN=SORTIN.DSN, DISP=SHR
```

The identifier in column 1 is indicated by two slashes (//). It identifies a record as a JCL statement. There are two exceptions to this:

1. Delimiter /*
2. Comment Statement //*

The Name field associates a name with a JCL statement. It may have one to eight alphanumeric or national characters, starting with a letter or national character. It must begin in column 3 if coded. The name field is always required on a JOB statement as it supplies a name for the job. It is optional on EXEC and DD statements, but is usually coded on DD statements. The name field is not used in the delimiter, comment or null statement.
Some of the valid & invalid ex fragment code

Valid	Invalid
//1TEST	1st character not A-Z, @, #, $
//CUSTOMERMAST	More than 8 characters

Operation field follows the name field. At least one blank after the name field must be provided. It specifies the statement’s function – JOB, EXEC, DD, PROC, PEND, OUTPUT. Delimiter, comment and null statements do not have an operation field since their unique identifier fields indicate their functions.

The operations field may be coded anywhere in the format provided it is separated from the name field by at least one blank. However, it is recommended that the coding for this field always starts in the column 12 to ensure easy readability.

Some of the valid & invalid ex fragment code

```
//MTRG       JOB (A123),'AIT', CLASS=A -VALID
//MTRG       JOB (A123),'AIT', CLASS=A -INVALID
```

Separate it from the operand field by at least one blank, and you can code it only there is an operation field.
The Operand field begins at least one position after the end of the operation field and can extend upto column 71. From coding neatness point of view it is recommended to start the coding for parameters from column 17. Parameters form the substantial part of coding of JCLs. There are only a few statements, but each statement has options for several parameters which can be coded in different ways. Learning to code parameters correctly is the fundamental part of learning JCLs.

When more than one parameter has to be coded, the separator is a comma, not a blank. A blank is taken to mean that the parameter field is complete. If a space is necessary as part of a parameter value, then the parameter value must be enclosed within apostrophes.
There are two types of parameters positional and keyword. A positional parameter must occur at a relative specific position within the parameters field. Keyword parameters are not position specific because they are identified by a keyword. Keyword parameters come after positional parameters have been coded. Some JCL parameters have positional sub-parameters, which are individual specifications within the parameter.

Operand Field Rules

Separate both positional & keyword parameters with commas. Blanks are not permitted.
Code positional parameters in the specified order, before any keyword parameters in the operand field.

Correct	(8,12),CLASS=A
Incorrect	CLASS=A,(8,12)

You need code nothing if all positional parameters are absent.

```
//MTRG                    JOB   CLASS=A
```

Rules for Continuation of JCL Statements

1. Break the parameter field after the comma that follows a parameter or sub-parameter.
2. Code slashes in columns 1 and 2 of the following line.
3. Code the next parameter or sub parameter beginning anywhere in columns 4 through 16.

General JCL Rules

Start all statements in column1 with the appropriate // or /*
Begin any entry you code in the name field (it is sometimes optional) in col 3 and follow it with at least one blank.

//AIT JOB	-	VALID ONE
// AIT JOB	-	INVALID ONE
//AIT 01 JOB	–	INVALID

Some parameters apply only to specific systems or hardware devices. The system ignores parameters if they are inappropriate for a feature or hardware device.No embedded blanks are allowed within the fields, and commas must separate each statement. All JCL statements must be in uppercase(CAPS)

JCL Statements & Parameters

There are primarily 3 types of JCL statements.

JOB :	Identifies a job and supplies accounting information
EXEC:	Identifies a job step by indicating the name of the program to be executed
DD :	Identifies a data set to be allocated for the job step

JES2/JES3

Batch operating environment

In batch mode you submit or send processing request (JCL) to the system and then go do something else, like read a manual or have coffee.
Your processing request is put into queue or line of other requests submitted by other people. The system will eventually get around to your request, process it, and return the results to you. Batch mode is generally cheaper than interactive mode. Interactive mode demands a higher level of service from MVS and requires fast response time

JOB Management

MVS facility that reads and stores jobs, selects jobs for execution based on relative importance of each JOB, allocates resources to jobs as they execute, processes print output produced by jobs

For this, MVS uses a JOB entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by MVS, and to control their output processing
There are two types of JES - JES2, JES3

JOB Statement

The JOB statement informs the operating system of the start of a job, gives the necessary accounting information, and supplies run parameters. Begin each job with a single JOB statement. The JOB statement has the following form:

```
//jobname    JOB (acct-no,acct-info),name,Keyword_parameters.
```

The jobname is a name you assign to the job.
The acct-no, acct-information is the account number to which job is charged and any additional accounting information. The name is whatever you select to identify the run.
A JOB is the execution of one or more related programs in sequence. Each program to be executed by a JOB is called a JOB STEP.

Ex:
Suppose you want to process a JOB that executes two programs. The first does sort on the input file and second prints a report. The JCL will have two JOB steps as follows:

```
//ABCED     JOB
//STEP1     EXEC PGM=ONE
//FILE1          DD ............
//FILE2          DD ..........
//STEP2          EXEC PGM=SECOND
//FILE2          DD .........
//OUTFIL    DD   SYSOUT=*
//*
```

Job Control Language

JCL is a set of control statements that provide the specifications necessary to process a JOB. The three basic JCL statements that are present in every JOB are: JOB, EXEC AND DD.

Ex:

```
//MMA2      JOB  USER=SUGUM, PASSWORD=XXXXXXX
//MYJOB     EXEC PGM=IEBGENER
//SYSPRINT       DD   SYSOUT=A
//SYSUT1    DD   DSN=XINDSM.COPLIB.COBOL, DISP=SHR
//SYSUT2    DD   SYSOUT=A
//SYSIN     DD   DUMMY
```

JCL Statements

- Columns 1-2 should be always //.
- Columns 3-10 is the name field. Must be alpha-numeric and should start with alphabet or national character. Optional field. Must be followed by at least one blank character.
- Column 12 onwards is the operator field indicating type of control statement (eg. JOB, EXEC). Must be preceded and followed by at least one blank.
- The parameter field follows the operator field and must be preceded and followed by at least one blank character. Contains parameter separated by commas.
- To continue a parameter on to the next line, insert a non blank character (typically *) in column 72.
- Comments in JCL begin with //*

Rules for coding JOB statement

- A valid name must be assigned to the job that is being submitted. This is called jobname, and it must follow the initial two slashes that precede all JCL statements.
- The word JOB must be coded in the operation field.
- Positional parameters can be coded in the operands field.
- Keyword parameters can also be coded in the operands field. Keyword parameters cannot precede positional parameters.
-

Comments can be coded as in ay JCL statements. Comments are recognized as such by coding as follows:
//* comment   this an ex for jcl comment

Kinds of Parameters

Two kinds of parameters - Positional & Keyword

Positional Parameters

Positional or keyword parameters are coded in the operands field. Positional parameters must precede the keyword parameters, if both exist. Positional parameters are called as such because of the position in which they appear in the operand field, they must be displayed in certain order.

Keyword Parameters

- Any sequence permitted
- Separated by commas
- Always follow positional parameters

Eg: //jobname <positional parm>,<keyword parm>

JOB Name

The jobname must start in column 3 immediately following the double slashes. It must be followed by at least one space. It must be the first statement for the particular job being submitted at the time.

Rules for Coding

Valid names can be 1 to 8 alphanumeric characters in length. The #,$,@ and symbols (also called national symbols) can also be included in the name. The first character of the name must be alphabetic or national; it cannot be numeric.

Ex:

//ARICHs JOB .... valid
//@Xin$123 JOB .... valid
//123AIT JOB………. invalid …Because it starts with numeric
//ARICHsAIT JOB ……… invalid ….Because it length is more than 8 characters

The Positional Parameters

Accounting Information

The accounting information parameter identifies the account number which will be billed for the CPU time utilized on the mainframe. This parameter is optional.

Rules for Coding

Parentheses or apostrophes must be used if additional accounting information is coded. If both parameters are coded then they must be separated from each other by a comma. If only additional accounting information is coded, then the absence of the account number must be indicated via a comma. This is because these are positional parameters if one is omitted, then its absence must be indicate in this way.

Ex: //xindsmc JOB (aa,bb,ccccc)

Programmer Name

Used for user identification for routing printouts. Specified in single quotes if name contains spaces or special characters other than periods. Single quotes in the name must be specified as double quotes.

Ex:

```
//Xindsmr    JOB (aa,bb,cc),‘prg xx’
//Pcicsd          JOB (dbss cics),‘prod. Cics’
```

The Keyword Parameters

A variety of keyword parameters can be specified on the JOB statement; these are:

CLASS COND MSGCLASS MSGLEVEL NOTIFY PASSWORD PERFORM PRTY REGION RESTART TIME TYPRUN USER

EX:

```
//TSOH23$  JOB (S-1465-3000-22-A),'JOHN SMITH',
//         CLASS=A, MSGCLASS=R,
//         NOTIFY=DTZ4522, USER=DTZ4522,
//         REGION=4096K, TYPRUN=HOLD
```

CLASS

CLASS=class specifies the job class. Job classes to use: A to Z, 0 to 9. To establish job classes that achieve a balance between I/O bound jobs and CPU-bound jobs, between big jobs and small jobs. Job Classes also determine the overall priority of a job, along with the PRTY parameter.

- Indicates the type of JOB submitted to the system
- Specifies input class for JOB scheduler
- Single alphanumeric character (a-z,0-9)
- Different classes will represent different types of jobs and will depend on installation.

Ex CLASS=A - test jobs
CLASS=B - production jobs

//DBSS1 JOB (‘ACCT’), CLASS=A

MSGCLASS

- Assigns the JOB log to a particular sysout class
- Specifies output class for the JOB log, list, system messages, JES2, MVS operator messages.
- Single alphanumeric character- A-Z;0-9
- Depends on JES set-up.

Ex: MSGCLASS=T - Output to terminal

//DBSS3 JOB CLASS=A, MSGCLASS=T

MSGLEVEL

MSGLEVEL=(Jcl, allocations) specifies the printing of jcl statements and allocation messages. Allocation messages, if requested, appear at the beginning of each job step to show the allocation of data sets to device and at the end of the step to show the allocation of data sets to devices and at the end of the step to show the data set disposition.

- Determines the JCL print & allocation messages.)
- JCL

0	Only JOB
1	All input & procedure statements
2	Only input statements

- Allocations

0	No allocation messages
1	All allocation messages

EX:

```
//TEST2$@ JOB ('123 456',TSO,2,1),TEST,MSGLEVEL=(0,0)
//CICSPROD        JOB (ACC-ED-231-0),'PROD',MSGLEVEL=(1,1)
```

COND

Each job step may pass a return code to the system when it reaches completion. The Cond parameter lets the execution of the steps depend on the return code from previous steps. If we omit cond parameter, the system makes no tests and executes the steps normally. The system ignores a cond on the first exec step of a job because there us no previous step to test. Two different codes are retuened in z/OS. An application program can return a code at the end of a step(termed a return code or completion code). Subsequent job steps can test the return code from a previous step by use of the cond parameter. The operating system also return a system completion code at the end of the job. The system code is a 3 digit code pre-fixed with an S. The usual value is 0000, indicating normal completion.
Specifies whether a JOB should continue to execute or terminate in relation to the outcome of any JOB step

Format is COND=(code, operator)

Code	0 thru 4095
Operator	GT, GE, EQ, LT, LE, NE

If condition is true, the JOB is terminated

Eg. Cond=(0, NE) should be read as:

If 0 is not equal to the condition code returned by any JOB step, terminate the JOB. In other words, the JOB will execute only if ALL JOB STEPS RETURN A CONDITION CODE OF 0

Ex

```
//step1    exec  pgm=one
//step2    exec  pgm=two
//step3    exec=pgm=three, cond=(4,gt,step1)
```

in the above ex the system bypasses step3 if 4 is greater than the return code from step1.

```
//step5 exec cob2db, cond=(4o95,lt)
```

step5 executes regardless of the return codes. Return codes cannot exceed 4,095 so the condition can never be met.

TIME

TIME coded on the Job statement sets the limit for the entire job.It does not override the limits set on exec statements – it is applied independent of them. Coding time=1440 or time= no limit on the job statement nullifies any time parameters on the exec statements. Specifies the maximum CPU time for a JOB
System will automatically cancel the JOB if its execution time exceeds the specified time. Format is time=(MINS,SECS)

E.G.	TIME=(1,30)	maximum CPU time for this JOB is 1minute and 30 seconds
	TIME=2	maximum CPU time for this JOB is 2 mins
	TIME=(,20)	maximum CPU time for this JOB is 20 secs

Comment Statement Format

In addition to the comments field of any JCL statement, comments can be added to a job in a comment statement. You may use a comment statement to help clarify a confusing JCL statement or to identify the purpose or operation of a job. You should also place comments at the beginning of the job to indicate who wrote the job, when it was written, and to explain what the job does. The entire comment is ignored when the job is processed and does not affect the output. A comment is identified by slashes in positions 1 and 2 and an asterisk in position 3.

Ex (A caret (^) indicates a space.):

//*^^^THIS IS A COMMENT

Other Keyword Parameters

TYPRUN

Typrun tells the system to not execute the job, either to check the jcl or to hold it for later execution. The general form is as follows;
Typrun=[{scan/hold/}]

Has two options

SCAN causes JCL to be scanned for syntax errors only
HOLD causes JOB to be detained in input queue until explicitly released by the operator

Restart

Enables the execution of a JOB from a particular step instead from the top of the JOB step. Format is RESTART=STEPNAME

PRTY

Prty specifies the job’s priority in the input queue.

- Determines priority for scheduling jobs within each class
- Ranges from 0-15

REGION

- Specifies amount of space JOB requires
- Can be specified in kb or mb
- Format is REGION=NK/NM

Ex:
//XINDPR JOB ‘ABCD’, CLASS=A, PRTY=3, REGION=4092K

USER/PASSWORD

The user parameter specifies your RACF user id. The user id is used during the job's processing by RACF and JES to identify the user who submitted the JOB for security checking purposes.The user, the password are required on JOB statements in these circumstances. Jobs submitted to execute at another network node that uses RACF security

Job Entry Subsystems

JOB SCHEDULER

- JES is the MVS component that keeps track of jobs that enter the system
- Presents them for MVS processing
- Sends the job's spooled output to the correct DESTINATION

TYPES

HASP	Houston automatic spooling program now called as JES2
ASP	Asymmetric multiprocessing system suitable for shops with more than one processor

Each MVS system can use JES2 or JES3 not both

JES2/JES3/JCL Implementation

JES control statements follow rules that are similar to the rules for coding JCL statements

The three main differences

1. JES2/JES3 statements use a different identifier in the first columns of each Statement
2. JES2/JES3 don't have a name field
3. JES2/JES3 follows different continuation rules

Format of JES2/JES3 control statements
Identifier operation [parameters]

PRIORITY

The /*PRIORITY statement is used to establish a JES2 selection priority for your JOB Within a JOB class, a JOB with a higher priority is selected for execution and printing sooner. The /*PRIORITY statement must immediately precede the JOB statement in a JES2 network. The /*PRIORITY statement must appear immediately after the /*XMIT statement and before the JCL JOB statement

Format:
/*PRIORITY n n- 0 to 15

Ex
/*PRIORITY 13
//SYS2451 JOB (S-FGG-23),'SMITH', CLASS=A, MSGCLASS=T

```
//JS10   EXEC PGM=IEBCOPY
```

ROUTE

The /*ROUTE statement is used to route a job's output to a specific network node, or to request execution at a specific network node
Data sets that are routed by a DEST parameter are routed separately from the rest of a JOB routed with a /*ROUTE statement
The /*ROUTE statement should be placed after the JOB statement and before or after the job's EXEC statements

```
/*ROUTE {PRINT | PUNCH} DESTINATION
        where DESTINATION is NODENAME.USERID
```

Ex:
```
/*ROUTE PRINT N4.R212
```

SET-UP

The /*set-up statement is used to identify tape or DASD volumes that the operator should mount before the JOB is executed
When a JOB containing /*set-up statements is read in, JES2 sends an image of each /*set-up statement in it to the MVS console along with a message stating "JOB held for the following volumes:". The JOB is then held until the operator mounts the needed volume(s) and manually releases the JOB. The /*set-up statements you use should appear after the JOB statement and before the first EXEC statement in the JOB.

Ex: /*Set-up volser{,volser,....}

SIGNOFF

The /*SIGNOFF statement is used to indicate the end of a remote JOB stream processing session. JES2 will then disconnect the remote work station from the system after any current data transmission to or from the remote has completed.

/*SIGNOFF
No parameters are coded on a /*signoff statement

Ex

```
/*SIGNOFF
```

SIGNON

The /*signon statement is used to initiate a remote JOB stream processing session

The /*signon statement can also be used to override the remote number normally assigned to the REMOTE. You should insert the /*signon statement at the beginning of an input JOB stream .

```
/*SIGNON {REMOTENNN} {PASSWORD1} {PASSWORD2}
```

Ex:

```
/*SIGNON REMOTE435FOXTROT
```

XEQ

The /*xeq statement is used to indicate at which node in your JES network this JOB is to execute. It should be inserted in your JOB after the JOB statement and either before or after the EXEC statements, but before all DD * or DD data statements in the JOB

```
/*XEQ NNNNN | NODENAME
NNNNN - NODE NUMBER
NODENAME - locally defined network-wide name
```

Ex:

```
/*XEQ SANDIEGO
```

XMIT

The /*xmit statement is used to transmit records from a JES2 network node to either another JES2 network node or an eligible non-JES2 network node.

```
/*XMIT  Nnnnn | Nodename.userid DLM=xx
```

Ex:

```
//C22901 JOB (125567),'XMIT TO SANDIEGO', CLASS=A
/*XMIT SANDIEGO DLM='??'
//C22901D JOB (344551),'COPY FILES', CLASS=Z
//JS10 EXEC PGM=IEBCOPY
```

EXEC Statement

A job step is a unit of work that is submitted to the operating system in the form of a collection of JCL statements. The EXEC statement is the first statement of each job step. A unit of work can be the execution of a program. Or, it can be the execution of a procedure that is pre-written and available for general use by users of the system.
JCL parameters on the EXEC statement

JCL parameter	**Function**
JOBLIB	Specification of location of program to be executed, location being universal for all job steps within a job
STEPLIB	Specification of location of program to be executed, location being

	Specific to a job step
PGM	Specification of program name
Stepname	Specification of job step name
ACCT	Specification of account number for particular job step
PARM	Sending values to a program when it is executed
ADDRSPC	Specification of storage requirements for a job step
DPTRY=(val1,val2)	Assignment of priority to a job step.
PERFORM	Specification of rate of access to system resources

//Stepname EXEC <positional>, <keyword>
Stepname is optional, but recommended. If used, must be unique in the JOB including any procedures called by the JOB

The positional parameter can be one of the following:
PGM=program
PROC=procedure
procedure name

EX

```
//PCICSD          JOB 'ABCD',, CLASS=A
//STEP1           EXECPGM=PROG1
//DD1             DD   .....
//STEP2           EXEC PROC=PROC1
//DD2             DD ............
//STEP3           EXEC PROC2
```

STEPNAME

The stepname is an optional field. Since system messages reference stepnames, it is strongly recommended that you give unique and meaningful names to each step.

A step name identifies the statement is on, and allows it to be referenced by another statement. Every step should be named. Name syntax similar to JOB name syntax

Keyword Parameters – PARM

The PARM parameter is used to supply information to a program as it executes. It is a very useful parameter.
PARM=Value Where value is a string from 1 to 100 characters long.

Rules for coding
- Passes parameters to the program that is being executed Maximum length of 100 characters
- Put quotes or parentheses around the parm value if special characters are used

- The information is placed in a COBOL program's linkage section automatically if procedure division using linkage-data- area is specified
- To continue on a second line, enclose the value in parentheses & use a continuation `,' (comma)

Ex

//Stepname EXEC pgm=pgm, parm='91/12/01' … valid.. since special characters are enclosed within quotes.
//STEP23 EXEC PGM=PRG2,PARM=(ARICH,AIT,CHENNAI)… valid
In this EX the string is enclosed within parentheses. The parentheses are not passed to the program as part of the string.

COND
The COND parameter coded on EXEC statement applies only to the job step that it is coded in. this job step is executed or bypassed, depending on the condition codes issued by one or more prior job steps.
Used to specify that a JOB step will be executed, based on return codes issued by one or more of the preceding JOB steps. The COND parameter lets the execution of one step depend on the outcome of a previous step. For ex in compile, link-edit JCL, if compile step condition code or return is not satisfactory, link edit step will not be performed as it depends on the successful execution of compile step.

COND=(value,comparison)
comparison - GT, LT, GE, LE, EQ, NE

COND=(comparison-code, condition, STEPNAME)
COND=EVEN
COND=ONLY
Comparison code is a number between 0 and 4095. Condition specifies the type of comparison to be made between the comparison code that has been coded and the return code of the prior step.

EVEN - Execute this step even if a preceding JOB step abends
ONLY - Execute this step only if a preceding JOB step abends

A maximum of 8 conditions may be checked in any one step (even & only count as one).Of the eight, omitting stepname in the parameter, will apply the test to all preceding steps. If one of the steps being checked was not executed, its condition check is ignored.

Ex. Cond=(0,eq,step) should be read as:

If zero equals the condition code passed from step1, do not execute this step
If condition is true, step is bypassed.

Ex:

//STEP2 EXEC PGM=READ, COND=(4,LT)
The cond=(4,lt) means that if 4 is less than the condition code of any previous step, then don't execute this step. It tells the system to execute this step, if previous condition codes were less than or equal to 4. The step executes for codes of 0-4. If the expression is true the step is not executed.

//STEP2 EXEC PGM=READ,COND.STEP1=(4,LT)
Maximum of eight conditions can be coded for the cond parameter. If any expression is true, the step is not executed

//STEP1 EXEC PGM=READ, COND=((4,GT),(6,LT))
The step is executed for 4, 5 and 6, and not for 0-3, 7+

SPECIFIC PRIOR STEP:

//   COND=(6,LT,STEPA)
//STEP1 EXEC PGM=ONE
//STEPA EXEC PGM=A, COND=(0,EQ,STEP1)
//STEPB EXEC PGM=B,COND=(0,LT,STEP1)
//STEPC EXEC PGM=C,COND=EVEN   - EVEN if any previous step abends execute this step
//STEPD EXEC PGM=D,COND=ONLY   - ONLY if any previous step abends execute this step
COND=(0,LE) - THE STEP IS NOT EXECUTED
COND=(0,GT) - THE STEP IS ALWAYS EXECUTED
COND=((0,GT), EVEN) - this step is always executed, even if previous step abends

TIME

This is an optional keyword parameter and is used to specify the amount of CPU time that a job step is permitted to utilize before it is terminated. If the TIME parameter is not specified then the installation defined TIME parameter is used.
Specifies maximum CPU time allowed for this step

- If actual CPU time exceeds time parameter, a system abend S322 will occur
- If you specify time both in the JOB and EXEC statement, then the time parameter on the EXEC applies only for that step

TIME = (minutes, seconds)
Where minutes is a number from 1 to 1439 (that's 24 hours) and the seconds is a number from 1 to 59.

Ex:
//Abcde   JOB 'hjdfs', ,class=a, time=(1,50) valid… the job will execute for 1 min & 50 seconds only

//Step1 EXEC pgm=prog1, time=(0,30)… valid.. the step will executed for only 30 seconds.

REGION

The region parameter coded on the EXEC statement requests storage only for that individual job step. If not enough storage is specified , then the job will abend abnormally.
The region parameter specifies the size of the execution region for a JOB step
The value specified is the amount in k (1024 byte units) or m (1024*1024 byte units) which will be the upper limit on the amount of virtual storage that can be obtained via getmain requests.

Region=nnnnk/nnnnm - bytes

Ex:
//Step1 EXEC pgm=iebcopy,region=100k …. Valid ..the step requires (100*1024) bytes of storage
//Step1 EXEC pgm=idcams,region=1M ..valid the step requires 1 million bytes of storage.

DD Statement – Data Definition Statement

The purpose of data processing is to take raw data, rearrange or modify it and produce results in the form of output. The DD statement (Data Definition) is used to identify the source of input and the placement of out information.

- Required for each input/output file accessed by the program
- Specifies the name of the physical file, and their properties
- Links between the symbolic file name (called DD name) of a file, as defined in the program, with its physical file

//DDname DD <positional>,<keyword>
Up to 3273 DD statements can be allocated to a single JOB step

DDNAME

Alphanumeric or national characters, and is up to 8 characters in length Mandatory in the DD statement.
If omitted, then the dataset is concatenated to previously defined DD statement Logical file name as defined in the program.

The DDname starts in column 3 on the DD statement
INSTREAM DATA (*) Indicates that instream data follows this statement
The end of data is indicated by a delimiter /* or //.
The instream cannot contain // in columns 1-2.

```
//INFILE DD *        DATA FOLLOWS
12345 SOME
374832 DATA
/*
```

DATA

Data is same as '*' operand except the instream
Data may contain // in columns 1-2.
Normally followed by a keyword parameter dlm=<chc> indicating the new delimiter to be used for end of DATA.

Ex:

```
//INFILE DD DATA DLM=$$
#62HJKS
//! 32647
$$
```

DUMMY

Dummy specifies that no devices are to be allocated to the file referenced by the DDname and that all i/o requests be bypassed. Normally followed by a keyword parameter DCB for output files.

Ex: //INFILE DD DUMMY
The other method is to code with DSN as nullfile
//INFILE DD DSN=NULLFILE

DATA SET NAME (DSN)

DSN parameter is a keyword parameter on the DD statement.
Mandatory. Specifies the full qualified name of the associated physical file. (input or output). Maximum of 44 characters with a period after every 8 characters or less
A temporary file starts with && followed by 1 to 6 characters.
For spooler file specify SYSOUT=class, where class identifies the class of spooler output. Sysout=* puts same as that of the JOB, as mentioned in MSGCLASS of JOB statement.

Types of DSN

- Non-Qualified
- Qualified

Non-qualified names are comprised of 1 to 8 alphanumeric or national characters. Non-qualified names are seldom used. It is impossible to catalog these data sets. Qualified names consist of two or more non-qualified names, each separated from the next by period. The first name is called the highest qualifier of that the name. when a qualified data set is referenced, the system determines if the highest qualifier is the alias given to a user catalog and attempts to find the data set in that catalog. If there is no corresponding data set of that name. Then the system attempts to create a new entry for it in the user catalog.

```
//AIT       JOB   A50,'ARICH'
//STEP1  EXEC PGM=PRGM2
//DD        DSN=ARICH.AIT.PGMLIB(TEST)
```

The name inside the parentheses (TEST) is the name of the data set being accessed. The advantage of qualifying data set names is that it gives you the ability to group data of similar types together.

DISP

DISP is an keyword parameter. It is used to instruct the system as to the current status of a dataset, and the steps to taken with the data set upon successful or unsuccessful execution of the job. DISP is required, unless the dataset is created and deleted within the same step. DISP=({status}{,normal-termination-disp}{,abnormal-disp})
Parameters on the DISP statement

Status	**Normal Disposition**	**Abnormal Disposition**
NEW	DELETE	DELETE
OLD	KEEP	KEEP
MOD	CATLG	UNCATLG
SHR	UNCATLG	CATLG
	PASS	

Rules for Coding DISP Parameter

One or more of the sub parameters may be skipped, however, at least one sub parameter must exist. The parentheses can be omitted if only the status field is coded, like this : DISP=NEW.
If normal-disposition and abnormal-disposition fields are coded, and status is omitted , then a comma must be coded in its position, like this. DISP=(,CATLG,DELETE)
If the first & third sub-parameters are coded, then a comma must be coded in the location of the second parameter, like this: DISP=(OLD,,DELETE)

DISP Defaults

If the DISP parameter is not coded, the following happens:

- The status field defaults to NEW.
- The normal-disposition & abnormal-disposition defaults to DELETE.
- DISP=(NEW,DELETE,DELETE) OR DISP=NEW.

If dataset is OLD,
DISP=OLD OR DISP=(OLD,KEEP,KEEP)

The NEW subparameter

Coding DISP=NEW indicates that a new dataset is to be created. New is also the default, If noting is coded in the status field.

```
//AIT       JOB     A50,ARICH'
//STEP1  EXEC PGM=PRG1
//DD1      DSN=MTRG.COBOL.PGMLIB,DISP=NEW
```

The OLD subparameter

Coding the status field as OLD results in the operating system searching for an existing data set of the name specified. The data set becomes available exclusively to the step in which it is referenced. If this file is being written to then its old data will be lost, and replaced by the new data. The data set may or may not be cataloged. If it is not cataloged, then the VOLUME parameter must be coded.

The MOD Subparameter.

The MOD subparameter is used to modify sequential data sets. These types of data sets can exist on DASD. The MOD subparameter does different things under different circumstances. If the data set already exists use of the MOD subparameter in the status field results in the device position itself just after the last record, so that records can be added to it easily. If the sequential data set does not exist, then the system replaces MOD with NEW and creates it.
If the VOL parameter is coded on the DD statement in conjunction with the MOD subparameter, the system searches for the data set on the volume specified.

The SHR subparameter

Setting DISP to SHR is identical to setting it to OLD except when OLD gives exclusive control of the data set to the user, whereas SHR allows multiple jobs to read the same data set.

The Delete subparameter

The Delete subparameter indicates that the data set being referenced is to be deleted after successful termination of the job, thereby releasing applicable resources for other users/jobs.If the DELETE disposition is specified for a tape data set which has a retention period or expiration date subsequent to the current date then it is deleted only after the retention period on the tape. If this disposition is specified for a disk data set, the system removes its entry from the system catalogs. The data is not actually removed until it is written over.

The KEEP subparameter

The KEEP subparameter indicates that the dataset is to be retained or kept upon successful execution of the job. This parameter should be used with permanent data sets. If it is used with a temporary data set the system automatically changes it to PASS.

The PASS subparameter
Setting DISP to PASS specifies that the data set is to be passed to a subsequent job step within the same job. Use of this subparameter saves time, because the location of the data set and information with reference to the volume that it exists on, remains in memory for the duration of the execution of the job.

The CATLG subparameter
The CATLG subparameter is used to specify that the data set is to be retained and recorded in the system catalogs after successful job termination. The data set name, unit and volume are recorded.

The UNCATLG subparameter
The UNCATLG subparameter is used to remove the entry of the data set from the system catalogs. The data itself is not deleted. If the data set is not found, then the job terminates abnormally.

UNIT
The UNIT parameter is coded on the DD statement to specify an input or output device that is to be accessed.

- Required if disp has before as new.
- Specifies the type and number of devices to be assigned to a dataset.
- Names disk unit or tape unit for the file.
- Name can be generic like SYSDA & SYSSQ which respectively means any disk or tape unit. Or it can be a specific IBM unit 3480/3420.

JCL coded on the UNIT parameter

FUNCTION	**Corresponding JCL parameters & subparameter**
Specification of device via address	UNIT =device-address
Specification of device by model type	UNIT=device-type
Specification of device as part of a group	UNIT=group-name
Specification of device used by a previous job step	UNIT= AFF
Deferring mounting of a tape device, until it is ready to be used	UNIT = (device, ,, DEFER)

UNIT=({device-number | device-type | group-name} {unit-count | P},DEFER)
OR UNIT=AFF=Ddname

EX

```
//AIT      JOB   A50,’ARICH’
//STEP1 EXEC PGM=PRGM1
//DD1     DD DSNAME=DATA21,UNIT=3380
//DD2     DD DSNAME=DATA1,UNIT=AFF=DD1
```

VOLUME

The VOL parameter is also coded on the DD statement to identify specific disk tape volumes. JCL coded on the VOLUME parameter

FUNCTION	**Corresponding JCL Prameters & sub parameter**
Specification of serial number	VOL =SER
Referencing VOL Specification from prior step	VOL =REF
Allowing access to volume by single user	VOL =PRIVATE
Inhibiting dismounting of volume until end of job	VOL = RETAIN
Specification of sequence on which volumes are to be mounted	VOL =SEQ

May be specified if disposition has before parameter as new
Format is VOL=(V1,V2,V3,V4,V5)

REF=dsname or
REF=*.DDname
REF=*.stepname.DDname
REF=*.stepname.procstepname.DDname

V1 - PRIVATE
V2 - RETAIN
V3 - VOLUME SEQ. NO
V4 - VOLUME COUNT
V5 - serial number. Indicates the serial numbers of the tape or disk volumes the dataset is on. Max. 6 characters.

Ex:

```
VOL=(PRIVATE,RETAIN,,2,SER=TAPE01)
//AIT       JOB   A50,ARICH’
//STEP1 EXEC PGM=PRGM1
//DD1     DD DSNAME=DATA21,VOL=(PRIVATE,SER=380)
```

SPACE

The SPACE and DCB parameter are also coded on the DD statement. The SPACE parameter is used to allocate storage for new data sets on direct access storage devices. The DCB parameter is used to customize the way data is stored on devices.

SPACE=(S1,(S2,S3,S4),S5,S6,S7)

S1 - (req) type : tracks, cylinder, block size
S2 - (req) primary quantity.
S3 - (opt) secondary quantity.
S4 - (opt) directory block
S5 - (opt) rlse.
S6 - (opt) contig.
S7 - (opt) round.

FUNCTION	**Corresponding JCL parameters and sub parameter**
Requesting space in tracks	TRK
Requesting space in cylinders	CYL
Requesting space in blocks	A number of blocks required is coded
Specification of specific storage space at the time that a data set is created	PRIMARY
Specification of additional storage space if amount originally specified us insufficient	SECONDARY
Specification of additional storage for recording of name and location of PDS	DIRECTORY
Request for release of space previously allocated	RLSE
Request for contiguous space	CONTIG
Request for largest area of contiguous space	MXIG
Request for entire cylinder for storage of data set	ROUND

Ex:

```
//AIT   JOB  A50,’ARICH’
//S1     EXEC PGM=PRGM1
//DD1  DD  DSN=DATA1,UNIT=3380,SPACE=(TRK,4)
```

DCB

The Data Control Block is used to supply information to the system that allows it to manage the data sets that are created as jobs are submitted. When the new data set is created the following has to be specified to the operating system.Record within a data set can have different types of formats. The number of bytes of a record within the data set must be specified. Multiple records can be logically grouped together inside what is called block. The block size helps establish the efficiency with which records within the data sets can be accessed.Dataset control block. Required when disp is new
Sub parameters are :

LRECL	Record length of file. For variable length file LRECL should be 4 bytes more
BLKSIZE	Multiple of LRECL

RECFM	**Format of file**
F	Fixed Length, Unblocked
FB	Fixed Length, Blocked
V	Variable Length, Unblocked
VB	Variable Length, Blocked
FBA	Fixed Block with carriage control as first character
VBA	Variable Block with carriage control as first character
U	Unknown

Program should refer to control blocks for length of each record

DSORG	data set organization
PS	physical sequential
PO	partitioned.
IS	indexed sequential
EROPT	specifies the option to be executed if an error occurs in reading or writing a record.
ABE	abnormal end of JOB
SKP	skip the block causing the error.
ACC	accept the block causing the ERROR.

Ex:
DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)

Special DD statements

JOBLIB
Used to identify a program library to search first when attempting to locate programs executed during the job's life. Must be placed after the JOB statement and before the first EXEC statement in the JOB. More than one program library can be concatenated after the first one on a joblib. If a steplib DD is specified in a JOB that also has a joblib, the steplib takes precedence when searching for a program. If you use a jobcat DD statement, it goes after the joblib. You can nullify a joblib for a job step by adding a steplib to the step that points to SYS1.LINKLIB

```
//JOBLIB DD DISP=SHR, DSN=PROGRAM-LIBRARY-
```

JCLLIB
JCLLIB is used only on MVS/ESA version 4 JES2 systems or higher
Used to identify a private library or a system library from which include groups and JCL procedures are to be retrieved. The order in which the library names appear on the JCLLIB statement is the order in which they are searched for any JCL procedures (procs) and include groups referenced by this JOB. Only one JCLLIB statement is

permitted in a JOB, and it must appear after the JOB statement and before the first EXEC statement in the JOB.

```
//{NAME}  JCLLIB ORDER=(DSN1{,DSN2}{,DSN3}...)
```

{NAME} - an optional name that must start in column 3 on the jcllib statement
Dsn1,dsn2, dsn3, etc. - The fully datasets containing the JCL procedures or include groups referenced in this JOB.

Ex:

```
//PRODJOB1 JOB (S-1233),CLASS=A,MSGLEVEL=(1,1),
//              MSGCLASS=T
//              JCLLIB ORDER=(SYS1.PROCLIB,
//              SYS3.USER.PROCLIB,OPS420.JCLLIB)
//JS10          EXEC ASMHCL,COND.L=(0,NE)
//JS20          EXEC PGM=IEBCOPY
//INC$          INCLUDE MEMBER=PRODINC3
```

STEPLIB

The steplib DD statement is used to identify a program library to search first when attempting to locate programs executed during the JOB step. The steplib can be placed anywhere in the step's JCL . More than one program library can be concatenated after the first one on a steplib. If a steplib DD is specified in a JOB that also has a joblib, the steplib takes precedence when searching for a program

```
//STEPLIB DD DISP=SHR, DSN=PROGRAM-LIBRARY-NAME
```

'PROGRAM-LIBRARY-NAME' is a data set name or a referback of the form *.Stepname.Steplib

JOBCAT

The jobcat DD statement is used to identify a VSAM catalog to search first when attempting to locate cataloged data sets during the job's execution
The jobcat must be placed after the JOB statement and before the first EXEC statement in the JOB. More than one catalog can be concatenated after the first one on a JOBCAT. If a stepcat DD is specified in a JOB that also has jobcat, the stepcat takes precedence. If you use a joblib DD statement, it goes in front of the JOBCAT.

```
//JOBCAT DD DISP=SHR, DSN=CATALOG-NAME
```

STEPCAT

The stepcat DD statement is used to identify a VSAM catalog to search first when attempting to locate cataloged data sets while the step is executing
The stepcat can be placed anywhere after the EXEC statement in the JOB step's JCL
If a stepcat DD is specified in a JOB that also has a jobcat, the stepcat takes precedence.If you also use a steplib DD statement in the JOB step, in goes in front of the stepcat

```
//STEPCAT DD DISP=SHR, DSN=CATALOG-NAME
```

OUTPUT
The output parameter is used with the sysout parameter to connect a sysout data with an output JCL statement.
OUTPUT=REFERENCE Or OUTPUT=(REFERENCE{,REFERENCE})

A 'REFERENCE' can be one of the following:
*.NAME
*.STEPNAME.NAME
*.STEPNAME.PROCSTEPNAME.NAME

INTRDR
To make a sysout data set from a JOB step be a new JOB, direct the data set to the internal reader. Input to the internal reader must be the JCL statements to run the later JOB

CODE:
//DDNAME DD SYSOUT=(CLASS,INTRDR)
INTRDR is an IBM-reserved name identifying the internal reader
The system places the output records for the internal reader into a buffer in your address space. When this buffer is full, JES places the contents on the spool
Later, JES retrieves the new JOB from the spool.

EX
```
//JOBA          JOB   D58JTH, HIGGIE
//GENER         EXEC  PGM=IEBGENER
//SYSIN         DD    DUMMY
//SYSPRINT            DD    SYSOUT=A, DEST=NODE1
//SYSUT2        DD    SYSOUT=(M,INTRDR)
//SYSUT1        DD    DATA
//JOBB          JOB   D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTA       EXEC  PGM=SUMMARY
//OUTDD1        DD    SYSOUT=*
//INPUT         DD    DSN=REPRTSUM,DISP=OLD
//JOBC          JOB   D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTB       EXEC  PGM=SUMMARY
//OUTDD2        DD    SYSOUT=A,DEST=NODE2
//INPUT         DD    DSN=REPRTDAT,DISP=OLD
/*EOF
```

Temporary datasets
A temporary data set is a data set that is created and deleted in the same JOB, and is identified by coding one of the following:

Dsname=&&dsname for a temporary data set
Dsname=&&dsname(member) for a pds

The system generates a qualified name for the temporary data set
The name begins with sys and includes the julian date, the time, the JOB name the temporary name assigned in the dsname parameter, if specified, or an identifying name and number, if a dsname is not specified

EX

```
//TEMPDS1 DD DSNAME=&&MYDS,DISP=NEW,UNIT=3350,
//                        SPACE=(CYL,20)
//TEMPDS2 DD DSNAME=&&DSA,
//                        DISP=(NEW,PASS),UNIT=3380,
//                        SPACE=(TRK,15)
//TEMPSMSDD DSNAME=&&ABC, DATACLAS=DCLAS2,
//                        STORCLAS=TEMP1,DISP=NEW
```

SYSOUT

The sysout parameter identifies a data set as a "system output" data set, consisting of printed or punched output that will managed by JES2 or JES3

SYSOUT=CLASS OR SYSOUT=(CLASS,{INTRDR})
CLASS - TO WHICH THE OUTPUT BELONGS
If SYSOUT=* the MSGCLASS specified in the JCL is taken.

SYSIN / SYSPRINT

Can be explicitly declared and used for other purposes

SYSIN

used to provide instream data or can refer a data set used mainly by IBM utility program.

SYSPRINT

Defines an output file contain messages from IBM utility programs

```
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ABC
/*
```

The names sysin and sysprint are not reserved by the compiler as file names

DUMPS

SYSUDUMP

It is used to obtain a dump of the contents of various registers and variables in case of abnormal termination of the job. This dump is in hexadecimal. Information about various subroutines called within the job and data sets accessed is also provided.

request an unformatted dump of the user address space and task control blocks, as well as the system areas, if an abend occurs.

```
//SYSUDUMP DD SYSOUT=A
```

SYSABEND

Like SYSUDUMP it is used to obtain a dump of the contents of various registers and variables in case of abnormal termination of the job. The dump is in hexadecimal , and information about various subroutines called within the job, and data sets accessed is provided. In addition to this, the system nucleus at the time of termination o f a job is also listed to the output device or data set.

Concatenation of datasets

You can logically connect or concatenate (link together) sequential or partitioned data sets (PDSs OR PDSEs) for the duration of a JOB step.
To concatenate data sets, omit the DDnames from all the DD statements except the first.The data sets are processed in the same sequence as the DD statements defining them.
Ex:

```
//INPUT DD DSNAME=FGLIB,DISP=(OLD,PASS)
//        DD DSNAME=GROUP2,DISP=SHR
```

REFERBACKS

Reference to an earlier DD statement in the JOB
Ex:

```
//STEP1     EXEC PGM=PGM1
//DD1 DD    DSN=INPUT1.DSN, DISP=OLD, VOL=SER=11111
//*
//STEP2     EXEC PGM=PGM2
//DD2 DD    DSN=INPUT2.DSN, DISP=OLD, VOL=REF=*.DD1
```

Procedures

A procedure is a set of standard JOB step definitions that can be used repeatedly to perform a FUNCTION.Procedures are invoked by EXEC statements
Procedures comprise standard JOB step definitions and:
PROC statement
PEND statement
Symbolic parameters

Proc Statement

Marks the beginning of a procedure. Optional for a cataloged procedure
Required for an instream procedure, it must appear as the first control statement in the instream procedure. Name is optional for cataloged procedures but required for instream procedures.

PEND statement

Signifies the end of a procedure.Name is optional

Ex

```
//SORTPROC      PROC
//S1            EXEC PGM=SORT
//SORTIN        DD   DSN=SORTIN.DSN, DISP=SHR
//SORTOUT       DD   DSN=SORTOUT.DSN, DISP=(NEW,CATLG)
//SORTPROC      PEND
```

EX OF A CATALOGED PROCEDURE

```
//SORTPROC      PROC
//*  THIS IS A CATALOGED PROCEDURE
//STEPSORT      EXEC PGM=SORT
//SORTIN        DD   DSN=SORTIN.FILE, DISP=SHR
//SORTOUT       DD   DSN=SORTOUT.FILE, DISP=(,CATLG),
//              UNIT=SYSDA, VOL=SER=VOL001,
//              SPACE=(CYL, (10,2)),
//                  DCB=(LRECL=80, BLKSIZE=80,RECFM=F)
//SORTLIB       DD   DSN=SYS1.SORTLIB, DISP=SHR
//SYSOUT        DD   SYSOUT=A
//SORTPROC      PEND
```

EX

To invoke the above procedure:

```
//SORTJOB  JOB  (IBM, TRNG), `SORT PROGRAM', CLASS = A
//*
//STEP1        EXEC  PROC=SORTPROC
//
```

Ex

```
//JOB2        JOB (IBM, TRNG),`INSTREAM PROCEX',
//            CLASS=A
//* DEFINE THE INSTREAM PROC
//PROC1         PROC
//S1            EXEC PGM=PGM1
//PRODFILE      DD   DSN=PRODUCT.MASTER, DISP=OLD
//REPORT        DD   SYSOUT=A
//PROC1         PEND
//STEP1       EXEC PGM=UPDATE
//TXN DD      DSN=UPDATE.TXN, DISP=OLD
//MASTER      DD   DSN=PRODUCT.MASTER,DISP=OLD
//ERROR       DD   SYSOUT=A
//*  INVOKE THE PROC
//STEP2       EXEC PROC=PROC1
```

Two types of procedures:

Cataloged procedures
Instream procedures

In stream procedures

- Completely contained within a JCL JOB
- String of JCL statements appearing between a proc & pend statement within a JOB Available to only one JOB
- Upto 15 procedures can be defined within a JOB
- Within a JOB, can be invoked any number of times

When you place a procedure in the job input stream, it is called an in-stream procedure. An in-stream procedure must begin with a PROC statement, end with a PEND statement, and include only the following other JCL statements: CNTL, comment, DD, ENDCNTL, EXEC, IF/THEN/ELSE/ENDIF, INCLUDE, OUTPUT JCL, and SET.

You must observe the following restrictions regarding in-stream procedures:
Do not place any JCL statements (other than the ones listed above) or any JES2 or JES3 control statements in the procedure. Do not place an in-stream data set (one that begins with DD * or DD DATA) in the procedure. Do not define one in-stream procedure within another, that is, nested. Do not use an in-stream procedure if the procedure will be run as a started job under the MASTER subsystem, that is, includes a JOB statement and is started via a START command such as S procname,SUB=MSTR.

Cataloged Procedures
A procedure that you catalog in a library is called a cataloged procedure.
A cataloged procedure may consist of these JCL statements: CNTL, command, DD, ENDCNTL, EXEC, IF/THEN/ELSE/ENDIF, INCLUDE, OUTPUT JCL, and SET. Optionally, a cataloged procedure can begin with a PROC statement and end with a PEND statement. If coded, PROC must be the first statement in the procedure.

Cataloging a Procedure

The library containing cataloged procedures is a partitioned data set (PDS) or a partitioned data set extended (PDSE). The system procedure library is SYS1.PROCLIB. The installation can have many more procedure libraries with different names. You can also have procedures in a private library. The name of a cataloged procedure is its member name or alias in the library. When a cataloged procedure is called, the calling step receives a copy of the procedure, therefore, a cataloged procedure can be used simultaneously by more than one job. If you are modifying a cataloged procedure, do not run any jobs that use the procedure during modification.

Symbolic parameters
A symbolic parameter is a symbol preceded by an ampersand that stands for a parameter, subparameter or value. Use it for values that can change with each execution of the procedure

Makes the procedure more flexible & general purpose. Can be 1-7 characters, alphanumeric or national, preceded by & . A keyword parameter that can be coded on the EXEC statement like time, cond, parm cannot be a symbolic parameter.
Default values can be specified for a symbolic parameter in the operand field of the proc statement

Ex

```
//COMPILE PROC &MEM=,&T=1
//* DEFINING THE PROC WITH SYMBOLIC PARAMETERS
//STEP1 EXEC PGM=IKFCBL00, TIME=&T
//SYSIN DD DSN=IBM.COBOL.SOURCE(&MEM)
//*
//JOB1  JOB
//*  INVOKE THE PROC
//S1  EXEC PROC=COMPILE,MEM=PROG01
//*  INVOKE THE PROC
//S2  EXEC  PROC=COMPILE, MEM=PROG02,T=2
```

Overriding parameters

To override a parameter on the EXEC statement, when invoking the proc, follow on the proc statement with: PARAMETER.PROC STEPNAME=VALUE

Ex.

```
DEFINING THE PROC
//PROC1     PROC
//STEP1     EXEC PGM=PROG01,PARM=`910101'
//DD1 DD    DSN=DD1.DSN,DISP=OLD
//*
//STEP2     EXEC PGM=PROG02,PARM=`910101'
//DD2 DD    DSN=DD2.DSN,DISP=OLD
INVOKING THE PROC
//JOB1      JOB   CLASS=5
//S1        EXEC PROC=PROC1,STEP1.PARM=`910201'
```

Overriding DD statements

Recode the DD statement with the DDname proc stepname.DDname
No need to recode the entire DD statement. Just code the changes.
If overriding more than one DD statement, put the overriding statements in the same order as they appear in the procedure. To add new DD statements, put them after all the overriding statements

Ex: Define the proc

```
//PROC1     PROC
//STEP1     EXEC PGM=PROG01
//INPUT     DD   DSN=INPUT.FILE,DISP=OLD
```

```
//OUTPUT   DD     DSN=OUTPUT.FILE,DISP=(NEW,CATLG),
//                UNIT=DISK,VOL=SER=111111,SPACE=(TRK,10),
//                DCB=(LRECL=100,BLKSIZE=1000,RECFM=10)
```

GENERATION DATA GROUPS (GDG)

The concept of Generation Data Groups has do with the chronological and functional relationships between the data sets. That is Generation Data Groups or GDGs are a group of datasets which are related to each other chronologically and functionally. Generations can continue until a specified limit is reached. This limit specifies the total number of generations that can exist at one time. Once this limit is reached, the oldest generation can be deleted. Thus, the cyclical nature of adding new generations and deleting the oldest one each time the limit is reached.

In order to create a GDG, the following must be specified to the operating system:

- The name of the GDG
- The number of generation that are to be retained
- Whether or not the oldest generation is to be uncataloged once the limit for the number of generations that are to be retained is reached.
- Whether or not an entry for a data set that is deleted from a GDG is to be uncataloged and physically deleted from the volume that it resides on.
- Once a model for a GDG has been established, the system must be informed each time a data set is to be added to it.
- The system must be able to specify the generation number of each data set within a GDG.
- The system must be informed if a data set within a GDG is to be deleted.
- The system must be informed if only the index of a GDG is to be deleted.
- The system must be informed if a data set within a GDG is to be deleted, even if its retention period has not expired.
- The system must be informed if the entire GDG is to be deleted; this includes the index and all related data sets.

JCL parameters and subparameters used to Create GDGs

FUNCTION	**Corresponding JCL parameters**
Creation of Generation Data Group(GDG)	Before GDG can be created its index is created and cataloged. The index conveys information relating to the next seven features. The DEFINE GDG statement in the IDCAMS utility is used to create the index of a GDG.
Specifying name of the GDG	NAME coded on the DEFINE GDG statement
Specifying number of generations to be retained	LIMIT coded on the DEFINE GDG statement
Uncataloging oldest generation of a GDG once the limit is reached	NOEMPTY code on the DEFINE GDG Statement

Physically deleting the entry of a GDG from the volume that it resides on	SCRATCH coded on the DEFINE GDG statement
Uncataloging the entry of a GDG from The volume that it resides on, without physically deleting it	NOSCRATCH coded on the DEFINE GDG statement
Defining a model for GDG which will specify the format of all applicable parameters for data sets that are added to the GDG	Creating a user-defined model
Adding a data set to a GDG	Data set is created in usual way, the name of the model containing the GDG DCB parameters is coded in the DCB parameter on the DD statement
Deleting a data set within a GDG	DELETE code on the DISP parameter in the DD statement
Deleting the GDG index only	IBM supplied IDCAMS utility is used to delete index only
Deleting entire GDG. Index and data sets inclusive	FORCE coded on the DD statement for the GDG
Deleting a data set within a GDG even though its retention period has not expired.	PURGE code on the DD statement for that GDG

Creating your own model

```
//JOB1            JOB   A50,'ARICH'
//STEP1     EXEC PGM=IDCAMS
//SYSIN           DD    *
   DEFINE GDG(NAME(ACCOUNTS.MONTHLY)
   LIMIT(5)   NOEMPTY    SCRATCH)
//STEP2     EXEC PGMIEFBR14
//MODEL1    DD    DSN=ACCOUNTS,MONTHLY,
//                DISP=(NEW,KEEP,DELETE),
//                UNIT=SYSDA,
//                SPACE=(TRK,0),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
```

The following JCL creates the index for the GDG:

```
//SYSIN           DD    *
   DEFINE GDG(NAME(ACCOUNTS.MONTHLY)   -
   LIMIT(5)   NOEMPTY     SCRATCH)
```

and this JCL creates the model:

```
  //STEP2     EXEC PGMIEFBR14
//MODEL1    DD    DSN=ACCOUNTS,MONTHLY,
//                DISP=(NEW,KEEP,DELETE),
//                UNIT=SYSDA,
//                      SPACE=(TRK,0),
//              DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
```

Creating a Generation Data set

```
//STEP3                 EXECPGM=GDG1
//FILE1                 DD     DSN=ACOUNTS.MONTHLY(+1),
//                      DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//                      DCB=(MODEL.DCB,REFCM=FB,LRECL=80,BLKSIZE=800)
```

The program GDG1 is executed a new generation data set belonging to the GDG ACCOUNTS.MONTHLY is created via the statement:

```
//MODEL1  DD  DSN=ACCOUNTS.MONTHLY(+1)
```

The syntax (+1) creates a new generation relative to the current generation which is referenced as generation 0. all existing generations are pushed down by one level at the end of the job. The generation numbers will not be updated until the end of the job.

Deleting Generation Data Group Index

The GDG index can be deleted using DELETE parameter in the IBM supplied IDCAMS utility . deletion of the index will result in references to any generation data sets of the same name generating an error. Thus resulting in abnormal termination of the job or job step.

```
//JOB1        JOB   A50,'ARICH'
//S1    EXECPGM=IDCAMS
//SYSIN        DD *
DELETE(ACCOUNTS.MONTHLY)GDG
/*
```

PURGE coded on DELETE statement

The PURGE parameter can be coded in conjunction with the DELETE statement to delete the GDG index, even if its retention period has not expired. When a data set is created a retention period is automatically record for it. When this period is up, the system automatically deletes that data set from the system. The PURGE parameter can be used to override this retention period.

```
//JOB1        JOB   A50,'ARICH'
//S1    EXECPGM=IDCAMS
//SYSIN        DD *
DELETE(ACCOUNTS.MONTHLY)GDG PURGE
/*
```

FORCE CODED ON DELETE STATEMENT.

The force parameter can be coded on the DELETE statement to delete the GDG index, the model and all related generation data sets, if they exist.

```
//JOB1        JOB   A50,'ARICH'
//S1    EXECPGM=IDCAMS
//SYSIN        DD *
```

```
DELETE(ACCOUNTS.MONTHLY)GDG FORCE
/*
```

PASSING DATA SETS

A data set read or written in a step can be given a disposition of pass to pass it on to a following step in the same JOB. It keeps information normally in the catalog(dsn, unit,vol,dcb).

Ex

PASS & DELETE

```
//STEP1    EXEC PGM=CREATE
//OUT      DD   DSN=&&TEMP,DISP=(NEW,PASS),UNIT=DISK,
//              DCB=(RECFM=FB,LRECL=80)
//STEP2    EXEC PGM=READ
//IN       DD   DSN=&&TEMP,DISP=(OLD,PASS)
//STEP3    EXEC PGM=SORT
//OUT      DD   DSN=&&TEMP,DISP=(OLD,DELETE)
```

PASS & CATALOG

```
//STEP1    EXEC PGM=CREATE
//OUT      DD   DSN=CATEMP,DISP=(NEW,PASS),UNIT=DISK,
//              DCB=(RECFM=FB,LRECL=80)
//STEP2    EXEC PGM=READ
//IN       DD   DSN=CATEMP,DISP=(OLD,CATLG)
```

Ex

Disposition if a step abends

```
//STEP1 EXEC PGM=CREATE
//OUT    DD     DSN=CATEMP,DISP=(NEW,PASS),UNIT=DISK,
//               DCB=(RECFM=FB,LRECL=80)
//STEP2 EXEC PGM=READ
//IN     DD     DSN=CATEMP,DISP=(OLD,DELETE,KEEP)
```

If a temporary data set abends, it is always deleted
If disp=(old, keep), for abend it is keep
If disp=(old, delete), for abend it is DELETE

The if/then/else/endif construct

The IF/THEN/ELSE/ENDIF construct in JCL allows the outcome of one JOB step to determine whether following steps are executed.

EX

```
//   IF (STEP4.RC EQ 4 ) THEN
//STEP6 EXEC PGM=ONE
//   ELSE
//STEP8 EXEC PGM=GSJ
//  ENDIF
```

The logical expression can contain 'EQ, NE, GT, LT, GE LE' and connect expression with 'AND, NOT, OR'
// IF NOT (STEP1.RC.EQ 1 AND STEP2.RC EQ 1) THEN

Test for ABEND
```
//   IF (STEP4 LE 4 AND STEP1.ABEND) THEN
//    IF   ((STEP6.RC EQ 1 AND STEP7.RC EQ  4)  OR
//   (STEP8.RC EQ 1 AND STEP9.RC EQ  4) THEN
```

RULES:
1) THE CONSTRUCT MUST ALWAYS INCLUDE AT LEAST ONE EXEC STATEMENT.
2) YOU CAN'T PLACE JOB, JOBCAT, JOBLIB

Limitations

STEP A JOB can have a maximum of 255 JOB steps. This maximum includes all steps in any procedures the EXEC statements call.

DD Maximum number of DD statements in a JOB step is 3273, in a JES2 system.

PROCEDURES You can code a maximum of 15 in-stream procedures in any JOB. An in-stream procedure cannot be defined within another procedure, or nested.

Include Is used only on MVS/ESA version 4 or higher systems to name an include group. An include group is a set of one or more valid JCL statements that are stored together in a jcllib data set or a system procedure library, and that are read in and used to replace the include statement itself within the JOB.
//{NAME} INCLUDE MEMBER=MEMNAME {COMMENTS}

Utilities Utility programs (or utilities) are generalized programs used in installations to perform common data processing functions. For EX copying files, which is a common function. It would be a waste of programmer time if one were to write a program each time a file were to be copied. So a utility programs is provided that can copy any sequential file.

Typical MVS Utilities
1. Application utilities to create change, or delete data sets
2. Sort/Merge
3. Access Method Services (AMS) to define and maintain VSAM datasets
4. System utilities

Some common utilities

Function	Utility name
Copy sequential files	IEBGENER
copy partitioned data sets / compress	IEBCOPY
Catalog /Uncatalog / Rename data sets/	IEHPROGM
Include members of partitioned datasets when implementing copy command	SELECT statement coded in the COPY command of the IEBCOPY
Exclude members of portioned data sets when implementing a copy command	EXCLUDE statement coded in the COPY command of the IEBCOPY
Compare sequential / partitioned data sets	IEBCOMPR
Code functions available on the DD statement without executing a program	IEFBR14

JCL REQUIRED TO EXECUTE UTILITIES

The following is the JCL in pseudo-code that executes a utility program.

```
//jobname       JOB      (acctinfo,progname)
//stepname      EXEC     PGM=utility-name,PARM=parm_value
//Printname     DD       Sysout=print-device-class
//inputfile     DD       input-file
//outputfile    DD       output-file
//workfile      DD       output-file
//inputdata     DD   *
/*
//
```

IEHPROGM

- Scratches datasets or members of a PDS on a DASD vol.
- Renames datasets or members
- Catalogs or Uncatalogs a dataset
- Builds or deletes a catalog index
- Builds or deletes a generation index (define GDG)

Ex for IEHPROGM (FOR CATALOGING DATA SETS)

```
//IEH      JOB    A50,'ARICH'
//S1        EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=*
//SYSUT1    DD UNIT=3390, VOL=SER=SMS005
//SYSUT2    DD UNIT=TAPE,VOL=SER=SMS007
//SYSIN          DD  *
              CATLG   DSNAME=DATA1,VOL=SER=SMS007
/*
//
```

In SYSUT1 and the permanently mounted unit, is identified. This is the volume called SMS005. In SYSUT2 the location of the data set that is to be cataloged is identified. The data set resides on volume SMS007 of the device called TAPE. CATLG statement identifies the name of the file (DATA1) and the device and serial number that it currently resides on.

IEHLIST

- List all or part of a catalog
- List the entries of a directory in a PDS
- List the VTOC of a DASD volume or dataset

IEBGENER

- Copies sequential data sets from one device to another
- Create a PDS from a sequential dataset
- Expand or add members to a PDS
- Produce an edited dataset
- Reblock or change logical record lengths of a dataset

Ex:

```
// STEP1          EXEC PGM=IEBGENER, REGION=0M
// SYSPRINT       DD SYSOUT=*
// SYSUT1   DD DSN=DSNNAME, DISP=SHR     INPUT DATA
// SYSUT2   DD DSN=DSNNAME, DISP=SHR     OUTPUT DATA
// SYSIN          DD DUMMY
//
```

IEBCOPY for partitioned data sets

- Backup (unload)
- Restore (reload)
- Copy
- Compress
- Updates

Ex:

```
// STEP1          EXEC PGM=IEBCOPY, REGION=0M
// SYSPRINT       DD SYSOUT=*
// DD1            DD DSN=AIT1, DISP=SHR      INPUT DATA
// DD2            DD DSN=AIT2, DISP=NEW,unit=tape1      OUTPUT DATA
// SYSIN          DD*
    COPY INDD=DD1,OUTDD=DD2
/*
//
```

The first dataset is defined in the DD1 statement. this file is shared among users. The characteristics of the second dataset is defined in the DD2 statement.

```
//SYSIN  DD *
    COPY INDD=DD1,OUTDD=DD2
```

```
/*
```

This statement specifies that the dataset defined DD! Will be input data set while that defined DD2 will be the output data set. AIT1 will be copied into AIT2.

Compressing data sets

```
// STEP1          EXEC PGM=IEBCOPY, REGION=0M
// SYSPRINT       DD SYSOUT=*
// DD1            DD DSN=COBOL.AIT1, DISP=SHR   INPUT DATA
// SYSIN          DD*
    COPY INDD=DD1,OUTDD=DD1
/*
//
```

IEBCOMPR

- Compare one PDS to another
- Compare one sequential dataset to another sequential dataset on a record by record basis.

Comparing Two Partitioned Data Sets

```
//JOB1          JOB   A50,'ARICH'
// STEP1          EXEC PGM=IEBCOPY, REGION=0M
// SYSPRINT       DD SYSOUT=*
// DD1            DD DSN=AIT1.SOURCE, DISP=SHR
// DD2            DD DSN=AIT2.SOURCE, DISP=SHR
// SYSIN          DD*
    COMPARE TYPORG=PO
/*
//
```

IEFBR14
Null or dummy program used to allocate or delete datasets.

```
//CREATE     JOB        A50,'ARICH'
//STEP3      EXEC PGM=IEFBR14
//DD1        DD   DSN=COBOL.DATA1,DISP=(NEW,KEEP),UNIT=SYSDA,
//                SPACE=(TRK,(4,2)),
//                DCB=(LRECL=80,RECFM=FB,BKSIZE=800)
//
```

SORT UTILITY

The sort function in the SORT utility takes records from an input file, sorts the records, and places them in an output file.

ISSUES RELATED TO SORTS

In order to sort a file, the following must be addressed:

- The name of the file that is to be sorted.

- The name of the file in which the sorted file is to be stored.
- The location of the record from which to start sorting.
- The location of the record at which to stop sorting, or the length from the starting point, for which the sort is to be implemented.
- Whether the records are to be sorted in ascending or descending order.
- The type of data stored in the data set that us being sorted. For EX it can be character, packed, etc.

Parameters on SORT Command

The SORT command is used to specify features specific to the sort itself. The syntax of this command. SORT FIELDS =(start-location,length,sort-sequence,format)
Start-location is the position from which length number of bytes will be used as the sort criteria. Sort-sequence specifies whether the file is to be sorted in ascending or descending order. Format specifies the type of data that will be sorted.

```
//SORT1         JOB  …  …
//STEP1         EXEC PGM=SORT
//SYSOUT   DD SYSOUT=A
//SYSPRINT      DD SYSOUT=A
//SORTIN        DD DSN=MTRGXXX.TRG.DATA,DISP=SHR
//SORTOUT  DD DSN=MTRGXXX.RECORDS.SORT,
                DISP= (NEW,CATLG,DELETE),
//UNIT=SYSDA, SPACE=(CYL,(1,1),RLSE)),
//DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SORTWK01      DD UNIT=SYSDA,SPACE=(CYL,(2,1),RLSE)
//SORTWK02      DD UNIT=SYSDA,SPACE=(CYL,(2,1),RLSE)
//SYSIN         DD*
     SORT FIELDS=(2,5,CH,A)
/*
```

MERGE UTILITY

The MERGE function combines one or more stored input files into a single, ordered, sequential file. Merging faster and more efficient, since the input files are already in sorted order before they are merged together.
MERGE FIELDS=(start-location,length,merge-sequence,format)
Start-location is the starting location in the record from where the merge is initiated. Length is the number of bytes, from this start location, that are to be used as the merge criteria. Merge-sequence specifies whether the file is to be merged in the ascending or descending order. Format specifies the format of the data that is being merged.

Rules for coding

- The record format of all input files that are being merged must be the same.
- If the input files are composed of fixed length records, then the logical record length of these files must also be the same.
- If the input files contain block sizes of different lengths, then those with larger block sizes should be placed before those with smaller block sizes.

- The ddname of the input files that are to be merged must begin with the letters SORTIN, followed by digit number. The digit number for all files that are to be merged must be in ascending order.

```
//JOB1          JOB   A50,'ARICH'
//STEP1    EXEC PGM=SORT
//SYSOUT   DD   SYSOUT=A
//SYSPRINT      DD    SYSOUT=A
//SORTIN01      DD    DSN=MTRG.DATA1,DISP=SHR
//SORTIN02      DD    DSN=MTRG.DATA2,DISP=SHR
//SORTIN03      DD    DSN=MTRG.DATA3,DISP=SHR
//OUTFILE  DD   DSN=MTRG.MERGE,DISP=(NEW,CATLG,DELETE),
//              UNIT=SYSDA,
//              SPACE=(CYL,(2,1),RLSE),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN         DD  *
MERGE FIELDS=(1,3,A,CH)
/*
//
```

COMMON ABEND CODES

S013	Member not found
S0C4	Caused by subscript being out of range
S0C5	Caused by invalid address specification i.e. the address points to an instruction, control word, or data outside the available real storage
S0C7	Caused by bad data, with the program is unable to detect
S222	Caused by job being canceled by the operator
S322	Caused when CPU time assigned to the job, job step, or procedure has been executed
S522	Caused when a wait state exceeds an installation defined time limit
SB37	Insufficient secondary space
SD37	Insufficient primary space
SE37	Insufficient space in a PDS (Partitioned Dataset)

VSAM Fundamentals

What is VSAM?

VSAM is a high-performance access method used in OS/VS & ZOS operating systems. VSAM software resides in virtual storage along with the program that needs its services for the manipulation of data on a direct access storage device (DASD). VSAM acts as interface between Operating System and Application Program.

Concepts

In almost every type of programming activity on an IBM Mainframe, one is bound to come across files supported by VSAM, the Virtual Storage Access Method. VSAM is by far the most commonly used access method on MVS systems. VSAM does more than just replace non-VSAM access methods with VSAM access methods; it also provides efficiency improvements and a comprehensive catalog facility that centralizes information about all VSAM data sets. In addition, the multi-functional utility program – AMS – has a variety of file related functions for VSAM as well as non-VSAM files.

DASD Labels and Space Management

Direct Access Storage Devices, or DASDs, allows direct access to large quantities of data. They are used not only to store user programs and data but also to store operating system programs and data. The various models of DASD units can be divided into two basic categories, depending on the format in which they store data – CKD devices and FBA devices. On count-key-data (CKD) devices, data is stored in variable-length blocks, while in fixed-block architecture (FBA) devices data is stored in 512-byte blocks. CKD devices are more common.
An important function of the operating system and access methods in managing how space on a DASD volume is allocated to the files that reside on it. The system must

- Keep track of the location of existing files
- Be able to add new files to the volume
- Allocate additional space to the existing files
- Remove files from a volume.

The information necessary to manage all this is stored in special DASD records called labels.

VSAM history

1973 - VSAM was introduced with ESDS & KSDS
1975 - Alternate Index, RRDS, Catalog Recovery features introduced
1979 - VSAM re-introduced with ICF (Integrated Catalog Facility)
1983 - DFP / VSAM ran under MVS / XA & (Data Facility Product)
1987 - DFP / VSAM 2.3 introduced with LDS
1988 - VSAM 3.1 under MVS / ESA with DFSMS.
1991 - VSAM 3.3 introduced with variable length records for RRDS.

There are two types of labels

1. Volume Labels – the VOL 1 Label

All DASD volumes must contain a volume label, called the VOL 1 Label. This label is always in the same place on a disk volume – third record of track zero in cylinder zero. It identifies the volume with a volume serial number, vol-ser, which is a unique six-character number. It also contains the disk address of the VTOC.

2. File Labels

The VTOC, Volume Table of Contents, is a special file that contains the labels for the files on the volume. These file labels, also called Data Set Control Blocks or DSCBs, have several formats, called Format-1, Format-2, and so on.
Each Format-1 DSCB in a VTOC describes a file. In addition to the DASD location of the file, it contains descriptive information about the file, such as whether it is sequential or indexed sequential, its record length, and so on. Space is allocated to DASD files in contiguous areas called extents.
There are four major processing environments under which VSAM is used.
It can be used as a stand-alone product. VSAM data sets can be created and processed by a powerful software product called IDCAMS.
Batch application programs in COBOL, PL/I, Assembler, etc. can access VSAM data sets.In data set management, PDS directories, catalogs, and generation data groups can be manipulated through specific IDCAMS commands. GDGs and catalogs are themselves VSAM data sets, Also, the IDCAMS utility can be used to process older, non-VSAM data set organizations. Many major IBM software products, such as CICS, IMS, DB2 and TSO, use VSAM data sets in a majority of applications. In some of them, VSAM data sets are the only ones that are compatible.
VSAM is available under all IBM's Mainframe operating systems; MVS, VM, and VSE. The MVS implementation of VSAM is the most comprehensive. Also, VSAM data sets can be stored on almost all standard disk drives, such as 3390, 3380, 3350, 3370, etc. Present and future mainframe software released by IBM will be designed to process records storage in VSAM data sets. VSAM is going to be around for a long time to come.

Advantages & Disadvantages Compared to other Access Methods

VSAM is superior to other access methods in the following aspects. The retrieval of records is faster because of an efficiently organized index. The index is small because of a key compression algorithm used to store and retrieve its records. Imbedded free space makes the insertion of records easy, and data sets therefore require less reorganization. The deletion of records in VSAM, unlike that in ISAM, means that they are physically deleted, thus allowing the reclaiming of free space within the data set.
VSAM data sets are device-independent. VSAM catalogs and data sets are portable between operating systems. Records can be accessed randomly by key or by address and can also be accessed sequentially at the same time.

Some of the major disadvantages of VSAM

To take advantage of the partial self-reorganization capabilities of VSAM data sets, free space must deliberately be left. For data sets that are used for read only purposes, no free space is required. Except for read-only data sets, the integrity of VSAM data sets in cross-system and cross–region sharing must be controlled by the user. Data integrity must be a prime consideration in the initial design of applications that will be shared across systems.

VSAM provides four types of file organizations along with their respective access methods and utilities. Access methods are system software that provide technical details to system developers.

- Entry Sequenced Data Set – ESDS - is like a standard sequential (QSAM) data set.
- Relative Record Data Set --- RRDS - is like a direct file (BDAM).
- Key Sequenced Data Set – KSDS- is like an indexed sequential access method file.
- Linear Data Set – LDS – with no record organization.

These three data set organizations were initially developed to replace existing non-VSAM data sets, such as QSAM, ISAM, BDAM, QISAM and BSAM, but this could obviously be done overnight. So VSAM data sets had to perforce co-exist with non-VSAM data sets. The VSAM data set organizations are superior to the native access methods. VSAM provides for alternate indexes, a feature not available in native access methods. It has admirable catalog facilities that stores more information about VSAM and other data sets. The powerful AMS utility in VSAM provides a variety of services dealing with catalogs, files, security, file management, etc. It provides comprehensive support for application development in many environments such as COBOL, PL/I, FORTRAN, Assembler and CICS.

Entry Sequence Data Sets (ESDS)

An ESDS is a sequential file, in which records are typically retrieved in the order in which they were written to the data set and additions are always made at the end of the file. Each record can be identified by a relative byte address or RBA. Records cannot be physically deleted. Since records in an ESDS are not sequenced on any key field, there is no primary key index component. Imbedded free space is not allocated at the time of allocation of an ESDS, since records are not added to the middle of the file. Records in an ESDS may be of variable length.

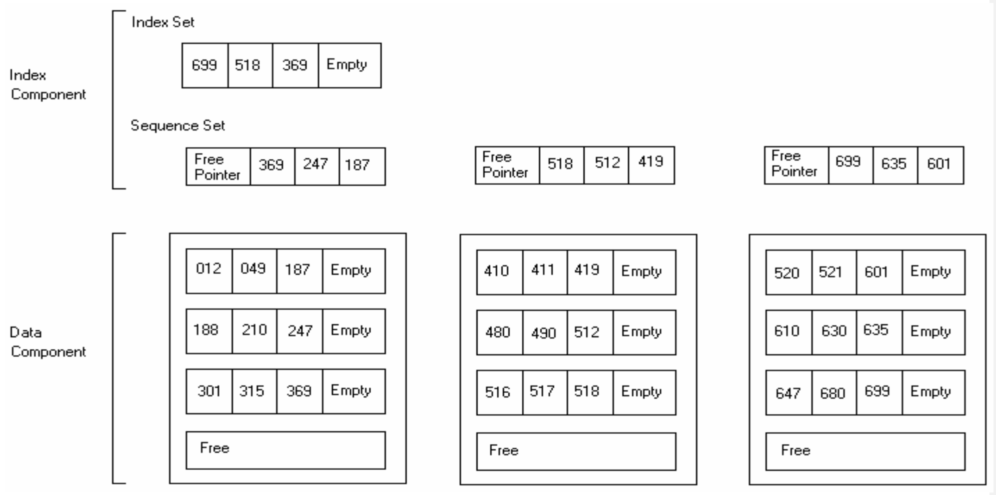
Relative Record Data Sets (RRDS)

In an RRDS the entire data set is a string of fixed length slots. Each slot occupies a fixed position and is identified by its position relative to the first slot of the data set. The relative position of each slot is called relative record number (RRN). An RRN is an integer that identifies the position of the slot no the value of a particular field within the slot. Each slot of an RRDS may or may not contain a record. Records in an RRDS may be inserted, retrieved, updated, and deleted both sequentially and randomly. Records in an RRDS are always fixed in the length. RRDS has only data component. When a record is deleted from a particular slot, the slot remains in the same physical location although a record does not exist in that slot anymore.

Key Sequenced Data Sets (KSDS)

A KSDS can be processed sequentially or randomly. When used for sequential processing, the records are processed in the order of the key values in the file index. A KSDS consists of two components, a data component and an index component. The data component contains the record and the index component contains the indexes necessary to access the records.

The index component of a KSDS has two parts: a sequence set and an index set. The sequence set is the lowest level of the index. There is one sequence set for each control area. Each sequence set record contains an index entry for each CI in the corresponding control area. This index entry contains the highest key value in the respective CI. Within the sequence set records, index entries are stored in sequence from right to left. The sequence set is searched to determine which CI in the data component contains a particular record.



The search sequence is as follows:

The index set is searched to determine which sequence set record to use.

The sequence set record is then searched to locate which control level to use.

Finally, the control interval is searched to locate the needed record.

An overview of the major advantages of using VSAM is as follows:

Very little JCL coding is needed for processing a VSAM file. The DD statement is a one-line statement needing only the DSNAME and DISP parameters. Records can be accessed through one or more secondary keys using alternate indexes built and maintained by VSAM. Up to 256 record fields can easily be used as keys.In KSDS and ESDS, any field can be used to sequentially or randomly process the records.

Byte addressing can be used as an alternative method to randomly access and process records in a KSDS and ESDS using their relative byte addressing (RBA). This imparts to a disk data set a concept generally associated with main memory and amply increases the processing capabilities of a VSAM data set. Status codes are returned following every I/O operation. Each IDCAMS command returns a condition code.

VSAM Clusters

In VSAM terms, a file is often called a cluster. A cluster is a set of catalog entries that represents a file catalog.

It consists of two components:

- A data component, which represents the actual records of a file
- An index component which represents the indexes for KSDS

VSAM Catalog Concepts

VSAM provides a comprehensive catalog facility that stores information about VSAM data sets and other files.

There are two types of catalogs:

- Master Catalogs
- User Catalogs

Each OS system has just one master catalog, but can have an unlimited number of user catalogs. All user catalogs must be cataloged in the master catalog and all VSAM data sets must be in the master catalog or the user.

File access methods

- Data (Records) is retrieved
- Sequential (Reading from beginning to end)
- Random (Records are read by the value in the key)
- Direct (Records are read based on their physical location/address on disk)
- VSAM provides all these methods
- One access method supporting all types of data retrieval
- Comparison with other Access Methods

Comparison of an indexed sequential data set and a KSDS

Characteristic	**Indexed Sequential /ISAM**	**KSDS / VSAM**
Data set allocation	JCL parameters with DISPOSITION of NEW	Access Method Services (IDCAMS utility)
Data set deletion	DISPOSITION parameter of DD Statement	Access Method Services (IDCAMS utility)
Alternate Index support	No	Yes
Deleting records	Records logically deleted	Records physically deleted & space reclaimed
Reorganization of data set	Needed more often	Needed less often

Disk space requirement	Less	Greater because of imbedded free space.
Concurrent sequential and random access	Not supported unless two blocks are created	Supported in one access control block(ACB)

Comparison of an Physical sequential (PS) data set and a ESDS

Characteristic	Physical Sequential	ESDS
Data set allocation	JCL parameters with DISPOSITION of NEW	Access Method Services (IDCAMS utility)
Data set deletion	DISPOSITION parameter of Delete Statement	Access Method Services (IDCAMS utility)
Alternate Index support	No	Yes (up to 253)
Can records be altered and replaced?	Yes (but record length cannot be changed)	Yes(but record length cannot be changed)
Random access of records	No	Yes(thru RBA)

Comparison of an Direct organization file and a RRDS

Characteristic	Physical Sequential	ESDS
Allocation & Deletion of data set	JCL parameters with DISPOSITION	Access Method Services (IDCAMS utility)
Device independent	Mostly No	Yes
Variable length records	Yes	No

VSAM File Organization
The physical organization of VSAM data sets differs considerably from those used by other access methods. VSAM data sets are held in control intervals and control areas; the size of these is normally determined by the access method, and the way in which they are used is not visible to you.

You can use three types of file organization with VSAM:
VSAM Sequential file organization: Also referred to as VSAM ESDS (Entry-Sequenced Data Set) organization.
VSAM Indexed file organization: Also referred to as VSAM KSDS (Key-Sequenced Data Set) organization
VSAM Relative file organization: Also referred to as VSAM fixed-length or variable-length RRDS (Relative Record Data Set) organization

The term VSAM relative record data set (or RRDS) is used to mean both relative-record data sets with fixed-length records and with variable-length records, unless they need to be differentiated.
VSAM data sets can be processed in COBOL for MVS & VM programs only after they are defined with access method services.

VSAM Sequential File Organization

In VSAM sequential file organization (ESDS), the records are stored in the order in which they were entered. VSAM entry-sequenced data sets are equivalent to QSAM sequential files. The order of the records is fixed. Records in sequential files can only be accessed (read or written) sequentially.

After you have placed a record into the file, you cannot shorten, lengthen, or delete it. However, you can update (REWRITE) a record if the length does not change. New records are added at the end of the file.

VSAM Indexed File Organization

In a VSAM indexed file (KSDS), the records are ordered according to the collating sequence of an embedded prime key field, which you define. The prime key consists of one or more consecutive characters in the records. The prime key uniquely identifies the record and determines the sequence in which it is accessed with respect to other records. A prime key for a record might be, for EX an employee number or an invoice number. In your COBOL program, code this key by using the clause:

RECORD KEY IS data-name

where data-name is the name of the key field as you defined it in the record description entry in the DATA DIVISION. You can also code one or more alternate keys to use for retrieving records. Using alternate keys, you can access the indexed file to read records in some sequence other than the prime key sequence. For EX you could access the file through employee department rather than through employee number. Alternate keys need not be unique. More than one record will be accessed, given a department number as a key. This is permitted if alternate keys are coded to allow duplicates. You define the alternate key in your COBOL program with the ALTERNATE RECORD KEY IS clause:

ALTERNATE RECORD KEY IS data-name

Where data-name is the name of the key field as you defined it in the record description entry in the DATA DIVISION.

To use an alternate index, you need to define a data set (using access method services) called the alternate index (AIX). The AIX contains one record for each value of a given alternate key; the records are in sequential order by alternate key value. Each record contains the corresponding primary keys of all records in the associated indexed files that contain the alternate key value.

VSAM Relative-Record File Organization

A VSAM relative record data set (RRDS) contains records ordered by their relative key—the relative key being the relative record number representing the record's location relative to where the file begins. The relative record number identifies the fixed or variable-length record. Your COBOL program can use a randomizing routine that will associate a key value in each record with the relative record number for that record. Although there are many techniques to convert a record key to a relative record number, the most commonly used randomizing algorithm is the division/remainder

technique. With this technique, you divide the key by a value equal to the number of slots in the data set to produce a quotient and remainder. When you add one to the remainder, the result will be a valid relative record number.
Relative files are identified in your COBOL program by the ORGANIZATION IS RELATIVE clause. Use the RELATIVE KEY IS clause to associate each logical record with its relative record number.

Alternate indexes are not supported for VSAM RRDS.

RRDS with Fixed-Length Records
In a VSAM fixed-length RRDS, records are placed in a series of fixed-length slots in storage. Each slot is associated with a relative record number. For EX in a fixed-length RRDS containing 10 slots, the first slot has a relative record number of 1, while the 10th slot has a relative record number of 10. Each record in the file occupies one slot, and you store and retrieve records according to the relative record number of that slot.
When you load the file, you have the option of skipping over slots and leaving them empty.

RRDS with Variable-Length Records
In a VSAM variable-length RRDS, the records are ordered according to their relative record number. Records are stored and retrieved according to the relative record number you set. When you load the file, you have the option of skipping over relative record numbers. Unlike fixed-length RRDS, variable-length RRDS does not have slots. Instead, user-defined free space allows for more efficient record insertions. VSAM variable-length RRDS is supported with VM/ESA and MVS/ESA with MVS/DFP Version 3 Release 3.

Simulating Variable-Length RRDS:
If you cannot or prefer not to use VSAM variable-length RRDS, COBOL for MVS & VM also provides another way for you to have relative-record data sets with variable-length records. This support, called COBOL simulated variable-length RRDS, is provided by the SIMVRD/NOSIMVRD run-time option. When you use the SIMVRD option, COBOL for MVS&VM simulates variable-length RRDS using a VSAM KSDS.
The coding you use in your COBOL program to identify and describe VSAM variable-length RRDS and COBOL simulated variable-length RRDS is similar. How you use the SIMVRD run-time option and whether you define the VSAM file as a RRDS or KSDS differs, however.

To use a VSAM variable-length RRDS, do the following:
1. Define the file in your COBOL program with the ORGANIZATION IS RELATIVE clause.
2. Use FD statements in your COBOL program to describe the records with variable length sizes.
3. Use the NOSIMVRD run-time option.
4. Define the VSAM file through access method services as an RRDS.

To use a COBOL simulated variable-length RRDS, do the following:

1. Define the file in your COBOL program with the ORGANIZATION IS RELATIVE clause.
2. Use FD statements in your COBOL program to describe the records with variable length sizes.
3. Use the SIMVRD run-time option.
4. Define the VSAM file through access method services as a KSDS with a 4-byte key.

File Access Modes

You can only access records in VSAM sequential files sequentially. You can access records in VSAM indexed and relative files in three ways: sequentially, randomly, or dynamically.

Sequential access

Code ACCESS IS SEQUENTIAL in the FILE-CONTROL entry.
For indexed files, records are accessed in the order of the key field selected (either primary or alternate).
For relative files, records are accessed in the order of the relative record numbers.

Random access

Code ACCESS IS RANDOM in the FILE-CONTROL entry.
For indexed files, records are accessed according to the value you place in a key field.
For relative files, records are accessed according to the value you place in the relative key.

Dynamic access

Code ACCESS IS DYNAMIC in the FILE-CONTROL entry. Dynamic access is a mixed sequential-random access in the same program. Using dynamic access, you can write one program to perform both sequential and random processing, accessing some records in sequential order and others by their keys.

For EX suppose you had an indexed file of employee records, and the employee's hourly wage formed the record key. Also, suppose your program was interested in those employees earning between Rs.7.00 and Rs.9.00 per hour and those earning Rs.15.00 per hour and above. To do this, retrieve the first record randomly (with a random-retrieval READ) based on the key of 0700.

Next, begin reading sequentially (using READ NEXT) until the salary field exceeds 0900. You would then switch back to a random read, this time based on a key of 1500. After this random read, switch back to reading sequentially until you reach the end of the file.

VSAM Internals

Data organization

Let us take an example of books
In early days books were sequential from Start to End with no breaks (corresponds to Linear Dataset).
Afterwards, organized into chapters and paragraphs (corresponds to Entry Sequenced Dataset).
Page numbers were introduced to identify a page (corresponds to Relative Record Dataset).
Indexed were later introduced to locate a topic directly (corresponds to key sequenced Dataset).

Traditional access methods

- QSAM (Queried Sequential Access Method)
- BSAM (Basic Sequential Access Method)
- for 'flat' files
- ISAM (Index Sequential Access Method)
- for Index files
- BDAM (Basic Direct Access Method)
- for direct access files

IDCAMS

- Handles VSAM data sets (exclusively)

Some of the functions

• Creates	(DEFINE)
• Copies	(REPRO)
• Prints	(PRINT)
• Delete	(DELETE)
• Lists Characteristics	(LISTCAT)

VSAM data set organization

- VSAM Data Set can contain three major components
- CLUSTER (Catalog entry)
- INDEX
- DATA (Actual data)

Data Set is referred by cluster name in JCL

Internal organization of VSAM

In physical sequential data sets, records are clustered in blocks and a block can have one or more records. VSAM does not use blocks in the traditional sense. It has another similar unit of record storage called Control Interval. A control interval may contain one

or more records. A control interval is the smallest unit of information storage transferred between the buffers and a direct access storage device.
Control intervals are part of a larger storage structure called a control area(CA). A control area may consist of many control intervals.

Storage Organization of a VSAM Data Set
Storage organization of the data component of a VSAM data set. The data set in this ex has 3 control areas. Each control area has 3 control intervals. Each control interval has a maximum of 8 fixed length records. Control fields are used by VSAM for internal housekeeping information for the control interval.

- Each component consists of one or more control areas.
- A control area may consist of many control intervals.
- A control interval may have one or more records
- For a data component, a record may span many control intervals.

VSAM internals
- CONTROL INTERVAL (CI)
- VSAM stores Data and Index in Control Intervals (CI)
- CI is similar to 'Block'

Control Interval
The fundamental building block of every component of a VSAM data set is the Control Interval (CI). It is the unit of data VSAM transfers between virtual and disk storage. Its concept is similar to that of blocking for non-VSAM files. The size of the control interval affects the performance of a VSAM file. The size of a CI must be between 512 and 32,768 bytes, i.e. 32K. Up to 8K, the CI size must be a multiple of 512 bytes; beyond that it a multiple of 2K.

- CI contains
- Records (or DATA)
- Free space (Optional)
- Control Interval Definition Field (CIDF)
- Record Definition field (RDF)
- CIDF & RDF are VSAM control functions
- Used by VSAM to access data

RDF and CIDF

- RDF
- 3 bytes long
- Indicates length of records
- How many adjacent records are of the same length

CIDF

It is contained in the rightmost four bytes of each control interval, consists of two binary fields, each of two bytes in length. Right most two bytes contain the amount of unused space and the left most two bytes identify the size of the data area, i.e. the area occupied by logical records. CIDF contains a flag that is set when the control interval is being updated.

- 4 bytes long
- One per CI
- Indicates Free space

Control Area (CA)

- CIs are grouped into CA
- Can have more than one CA in a VSAM data set
- CA is VSAMs internal unit for allocating space
- Smallest is a TRACK, and the largest is a CYLINDER
- VSAM determines CA size & number of CIs in CA based on
- CI size specified
  Record size

Spanned Records

When a record size is larger than CI size (minus 7 bytes), the record must be contained in more than one CI. Such a record is called spanned record. A spanned record may span two or more control intervals, however, it may not span more than one control area. Spanned records are possible only in ESDS and KSDS. They are not available in RRDS.

Index

- Separate entity
- Organized similar to Data component (CI & CA)
- Has different CI size
- VSAM builds Index when data is loaded
- Index is organised as Inverted Binary Tree
- VSAM compresses keys to conserve space
- Can have several Levels of Indexes
- Lowest level of index is called ‘Sequence set’
- One sequence set for one Control Area
- Are always in the key order whereas data CI need not be in key order
- Each entry has the highest key (or “one” less than the 1st key of the next CI so that the sequence set need not be updated when higher key added to the CI) in the CI and points to the RBA of the CI (Vertical Pointer)
- Next level is called index set
- Contains Primary keys and pointers to the sequence set
- More than one level of Index set based on number of Sequence set records
- Top level index always one record

CI & CA Split
In a KSDS records are stored in key sequence. When a record is added to a KSDS, it is inserted in its correct sequential location in the proper control interval, and the records that follow in the control interval are shifted.
If an insertion is made into a control interval that is already full, some of the records in it are moved into a free control interval. This is called a control interval split.

Control Area Split
When a record in a KSDS is to be inserted and all the control intervals within a control area are full, a control area split takes place before a control interval split. To do this, a new control area is allocated, about half the control intervals from the original control area are moved to the new one, and a new sequence set record is created for the new control area. Then, the control interval split is performed.

- CI split happens while
- Adding new records or extending an old record
- And not enough room in the CI to complete the operation
- CI split may trigger a CA split
- Splits generally degrades performance
- Specify freespace to reduce CI & CA splits

CI SIZE

- VSAM can select an efficient size if you ignore
- Large CI size for Batch Sequential Processing
- Small CI size for random processing

ESDS

- Similar to Sequential File
- Sequenced by the order in which data is entered/loaded
- New Records are added at the end only (chronological order)
- Supports both Fixed and Variable formats
- Contains only CLUSTER & DATA components
- Only sequential access in Batch Cobol Programs
- Random access is supported in on-line applications (CICS) using Relative Byte Address (RBA)
- Alternate Index is supported in on-line applications (CICS)
- NO primary index

RBA
Record location relative to the beginning of the file (Relative Byte Address)

RRDS

- Has only CLUSTER and DATA components
- Records are stored in numbered, fixed length slots
- Each slot is given a number 'Relative Record Number (RRN)'

- VSAM determines the number of slots by
- Size of CI
- Length of Record
- Records can be deleted physically
- Empty slots are filled up with new records without shifting existing records
- No primary Index or Alternate Index
- Supports Fixed and Variable formats
- From Ver 3 Rel 3
- RRN cannot be changed
- Each slot has RDF and RDF indicates Full/Empty

KSDS

- Has all three components of VSAM (CLUSTER, INDEX and DATA)
- Key sequenced
- Primary key should be
- Unique
- Same position in every record
- Is not split (has to be contiguous)
- Records can be deleted physically
- Primary key cannot be changed
- Allows Alternate Index
- Has all the access methods
- Sequential
- Random
- Dynamic (SKIP sequential)

JCL Requirements for VSAM Data Sets

VSAM accomplishes JCL simplification by centralizing functions such as defining, deleting, and altering file characteristics in the AMS Utility program. VSAM has much simpler JCL requirements than files of other access methods.

How to allocate existing VSAM files

- The DSNAME parameter
- The DUMMY parameter
- The DISP parameter
- The AMP Parameter

How to create VSAM files using JCL.

The DD statement for allocating existing VSAM File

```
//ddname     dd {DSNAME=data-set-name}
                    {Dummy}
[,Disp={old/shr},Normal-disp,Abnormal-disp]
[,AMP=(option,option….)]
```

DSNAME	Specifies the name of the VSAM dataset. Normally, the high level qualifier of the name identifies the owning catalog.
DUMMY	Specifies that a VSAM file should not be allocated; instead, MVS should simulate a VSAM FILE.
DISP	Specifies the file's status and, the file's normal & abnormal disposition. The valid status options are OLD for exclusive access and SHR for shared access.All of the disposition options except UNCATLG are valid for VSAM files.
AMP	Specifies one or more processing options for VSAM files.

The AMP parameter

The AMP parameter is for VSAM files & the DCB parameter is for non-VSAM files. It specifies execution time information that affects how the file is processed.

The AMORG sub parameter

AMORG indicates that the files being accessed is a VSAM file. Normally, MVS realizes that a VSAM file is being processed when it retrieves the catalog information for the file. So we need to specify AMP=AMORG when MVS doesn't search the catalog.

The AMP parameter

```
AMP=[  AMPORG]
[,BUFND=n]
[,BUFNI=n]
[,BUFSP=n]
[,OPTCD=OPTIONS]
[,RECFM=FORMAT]
[,STRNO=N]
```

AMORG	Specifies that the data set is a VSAM file. Normally not required.
BUFND	Specifies the number of buffers to allocate for the data component.
BUFNI	Specifies the number of buffers to allocate for the index component.
BUFSP	Specifies the total amount of space in bytes to allocate for the data and index buffers.
OPTCD	options for the ISAM interface.Code I , L, or IL. 'I ' means that if OPTCD=L is specified for the file in the processing program, records marked for deletion by hex FF in the first byte should be physically deleted from the file. If OPTCD=L is not specified in the program, Specify OPTCD=IL in the DD statement for the same effect.
RECFM	Specifies the format in which the ISAM program expects to process records.
STRNO	Specifies the number of concurrent requests the program may issue against the file.

Dataset type parameters

KSDS	:	INDEXED or IXD
ESDS	:	NONINDEXED or NIXD
RRDS	:	NUMBERED
LDS	:	LINEAR

Data & index components

Required when installation default names are to be overridden

```
DEFINE CLUSTER                                          -
        (NAME (XIND.NLT.CLUSTER)                        -
        CYLINDER (5  1) VOLUMES (WORK01)                -
        RECORDSIZE (120   124) KEYS (8   0) INDEXED)    -
DATA                                                    -
        (NAME(XIND.NLT.CLUSTER.DATA))                   -
INDEX                                                   -
        (NAME(XIND.NLT.CLUSTER.INDEX))
```

Ex of the DEFINE CLUSTER command

Define a key-sequenced data set (KSDS)

```
        DEFINE CLUSTER  ( NAME (MTRG.CUSTOMER.MASTER)  -
                        OWNER(MTRG)                    -
                        INDEXED                        -
                        RECORDSIZE(80 80)              -
                        KEYS(6  0)                     -
                        VOLUMES(SMS007)                -
                        UNIQUE                         -
                        SHAREOPTIONS( 2  3)            -
                        IMBED)                         -
                DATA   ( NAME(MTRG.CUSTOMER.MASTER.DATA)   -
                        CYLINDERS(50  5)               -
                    CISZ(4096)  )                      -
                INDEX ( NAME(MTRG.CUSTOMER.MASTER.INDEX))
```

Ex 2

Define an entry-sequenced data set (ESDS)

```
        DEFINE CLUSTER  ( NAME (MTRG.CUSTOMER.MASTER)   -
                        OWNER(MTRG)                    -
                        NONINDEXED                     -
                        RECORDSIZE(180 80)             -
```

```
                         VOLUMES(SMS007)                      -
                   REUSE)                          -
             DATA   ( NAME(MTRG.CUSTOMER.MASTER.DATA)   -
                         CYLINDERS(50  5) )
```

Ex 3
Define an relative-record data set (RRDS)

```
      DEFINE CLUSTER  ( NAME (MTRG.CUSTOMER.MASTER)   -
                         OWNER(MTRG)                          -
                         NUMBERED                             -
                         RECORDSIZE(180 80)                   -
                         VOLUMES(SMS007)                      -
                   UNIQUE)                                    -
             DATA   ( NAME(MTRG.CUSTOMER.MASTER.DATA)   -
                         CYLINDERS(50  5) )
```

What is AMS?

Access method services is invoked by executing a utility program called IDCAMS in a regular job step. The different functions of AMS are performed thru functional commands. A functional command can have one or more parameters that can be positional or keyword . A positional parameters is identified by its position in relation to other parameters: a keyword parameters can be used in any place in relation to other keyword parameters because it is identified by the particular keyword used.

Types of AMS commands
- Functional commands
- Modal commands

AMS command	**Function**
Functional commands	
ALTER	Changes information specified for a catalog, cluster, alternate index, or path at define time.
BLDINDEX	Builds an alternate index.
DEFINE ALTERNATE INDEX	Defines an alternate index.
LISTCAT	List information about data sets
REPRO	Copies records from one file to another. The input and output files can be VSAM or non-VSAM
PRINT	Prints the contents of a VSAM or non-VSAM file.
DEFINE CLUSTER	Defines a VSAM file, whether it's KSDS, ESDS
DEFINE MASTER CATALOG	Defines a master catalog
DEFINE PATH	Defines a path that relates an alternate index to its base cluster
DEFINE USERCATALOG	Defines a user catalog.

DELETE	Removes a catalog entry for a catalog, cluster, alternate index or path
EXPORT	Produces a transportable file.
IMPORT	Copies a previously exported file.
Modal commands	
IF	Controls the flow of command execution by testing condition codes returned by functional commands
SET	Controls the flow of command execution by altering condition codes returned by functional commands
PARM	Sets options values that effect the way AMS executes.

AMS performance parameters

CONTROL INTERVAL SIZE
Syntax : CONTROLINTERVALSIZE (bytes)
Abbr : CISZ or CISIZE

Default: Calculated by VSAM
Control Interval size (CIDF & RDF) & FREESPACE must be taken into account
Thumb Rule for CISIZE.
If the record length is <8192, multiples of 512, >=8192, multiples of 2048

FREESPACE

Ex	:	FREESPACE (CI% CA%)
Default	:	FREESPACE(0 0)

High Freespace results in more I/O & consumes larger DASD space,Very low FREESPACE results in CI splits and degrades performance.

Amount of FREESPACE depends on
- Rate of growth of records
- Expected number of records to be deleted
- Reorganization frequency
- Performance consideration

BUFFERSPACE

Ex	:	BUFFERSPACE (bytes)
Abbr	:	BUFSP
Default	:	Two data buffers plus one additional index buffer for KSDS

Used to improve Input/output performance
Can also be specific in JCL EXEC parameter

```
//DD1  DD    DSNAME=ANYVSAM1,
//              AMP=('BUFND=4,BUFNI=4,STRNO=2')
```

More on VSAM Buffers

- Strings (STRNO in AMP parameter)
- Defines number of concurrent access
- For batch processing, a string of 1 is enough
- For online processing more strings is required
- Data Buffers (BUFND in AMP parameter)
- Random Access
- Minimum of 2 (One for normal access and one for split processing) is required
- Sequential Access
- Ideal to process a track
- Allocate number of buffers based on the number of CI/TRK
- For READ add 1 more buffer and for WRITE add 2 more buffers
- Index Buffers (BUFNI in AMP parameter)
- Random Access
- Ideal to keep the index set in virtual memory
- Determine the number of levels (using LISTCAT)
- Allocate number of levels + 1 buffer
- Sequential Access
- Default of 1 index buffer is enough

RECOVERY / SPEED
Mutually Exclusive
RECOVERY allows you to recover if the job initially loading the dataset fails
SPEED is faster, but does not provide restart feature
Default: RECOVERY

SPANNED

Ex:		SPANNED/NONSPANNED
Default	:	NONSPANNED

Allows large records to span more than one Control Interval.However, the records cannot span across Control areas . RRDS does not support spanned records.

KEYRANGES

Ex	:	KEYRANGES (Low - Val; High - Val)
Default	:	None (No range assumed)

The Key Ranges correspond to VOLUMES if ORDERED clase is specified.

ORDERED

Ex	:	ORDERED / UNORDERED
Default	:	UNORDERED

Goes together with KEYRANGES clause specifies the volume to which the key values should go.

REUSE

Ex	:	REUSE/NOREUSE
Default	:	NOREUSE

REUSE specifies that the cluster can be loaded with fresh records with an implicit delete of existing records. REUSE cannot be used under following circumstances and hence not recommended. When KEYRANGES parameter is coded. When alternate indexed are defined

REPLICATE
Directs the VSAM to duplicate each index as many times as it will fit on its assigned track.Applies only to KSDS index component.
To reduce rotational delay and to make I/O faster.

Ex: Replicate / Noreplicate
Default : Noreplicate

VOLUMES Can specify different volumes for Data component,Index component

IMBED
Directs the VSAM to place the sequence set (the lowest level of index next to the data component) on the first track of the data control area and duplicate it as many times as it will fit This process will reduce rotational delay because the desired sequence set record is found faster
Default : NO IMBED
Syntax : IMBED / NO IMBED

Share option
SHAREOPTIONS (Cross-region-value Cross-system-val)

Cross-region	: Concurrent data access on a standalone system (ex: TSO&CICS accessing same data)
Cross-system	: Data access for multiple computers (Two different computers that are inter-connected)
Default	: SHAREOPTIONS (1 3)

Multiple jobs can read only if no update takes place -Complete data integrity
Multiple jobs can read and at the same time one job can update - Write, but not read integrity.

Multiple jobs can read & write simultaneously - No integrity
Same as option 3, but refreshes buffer after every read
Share options 1 & 2 are not allowed for cross-system
For cross region sharing, each batch job must have DISP=SHR
For cross system sharing DISP parameter in the JCL is immaterial

ERASE / NOERASE
Default : NOERASE
ERASE instructs VSAM to move zeroes to all the bytes once the cluster is deleted

MORE AMS COMMANDS

REPRO

- All purpose load and backup utility command
- Can be used against empty / loaded VSAM file with another VSAM file or sequential file
- Much easier to use than IEBGENER
- Can be used against all four types of VSAM datasets

```
REPRO {INDATASET (DSN)     or INFILE (DD1)}
        {OUTDATASET (DSN) or OUTFILE (DD2)}
          {SKIP    (count)}  {COUNT  (count)}
       {FROMKEY}      { FROMADDRESS}
      {FROMNUMBER}  { TOKEY}
      {TOADDRESS} {TONUMBER}
      { REUSE/NOREUSE}      {REPLACE / NOREPLACE}
```

INFILE or INDATASET	parameter is mandatory, similarly OUTFILE or OUTDATASET is mandatory All other parameters are optional
SKIP	specifies number of input records to skip before beginning to
COUNT	specifies number of output records to copy.

REUSE / NOREUSEparameter

REUSE	specifies that the output file should be reset it is reusable.
NOREUSE	specifies that a reusable output file should not be reused
NOREUSE	is the default

REPLACE parameter

Replaces the records for which primary keys are matching between input and output records. f not specified, the matching key records are untouched
If the target is ESDS the records are appended and REPLACE is inappropriate.

Repro

```
// REPRO EXEC PGM=IDCAMS
// SYSPRINT DD SYSOUT=*
// DD2 DD DSN=XIND.NLT.CLUSTER.BACKUP (+1)
//          DISP=(NEW,CATLG,CATLG), UNIT=TAPE
//          VOL=(SER=121213, LABEL=(1,SL),
//          DCB=(RECFM=FB, LRECL=80)
//DD1 DD DSN=XIND.NLT.CLUSTER, DISP=SHR
// SYSIN DD*
REPRO INFILE (DD1) OUTFILE (DD2) REUSE
/ *
```

EXPORT/IMPORT

- Used for backup and recovery
- Catalog information also is exported along with the data, unlike REPRO
- DFSMS classes are preserved
- Cluster deletion and redefinition are not necessary during the import
- Can be easily ported to other systems
- Disadvantages
- The EXPORTED file not reusable until it is imported
- Slower than REPRO

VERIFY

VERIFY FILE (<ddname>)

VERIFY DATASET (<datasetname>)

- Can be issued from TSO or from a JCL
- Verifies the catalog HURBA (High Used Relative Byte Address) field and stores the true values from the control block HURBA field.
- Should be used against cluster name only and not against data or index components
- Used to rectify some of the problems due to data corruption

LISTCAT

LISTCAT Command identify the catalog, the names of the entries to be listed, the types of entries to be listed, and the amount of information about each entry to be listed.

```
LISTCAT    [       CATALOG (name) ]
           [  ENTRIES   (name - of - entries) ]
            [LEVEL      (generic-level-names)]
           [{ NAME/HISTORY/VOLUME/ALLOCATION/ALL}]
```

Catalog(name[/password]) Specifies the name and if, required, password of the catalog from which entries are to be listed.

ENTRIES(entry-name[/pd]) Specifies the names of the entries you want to list. If omitted, all entries in the specified catalog are listed.

LEVEL(level)	Specifies one or more levels of qualifications. Any data sets whose name matches those levels are listed.
Entry –type	Specifies the type of entries you want listed. If both ENTRIES / LEVEL and entry-type are omitted, all entries of all types in the specified catalog are listed. Code one of these values: ALIAS, CLUSTER, DATA, INDEX, & PATH
NAME	Specifies that only the names and types of the specified entries are to be listed. Name is the default.
HISTORY	Specifies that the information listed by NAME, Plus the history information is to be listed.
VOLUME	Specifies that the information listed by HISTORY, plus the volume locations of the specified entries, is to be listed.
ALLOCATION	Specifies that the information listed by VOLUME, plus detailed extent information, is to be listed.
ALL	Specifies that all available catalog information for the specified entries is to be listed.

Ex

To list the catalog entry for a VSAM file named MTRG.CUSTOMER.MASTER

```
ENTRIES(MTRG.CUSTOMER.MASTER)
```

To list information for more than one file, just code several file names in a single ENTRIES parameter, like this:

```
ENTRIES (MTRG.CUSTOMER.MASTER          -
         MTRG.SUPPLIER.MASTER                -
         MTRG.ITEM.TRANS)
```

We can specify a generic entry name by replacing one or more levels of the file name with an asterisk.

```
ENTRIES(MTRG.*.MASTER)
```

All files whose names consists of three levels with MTRG as the first level and MASTER as the third level, are listed. MTRG.CUSTOMER.MASTER AND MTRG.SUPPLIER.MASTER meet these criteria, so they would be listed.

ALTER

Used to change parameters such as FREESPACE
Has no effect on existing CI & CA splits

ALTER entry-name

Ex

```
    ALTER MTRG.CUSTOMER.MASTER                -
         NEWNAME(MTRG.CUSTMAST)
     ALTER EMPLOYEE.KSDS.CLUSTER
      FREESPACE(25,25)
```

Change the filename of MTRG.CUSTOMER.MASTER to MTRG.CUSTMAST

```
ALTER MTRG.CUSTOMER.MASTER.DATA                -
        ADDVOLUMES(VOL291 VOL292)              -
        REMOVEVOLUMES(VOL 281  VOL 282)
```

Add VOL291 and VOL292 to list of eligible volumes for MTRG.CUSTOMER.MASTER.DATA and remove VOL281 and VOL282.

DELETE

DELETE command is used to remove entries from a VSAM catalog. To delete more than one file, list the names in parentheses.

DELETE <object name> (parameters)

Ex: DELETE XIND.NLT.CLUSTER

All the parameters are optional.Deletes all subordinate objects such as AIX, Path

Some Common DELETE Parameters

ERASE / NO ERASE	ERASE writes binary zeroes after deletion
PURGE / NO PURGE	PURGE allows deletion even though expiration date is still Due.
ALTERNATE INDEX or AIX	Deletes only Alternate index of the cluster
PATH	Requests only path name to be deleted.
FORCE / NO FORCE	FORCE deletes the dataset even if it is not empty.

Ex 1

We can specify a generic name in a DELETE command by replacing ONE level of the entry name with an asterisk, like this

DELETE MTRG.CUSTOMER.*

Ex 2

To delete MTRG.CUSTOR.MASTER, whether or not it is expired.

DELETE MTRG.CUSTOMER.MASTER PURGE

Ex 3

To delete the 3 named files.

```
DELETE ( MTRG.CUSTOMER.MASTER MTRG.CUSTMAST.AIX
     MTRG.CUSTMAST.PATH)
```

PRINT

The PRINT command of Access Method Services is used to print the contents of both VSAM and non-VSAM data sets. The command syntax is similar to that of REPRO.

While REPRO copies an input data set into another output data set, PRINT dumps an input data set to a printer. This command is versatile and can be used to print a complete data set or only a selected part of it.

The PRINT command

```
PRINT { INDATASET(entry-name [/ password])}
            [{CHARACTER/HEX/DUMP}]
            [{SKIP(count)/
FROMKEY(key)/FROMNUMBER(number)/FROMADDRESS(address)}]
[{COUNT(count)/TOKEY(key)/TONUMBER(number)/TOADDRESS(address)]
```

CHARACTER/HEX/DUMP Specifies the format of the output. CHARACTER & HEX print the data in character or hex format.
DUMP prints data in both character and hex format. DUMP is default.

Ex

To print records 29,30 & 31 in character format.

```
PRINT INDATASET(MTRG.CUSTOMER.MASTER)        -
         CHARACTER SKIP(28)  COUNT(3)
```

To print records 29,30 & 31 in dump format.

```
PRINT INDATASET (MTRG.CUSTOMER.MASTER)       -
         DUMP  SKIP(28)   COUNT(3)
```

ALTERNATE INDEXES

What is Alternate Indexes?

An alternate index is used to a access the records of a VSAM key-sequenced data set in an order other than the file’s primary key(or base key). The data set over which an alternate index exists is called a base cluster. Even we can use an entry-sequenced data set as the base cluster for an alternate index, most alternate indexes are built over KSDS clusters.Used whenever the data is required to be retrieved on the basis of other field (than primary key field)

- Can be defined for both KSDS & ESDS
- Reduce data redundancy
- Can have duplicates
- Easy to define using IDCAMS
- Allow datasets to be accessed sequentially or randomly
- Can be updated automatically

AIX – Disadvantages

Performance degradation
Complex update logic

Alternate index organization

Alternate Indexes are KSDS by themselves and have their own data and index components.AIX data components contain the alternate index key values and pointers to each record containing the key value. AIX index components contain highest key value in an AIX data CI and a pointer to that CI AIX data component has variable length records if duplicate is allowed.

Steps for Creating Alternate Index

- Define AIX using IDCAMS DEFINE AIX
- Specify Alternate Index Path using IDCAMS DEFINE PATH
- Build AIX & populate it using IDCAMS BLDINDEX

Ex:

```
Define AIX                                      -
(NAME (XIND.NLT.VSAM DEPT. AIX)                 -
VOLUMES (WORK01)                                -
RELATE (XIND.NLT.VSAM)                          -
UPGRADE                                         -
CYLINDERS (5, 1)                                -
KEYS (8 10)                                     -
RECORDSIZE (124  124)                           -
FREE SPACE (20  10)                             -
SHARE OPTIONS(1  3)                             -
NON UNIQUEKEY)                                  -
INDEX                                           -
 (NAME (XIND.NLT.VSAM.DEPT.AIX.INDEX))          -
DATA                                            -
 (NAME(XIND.NLT.VSAM.DEPT.AIX.INDEX))
```

DEFINE ALTERNATE INDEX or
DEFINE AIX REQUIREMENT PARAMETER

Name

NAME required parameter. Advisable that name conveys AIX purpose

Volumes

Volumes (Vol-ser-1 ..... vol-ser-n) required parameter.
Assigning base cluster & AIX in different volumes improves performance

RELATE
RELATE (base cluster name) required parameter.
Establishes relationship between the base cluster & AIX

UPGRADE/NOUPGRADE
UPGRADE/NOUPGRADE.
UPGRADE specifies that records in AIX are to be updated automatically whenever the base cluster is updated. Default: UPGRADE

KEYS
KEYS (length offset)
Optional: primary key values are taken as default, if not specified
Defeats the purpose of Alternate Index, if not specified

RECORD SIZE
RECORD SIZE (average maximum)
Default: RECORDSIZE (4086 32600)
Abbr: RECSZ
Average & max. are same for UNIQUEKEY AIX and may be different for NONUNIQUEKEY

RECORD SIZE - How to calculate?
AIX for a KSDS RECSZ = 5 + AIXKL + (n x BCKL)
AIX for ESDS RECSZ = 5 + AIXKL + (n x 4)
Where: AIXKL is the alternate-key length
BCKL is the base cluster's prime-key length
n = 1 when UNIQUEKEY is specified
n = the number of data records in the base cluster that contain the same alternate-key value, when NONUNIQUEKEY is specified.

Alternate Index for ESDS
Same syntax & parameters as KSDS
Not supported in Batch COBOL (OS/VS & VSII)
Used under CICS environment

Step 2: Specifying the Alternate Index Path

Ex:
```
DEFINE PATH                                            -
          (NAME(XIND.NLT.VSAM.DEPT.PATH)          -
                 PATHENTRY                             -
             (XIND.NLT.VSAM.DEPT.AIX) UPDATE)
```

Path is a VSAM object though it doesn’t contain any records
Same command for KSDS & ESDS

Path is used to link JCL DSN to VSAM AIX (specifies that the given AIX is to be used).

Ex:

```
DEFINE PATH                                        -
            (NAME(XIND.NLT.VSAM.DEPT.PATH)            -
                     PATHENTRY                     -
               (XIND.NLT.VSAM.DEPT.AIX) UPDATE)
```

Path is a VSAM object though it doesn't contain any records
Same command for KSDS & ESDS
Path is used to link JCL DSN to VSAM AIX (specifies that the given AIX is to be used).

NAME

NAME (pathname). Pathname becomes the DSN in the run JCL PATHENTRY

PATHENTRY (entry name/password). Entry name: name assigned to alternate index cluster. Required for an alternate index

Step 3: Building the Index
BLDINDEX Command actually builds the index and populates it with records

```
//BLDINDX   EXEC PGM=IDCAMS
//SYSPRINT          DD SYSOUT = *
//DD1        DD DSN=XIND.NLT.VSAM,DISP=OLD
//DD2        DD DSN=XIND.NLT.VSAM.DEPT.AIX,DISP=OLD
//IDCVT1     DD  DSN=XIND.NLT.WRKFILE1, DISP=OLD
//IDCVT2     DD  DSN=XIND.NLT.WRKFILE2, DISP=OLD
//SYSIN      DD *
   BLDINDEX       INFILE (DD1) OUTFILE (DD2) -
         INTERNALSORT
/*
```

Processing VSAM Files
Virtual Storage Access Method (VSAM) is an access method for files on direct access Storage devices. The basic ways to use VSAM are:

- To load a file
- To retrieve records from a file
- To update a file.

VSAM processing has these advantages over QSAM:

- Data protection against unauthorized access.
- Cross-system compatibility.

- Device independence (no need to be concerned with block size and other control information).
- JCL for COBOL programs using VSAM files is simpler (information needed by the system is provided in VSAM or ICF catalogs).

VSAM processing is the only way for your COBOL for MVS & VM program to use indexed or relative file organizations.
This chapter provides a brief introduction on VSAM data set organization and access modes, describes the coding your COBOL programs need to identify and process VSAM data sets, and explains how VSAM data sets must be defined and identified to the operating system before your program can process them.
If you have complex requirements or are going to be a frequent user of VSAM, review the VSAM publications for your operating system.

VSAM Terminology

The term file in the following discussion refers to either a COBOL file or to a VSAM data set.

VSAM Dataset	**COBOL file**
Entry-sequenced data set (ESDS)	Sequential file
Key-sequenced data set (KSDS)	Indexed file
Fixed- and variable-length Relative-Record Data Set (RRDS)	Relative file

VSAM Term	**Similar Non-VSAM Term**
ESDS	QSAM data set
KSDS	ISAM data set
RRDS	BDAM data set
Control interval size (CISZ)	Block size
Buffers (BUFNI/BUFND)	BUFNO
Access Method Control Block (ACB)	Data control block (DCB)
Cluster (CL)	Data set
Cluster Definition	Data set allocation
AMP parameter of JCL DD statement	DCB parameter of JCL DD statement
Record size	Record length

IPL – Initial Program Load

What is IPL?

MVS initialization or booting the mainframe is referred to as "IPL'ing". Loading a copy of the O/S from disk to central storage & executing it is called IPLing.

IPL FLOW

- HARDWARE IPL
- IPL RIMS
- NIP RIM
- MSI
- JES

IPL Process

- The disk in which the copy is kept is generally referred as the “IPLable” disk and more specifically the SYSRES volume (System Resident 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.
- IEAIPL00 clears all online real storage (except the part it occupies itself!) to hex zeroes, it maps virtual storage required for the Master Scheduler address space to real storage locations.
- IEAIPL00 invokes a series of programs called "IPL Resource Initialization Modules" (IRIMs), which it loads from SYS1.NUCLEUS.

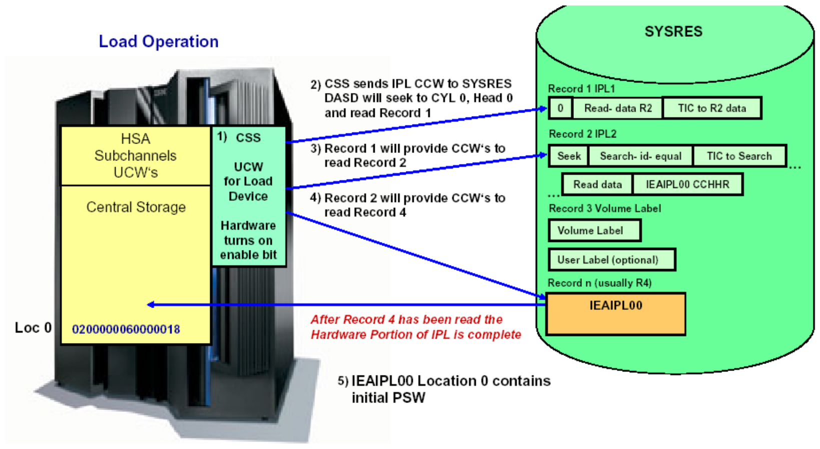
Hardware IPL

Process is defined by the z/Architecture Controlled by hardware. A single CPU is used for IPL -all other CPUs are placed into a manual (i.e. stopped) state.A hardware system reset occurs before the process begins. IPL records are written with ICKDSF Cyl 0, Trk 0, R1, R2, IEAIPL00.

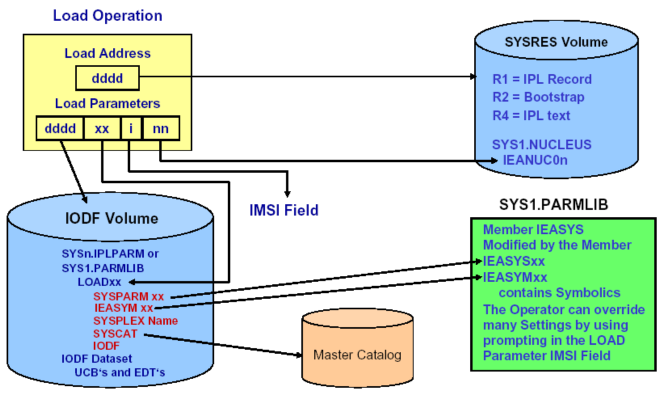
IPL Resource Initialization

Originally just loaded the Nucleus and set up the Master address space environment, Processing has gotten more complex with the XA architecture and Dynamic I/O support, Processing is single threaded. The IPL vector table (IVT) contains global information during this phase.

Hardware generates an IPL read of 24 bytes into location 0. For DASD, this always reads cylinder 0, track 0, record 1. Location 8 treated as a command chained CCW. Read record 2 into storage, command chain to next CCW


Load Operation
SYSRES
HSA Subchannels UCW's
Central Storage
Loc 0
0200000060000018
1) CSS
UCW for Load Device
Hardware turns on enable bit
2) CSS sends IPL CCW to SYSRES DASD will seek to CYL 0, Head 0 and read Record 1
3) Record 1 will provide CCW's to read Record 2
4) Record 2 will provide CCW's to read Record 4
After Record 4 has been read the Hardware Portion of IPL is complete
5) IEAIPL00 Location 0 contains initial PSW
Record 1 IPL1
0
Read- data R2
TIC to R2 data
Record 2 IPL2
Seek
Search- id- equal
TIC to Search
Read data
IEAIPL00 CCHHR
Record 3 Volume Label
Volume Label
User Label (optional)
Record n (usually R4)
IEAIPL00

Transfer CCW execution to record 2 location. Seek and search for IEAIPL00 record. Read IEAIPL00 into location 0.CCW chain completion, PSW is loaded from absolute 0 and execution begun. IEAIPL00 location 0 contains initial PSW



IEAIPL00

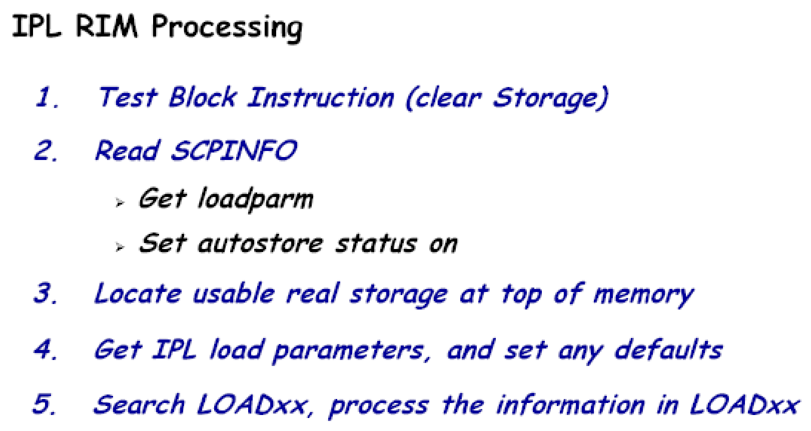
A mini operating system -non relocatable Builds an initial virtual environment, IPL workspace located at X'20000000' virtual memory.

Provides services to

- Back virtual storage with real frames
- Do I/O

Controls the IPL initialization process

- Loads IPL Resource Initialization Modules (RIMs) into workspace
- Gives them control



IPL RIM Processing

1. Test Block Instruction (clear Storage)
2. Read SCPINFO
   - Get loadparm
   - Set autostore status on
3. Locate usable real storage at top of memory
4. Get IPL load parameters, and set any defaults
5. Search LOADxx, process the information in LOADxx

IEA371I SYS0.IPLPARM ON DEVICE 5411 SELECTED FOR IPL PARAMETERS first Message displayed on NIP Console

IEA246I LOAD ID 00 SELECTED

6. Search IODF, process the information in the IODF

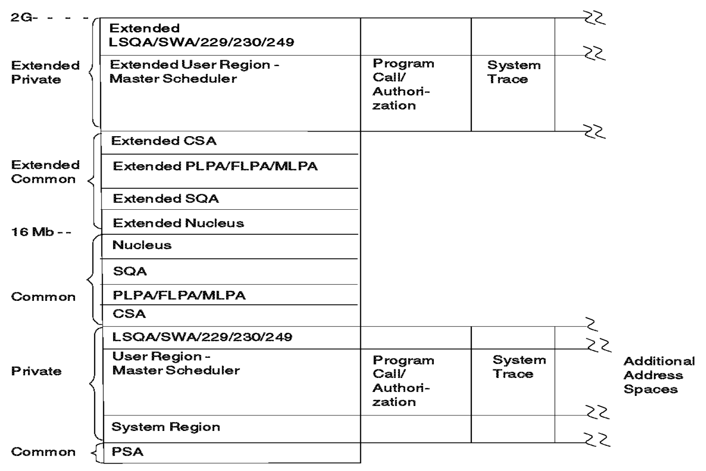
IEA246I NUCLST ID 00 SELECTED

IEA519I IODF DSN = SYSIOD.IODF24

IEA520I CONFIGURATION ID = SM15DPRI. IODF DEVICE NUMBER = 5411

Build a table of NIP consoles

Max. number of NIP consoles supported by IPL RIM is 64 (HCD supports 128)



Some Additional Address Spaces include:

- Global Resource Serialization
- Dumping Services
- Communications Task
- Allocation
- System Management Facilities(SMF)
- JES2 or JES3
- JES3AUX
- VTAM
- TCAM
- Input/Output Supervisor (IOS)
- Library Lookaside (LLA)
- User batch job or TSO/E logon session

6. *process the information in the IODF (cont.)*

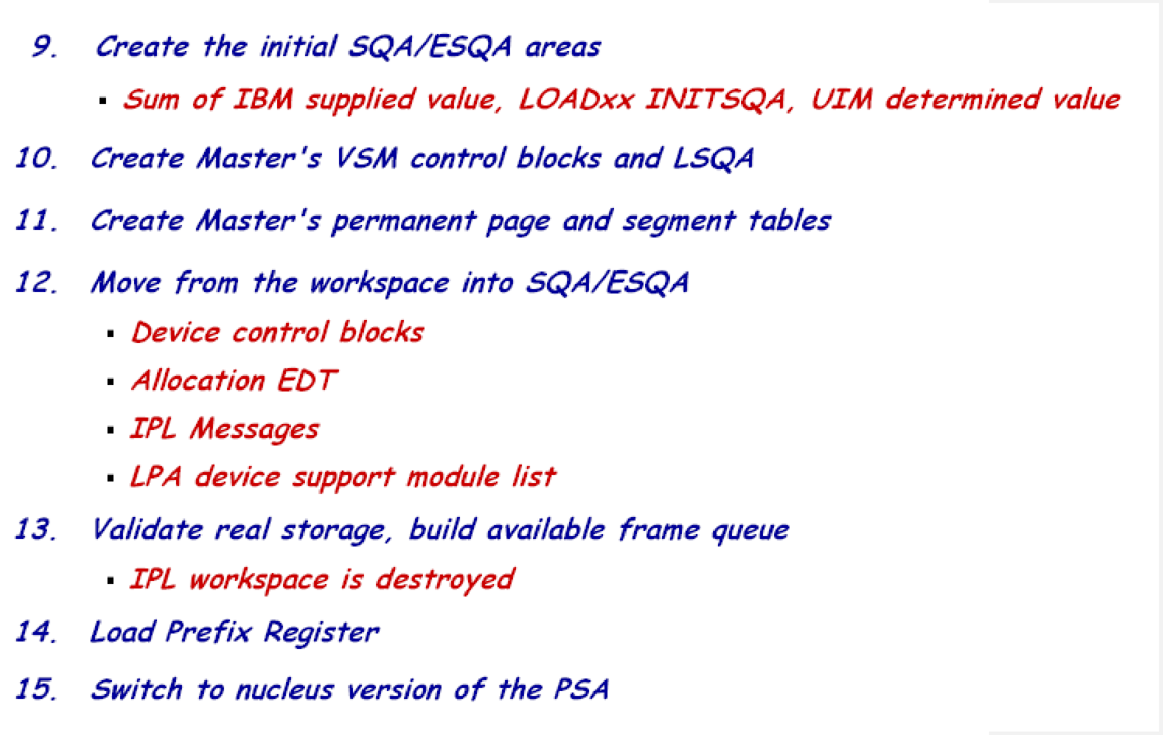
- *Invoke the device UIMs to*
  - *Identify device specific nucleus and LPA modules*
  - *Calculate required SQA and ESQA*
  - *Build device control blocks in the workspace*
  - *Build the Allocation EDT in the workspace*

7. *Create a map of the DAT-on nucleus CSECTs*

```
IEA091I NUCLEUS 1 SELECTED
IEA093I MODULE IEANUC01 CONTAINS UNRESOLVED WEAK EXTERNAL REFERENCE
IFFIOM
IEA093I MODULE IEANUC01 CONTAINS UNRESOLVED WEAK EXTERNAL REFERENCE
IEDQATTN
IEA093I MODULE IEANUC01 CONTAINS UNRESOLVED WEAK EXTERNAL REFERENCE
IECTATEN
```

- *Includes modules identified by NMLs, NUCLSTxx, and UIMs*
- *CSECTs are grouped/positioned by attributes, RMODE and read-only*

8. *Load modules, dynamically resolving external references*



9. Create the initial SQA/ESQA areas
   - Sum of IBM supplied value, LOADxx INITSQA, UIM determined value
10. Create Master's VSM control blocks and LSQA
11. Create Master's permanent page and segment tables
12. Move from the workspace into SQA/ESQA
    - Device control blocks
    - Allocation EDT
    - IPL Messages
    - LPA device support module list
13. Validate real storage, build available frame queue
    - IPL workspace is destroyed
14. Load Prefix Register
15. Switch to nucleus version of the PSA

DDDD *Device number of the volume containing the IODF dataset(Default is SYSRES).*
XX *ID of the LOADxx member to be used (the default is LOAD00).*

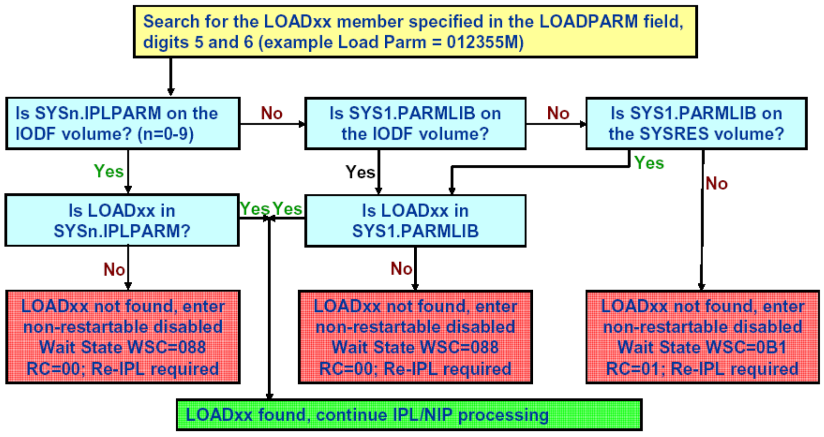
Initial Message Suppression Indicator (IMSI): *The default suppresses most informational messages and does not prompt for system parameters, will use the LOADxx values.*

NN : *Nucleus ID to be used (default is 1: IEANUC01)*

LOADxx Search Sequence

NIP Resource Initialization

Initializes basic system resources. Processing is multithreaded -normal dispatching of work is done. Basic system service (SRBs, WAIT, POST, EXCP, ATTACH, etc.) are initially available.Additional services enabled as NIP RIMs run. The NIP vector table (NVT) contains global information during this phase.


Search for the LOADxx member specified in the LOADPARM field, digits 5 and 6 (example Load Parm = 012355M)
Is SYSn.IPLPARM on the IODF volume? (n=0-9)
No
Is SYS1.PARMLIB on the IODF volume?
No
Is SYS1.PARMLIB on the SYSRES volume?
Yes
Yes
Yes
No
Is LOADxx in SYSn.IPLPARM?
Yes
Yes
Is LOADxx in SYS1.PARMLIB
No
No
LOADxx not found, enter non-restartable disabled Wait State WSC=088 RC=00; Re-IPL required
LOADxx not found, enter non-restartable disabled Wait State WSC=088 RC=00; Re-IPL required
LOADxx not found, enter non-restartable disabled Wait State WSC=0B1 RC=01; Re-IPL required
LOADxx found, continue IPL/NIP processing

Control routine

- Sets traps for unexpected errors (no RTM support is available yet)
- Verifies the hardware environment
- Creates IPL processor control blocks
- Creates global VSM control blocks
- Creates I/O control block pools
- Creates the initial system trace table
- Opens SYS1.NUCLEUS as the LNKLST
- Loads and invokes NIP RIM routines

UCW to UCB Mapping

During device mapping:

- E*ach matching UCW is enabled • each matching UCB is connected* In order for MVS to use a device
- A *UCW for the device must exist • a UCB for the device must exist.*

During the mapping process, the I/O configuration (UCWs) loaded into the HSA with a POR (or updated via dynamic I/O) is matched with the operating system configuration (UCBs) defined in the IODF. The UCWs are placed in the *disabled* state after POR or system reset.

Initial UCB state:

The UCBs are built with the “not connected“ state bit = 1 (UCB byte 7, bit 2)

At the completion of this mapping process all devices defined to both the channel subsystem (UCWs) and MVS (UCBs) will be enabled and connected.

Any UCWs without corresponding UCBs will be left disabled

Any UCBs without corresponding UCWs will be left not connected

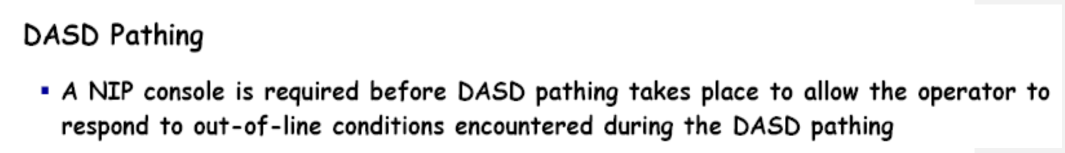
Devices in either one of these states cannot be used by the system

Non-DASD Pathing

The process of determining path availability is referred to as Pathing.

During this process MVS will check all paths for devices ginned to come up online by attempting to complete an I/O operation down each path defined to a device.

If at least one path is operational the device will be online. Tapes are an exception: pathing is performed to offline tape devises MVS does not report any paths or devices that are found to be not operational during pathing.



DASD Pathing

- A NIP console is required before DASD pathing takes place to allow the operator to respond to out-of-line conditions encountered during the DASD pathing

DASD pathing consists of 4 different phases:

Path testing on each path (P)

Read device characteristics (D)

Self-describing product (S)

VOLSER processing (V)

Any error consitions detected during the DASD pathing steps are reported to the NIP console via messages IGGN504A, IGGN505A, IEC334I, IOS291I, IEA213A or IEA214A (*any A or action messages requires operator response*)

CCW = Channel Command Word

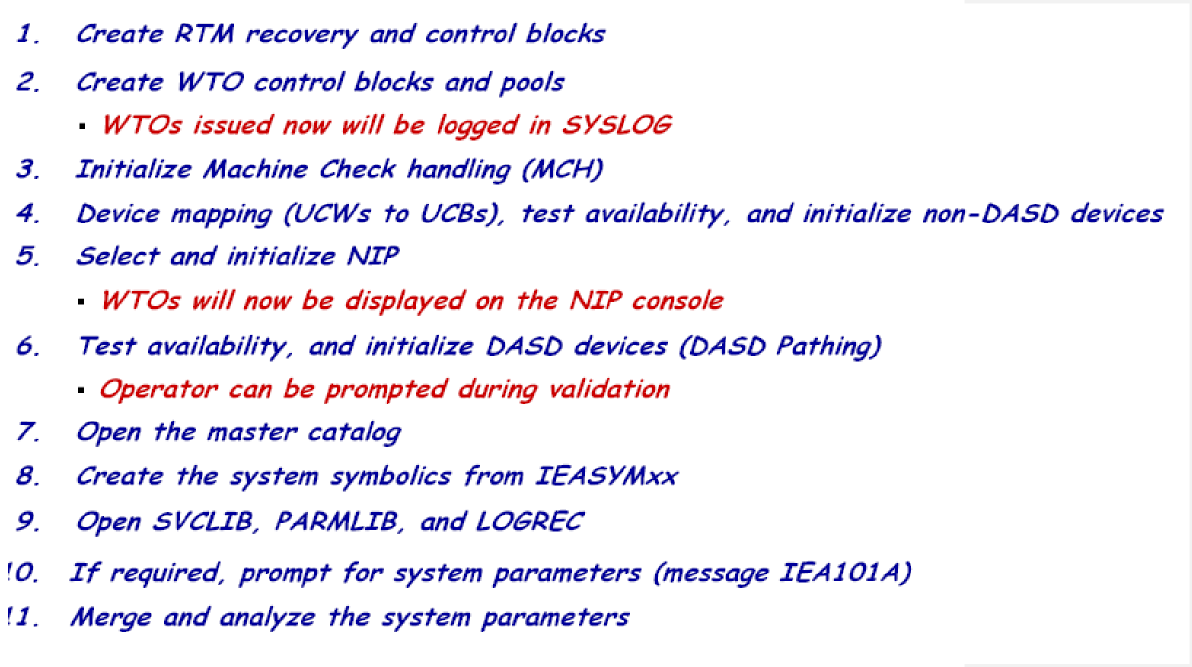
RCD = Read Configuration Data

RDC = Read Device Characteristics

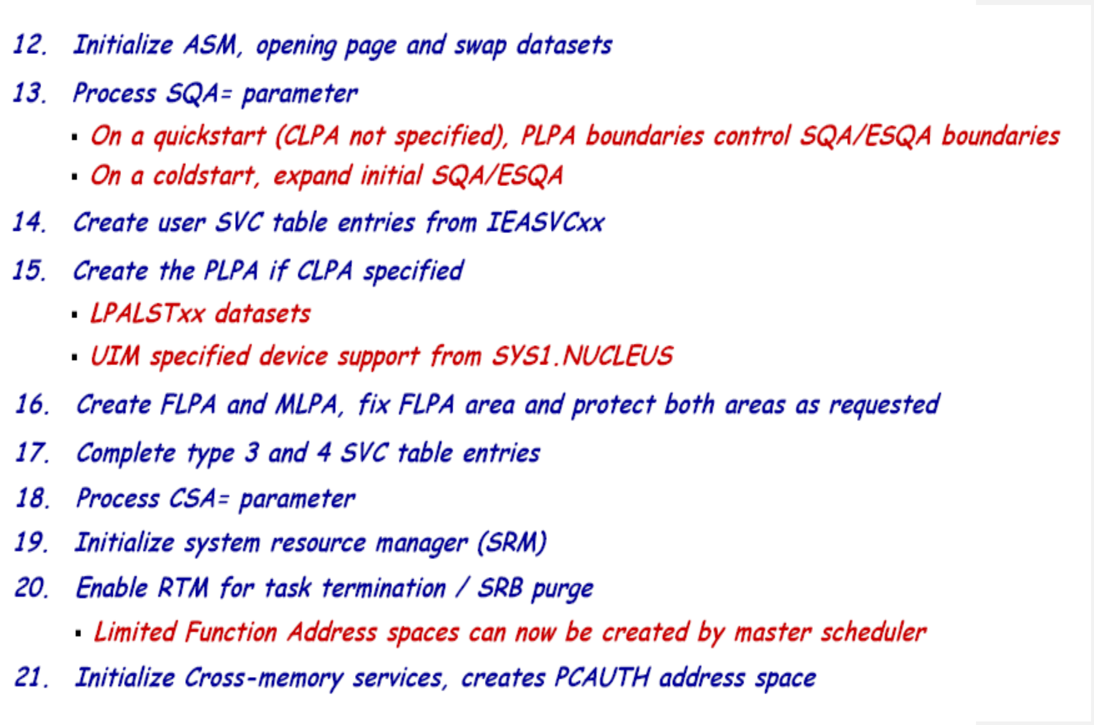
SDP = Self-describing Product

SSID = Subsystem ID (DASD CUs)

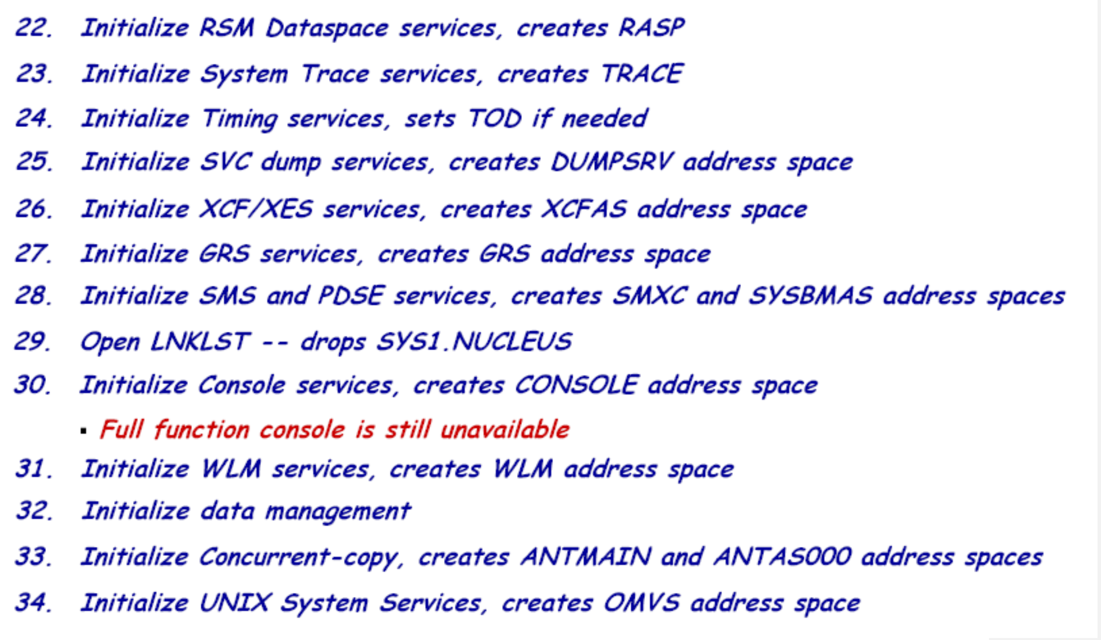
NIP RIM Processing



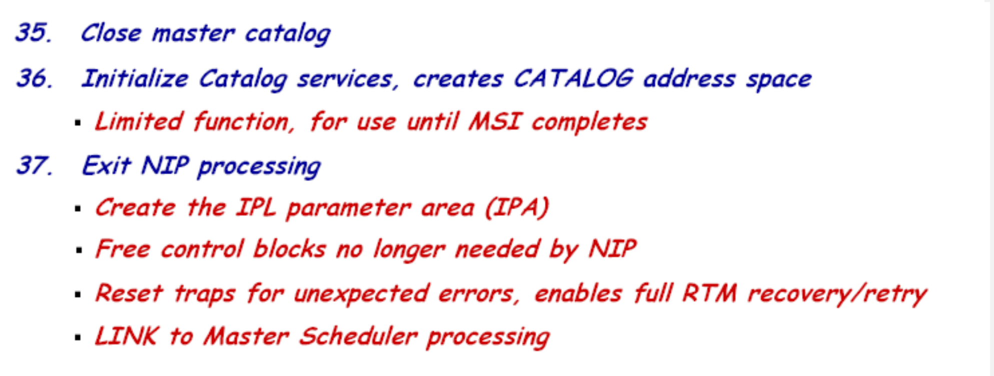
1. Create RTM recovery and control blocks
2. Create WTO control blocks and pools
   - WTOs issued now will be logged in SYSLOG
3. Initialize Machine Check handling (MCH)
4. Device mapping (UCWs to UCBs), test availability, and initialize non-DASD devices
5. Select and initialize NIP
   - WTOs will now be displayed on the NIP console
6. Test availability, and initialize DASD devices (DASD Pathing)
   - Operator can be prompted during validation
7. Open the master catalog
8. Create the system symbolics from IEASYMxx
9. Open SVCLIB, PARMLIB, and LOGREC
10. If required, prompt for system parameters (message IEA101A)
11. Merge and analyze the system parameters



12. Initialize ASM, opening page and swap datasets
13. Process SQA= parameter
    - On a quickstart (CLPA not specified), PLPA boundaries control SQA/ESQA boundaries
    - On a coldstart, expand initial SQA/ESQA
14. Create user SVC table entries from IEASVCxx
15. Create the PLPA if CLPA specified
    - LPALSTxx datasets
    - UIM specified device support from SYS1.NUCLEUS
16. Create FLPA and MLPA, fix FLPA area and protect both areas as requested
17. Complete type 3 and 4 SVC table entries
18. Process CSA= parameter
19. Initialize system resource manager (SRM)
20. Enable RTM for task termination / SRB purge
    - Limited Function Address spaces can now be created by master scheduler
21. Initialize Cross-memory services, creates PCAUTH address space



22. Initialize RSM Dataspace services, creates RASP
23. Initialize System Trace services, creates TRACE
24. Initialize Timing services, sets TOD if needed
25. Initialize SVC dump services, creates DUMPSRV address space
26. Initialize XCF/XES services, creates XCFAS address space
27. Initialize GRS services, creates GRS address space
28. Initialize SMS and PDSE services, creates SMXC and SYSBMAS address spaces
29. Open LNKLST -- drops SYS1.NUCLEUS
30. Initialize Console services, creates CONSOLE address space
    - Full function console is still unavailable
31. Initialize WLM services, creates WLM address space
32. Initialize data management
33. Initialize Concurrent-copy, creates ANTMAIN and ANTAS000 address spaces
34. Initialize UNIX System Services, creates OMVS address space

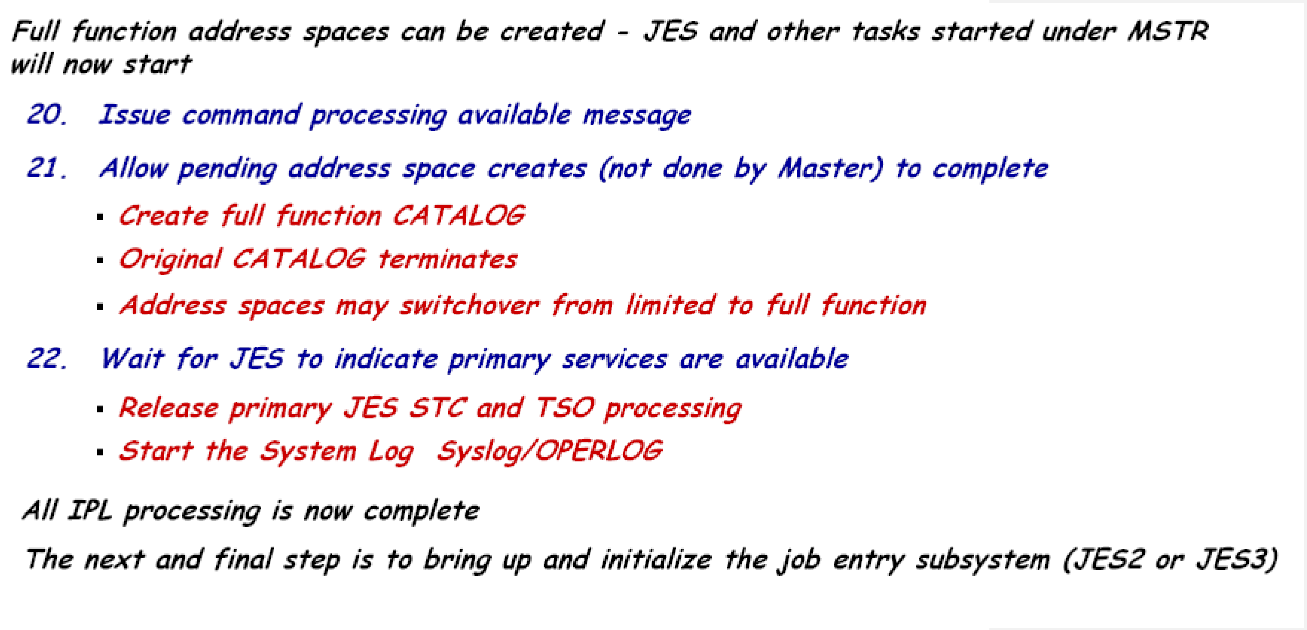


35. Close master catalog
36. Initialize Catalog services, creates CATALOG address space
    - Limited function, for use until MSI completes
37. Exit NIP processing
    - Create the IPL parameter area (IPA)
    - Free control blocks no longer needed by NIP
    - Reset traps for unexpected errors, enables full RTM recovery/retry
    - LINK to Master Scheduler processing

Master Scheduler Initialization(MSI)

- Completes initialization of system functions
- Coordinates final completion with primary subsystem (JES2/JES3)
- Basic Processing : Initialize Master Trace processing
- Enable full function Console processing All MCS consoles are now available
- Initialize Sysplex-wide ENF services, creates IEFSCHAS address space
- Initialize MSTR subsystem
- Initialize Common JES services, creates JESXCF address space
- Initialize Allocation services, creates ALLOCAS address space
- Attach Initiator to start Master JCL

1. Initialize MIH services
2. Complete ASM initialization
3. Initialize IOS dynamic pathing, create IOSAS
4. Initialize Master's security environment
5. Initialize Console attributes, DEL=RD etc.
6. Initialize APPC services
7. Initialize TSO services
8. Initialize LOGREC Logstream recording
9. Enable ENF services
10. Initialize System Logger services, creates IXGLOG address space
11. Vary all available CPs online
    - we are now multiprocessing
12. Initialize SMF services, creates SMF address space
13. Issue commands in IEACMD00 and COMMNDxx parmlib members
    - only commands processed by CONSOLE will execute now
14. Initialize RTM services
    - LOGREC recording
    - Address space termination
    - SVC dump processing
15. Initialize System security processing
16. Build defined subsystems
    - Invoke initialization routine
    - Issue START for primary JES subsystem, if requested
17. Hold primary JES STC and TSO processing
18. Indicate MSI is complete
19. Initialize Master command processing
    - Any pending commands that execute in Master will now be executed
    - Start commands are executed by Master



Full function address spaces can be created - JES and other tasks started under MSTR will now start

20. *Issue command processing available message*
21. *Allow pending address space creates (not done by Master) to complete*
    - *Create full function CATALOG*
    - *Original CATALOG terminates*
    - *Address spaces may switchover from limited to full function*
22. *Wait for JES to indicate primary services are available*
    - *Release primary JES STC and TSO processing*
    - *Start the System Log Syslog/OPERLOG*

All IPL processing is now complete

The next and final step is to bring up and initialize the job entry subsystem (JES2 or JES3)

Types of IPL:

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

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

3. Warm Start: Any IPL that does not reload the PLPA, but does preserve journaled VIO data set pages. **Don't specify CLPA or CVIO option**

<u>Overview of parmlib members</u>

PARMLIB – Parameter Library

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.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.

```
D PARMLIB
```

Three of the most important parmlib members are IEASYSxx , IEASYMxx & LOADxx.

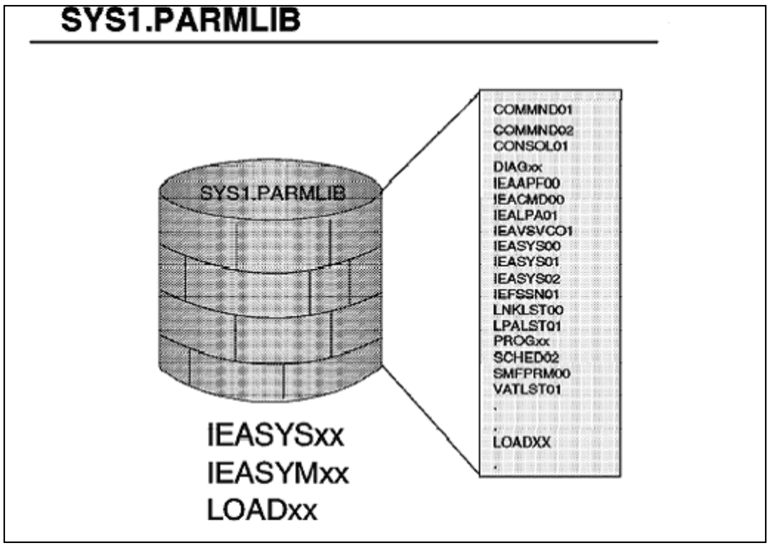
IEASYSxx

This parmlib member allows the specification of system parameters.The available options for specifying the system parameters are

1. **Respond to the IEA101A message during IPL or** MESSAGE IEA101A SPECIFY SYSTEM PARAMETERS message

This message is issued during system initialization to allow the operator to change certain system parameters.

```
REPLY 00,CLPA,SYSP=01,LNK=(04,05,PQ),SYSNAME=AQ
```



This is the most important data set in an operating system.

2. **Specify it through the SYSPARM statement of the LOADxx parmlib member or**

3. **System will take the default member IEASYS00.**

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.

IEASYSxx (System Parameter List):

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

LOADxx (System Configuration Data Sets):

The LOADxx member specifies:

- Information about your I/O configuration.(IODF statement)
- An alternate nucleus ID.(NUCLEUS statement)
- The NUCLSTxx member that you use to add and delete modules from the nucleus region at IPL-time.(NUCLST statement)
- Information about the master catalog.(SYSCAT statement)
- 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.(IEASYM,SYSPARM stet.)
- 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.

To display the parmlibs defined but not found, enter:

```
D PARMLIB,ERRORS
```

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. You can place the LOADxx member in one of the following system data sets: SYSn.IPLPARM, SYS1.PARMLIB
Consider placing LOADxx in the SYSn.IPLPARM data set. During IPL, the system looks for LOADxx in the following order:

1. SYS0.IPLPARM through SYS9.IPLPARM on the IODF volume
   SYS1.PARMLIB on the IODF volume
   SYS1.PARMLIB on the sysres volume.

LOADxx is a column-dependent parmlib member. An asterisk in column 1 denotes a comment. Parameters begin in column 1. Data begins in column 10. Columns 73-80 are ignored.

IEASYMxx (Symbol Definitions) :

IEASYMxx contains statements that: Define static system symbols, Specify the IEASYSxx parmlib members that the system is to use.
Ex:

```
SYSPARM(00,01)                    /* IEASYSxx parmlib members 00 and 01 */
SYSNAME(S1MVS)                    /* Specify name for first test system */
SYSCLONE(&SYSNAME(1:2))     /* Resolves to first 2 chars in SYSNAME */
SYMDEF(&MARYJOE.='1234568')
```

&MARYJOE is a static system symbol referring to 1234568. These symbols can be used IEASYSxx parmlib members.

```
D SYMBOLS
```

IEAFIXxx (Fixed LPA List):

Use the IEAFIXxx member to specify the names of modules that are to be fixed in central storage for the duration of an IPL. (The libraries are cataloged in the system master catalog.)

When a module is requested, the program manager searches the list of fixed routines before it examines the pageable LPA directory. The price for this performance improvement is the reduction in 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.

Ex: INCLUDE LIBRARY(SYS1.LINKLIB) MODULES(IKJPARS IKJPARS2 IKJSCAN IKJEFD00 IKJDAIR)

IEALPAxx (Modified LPA List):

Use the IEALPAxx member to specify the reenterable modules that are to be added as a temporary extension to the pageable link pack area (PLPA). (The modules are cataloged in the system master catalog.) The system searches the fixed LPA before the modified LPA for a particular module and selects the module from the modified LPA only if the module is not also in the fixed LPA.

Ex: INCLUDE LIBRARY(SYS1.LINKLIB) MODULES(IKJPARS IKJPARS2 IKJSCAN IKJEFD00 IKJDAIR)

LPALSTxx member:

Use the LPALSTxx member to add your installation's read-only reenterable user programs to the pageable link pack area (PLPA). Placing programs in the PLPA allows them to be shared among users of the system. Use one or more LPALSTxx members to concatenate your installation's program library data sets to SYS1.LPALIB. The system uses this concatenation, which is referred to as the LPALST concatenation, to build the PLPA.

Ex: IEASYSxx: ..., LPA=(00)
LPALST00: SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB

PROGxx:

The PROGxx parmlib member contains the following optional statement types:

- APF, which defines the format and contents of the APF-authorized program library list.
- EXIT, which controls the use of exits and exit routines.
- LNKLST, which controls the definition and activation of a LNKLST set of data sets for the LNKLST concatenation.
- LPA, which defines the modules to be added to, or deleted from, LPA after IPL

APF statement:

Use the APF statement to specify the following:

- The format (dynamic or static) of the APF-authorized library list
- Program libraries to be added to the APF list

If you specify both the PROG=xx and the APF=xx parameters, then the system places into the APF list the libraries listed in IEAAPFxx, followed by the libraries listed in the PROGxx member or members.

Ex:
```
APF FORMAT(DYNAMIC)
APF ADD DSNAME(SYS1.SUPER.UTILS)  VOLUME(614703)
APF ADD DSNAME(SYS1.ACCTG.RECRDS)  VOLUME(******)
APF ADD  DSNAME(SYS1.COMPU.DATA)  SMS
```

```
D PROG,APF
```

EXIT Statement:

Use the EXIT statement type to specify statements that:

1. Add exit routines to an exit
2. Modify exit routines for an exit
3. Delete exit routines from an exit
4. Undefine implicitly defined exits
5. Change the attributes of an exit.

The system will process first PROGxx and then EXITxx parameters if both are specified.

Ex:
```
EXIT ADD EXITNAME(SYS.IEFUJI) MODNAME(R1)
```

```
D PROG,EXIT
```

LNKLST Statement:

A LNKLST set consists of an ordered list of data sets for processing as the LNKLST concatenation. Every LNKLST set contains the LINKLIB, MIGLIB, and CSSLIB data sets as the first data sets in the LNKLST concatenation. You can use LNKLST statements in PROGxx instead of using LNKLSTxx to define the LNKLST concatenation. At IPL, ensure that you have a LNKLST ACTIVATE statement for the LNKLST set that you have defined, and specify PROG=xx instead of LNK=xx. You can define multiple LNKLST sets, but only one LNKLST set is current in the system at any time.If you update members in LNKLST data sets, be sure to refresh LLA's directory table after completing the updates if you want to have the changes take effect immediately. You can refresh LLA in the following ways:

- To update specific entries in LLA's directory table, enter the MODIFY LLA UPDATE command
- To refresh all entries in LLA's directory table, enter the MODIFY LLA REFRESH command
- Recycle (stop and restart) LLA.

If you define a LNKLST set to be activated through PROGxx and specify both PROG=xx and LNK=xx, the system uses the definitions in PROGxx and issues message CSV487I:

LNK IPL PARAMETER HAS BEEN IGNORED. LNKLST SET lnklstname IS BEING USED

Ex LNKLST DEFINE NAME(NEWLLSET)
LNKLST ADD NAME(NEWLLSET) DSNAME(dataset.to.be.added)
LNKLST ACTIVATE NAME(NEWLLSET)

```
D PROG,LNKLST
```

LPA Statement

Use the LPA statement to specify:

- Modules that are to be added to the LPA following IPL
- Modules that are to be deleted from the LPA following IPL
- Threshold values for minimum amounts of CSA storage that must still be available after an ADD operation.

```
D PROG,LPA
```

IEFSSNxx (Subsystem Definitions)

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.

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.

```
SUBSYS     SUBNAME(subname) [CONSNAME(consname)][INITRTN(initrtn)
               [INITPARM(initparm)]] [PRIMARY({NO|YES}) [START({YES|NO})]]
```

CONSNAME(consname) The name of the console to which any messages that the SSI issues as part of initialization processing are to be routed.
INITRTN(initrtn) The name of the subsystem initialization routine.
INITPARM(initparm) Input parameters to be passed to the subsystem initialization routine.
PRIMARY({NO|YES}) Parameter indicating whether this is the primary subsystem.

The primary subsystem is typically a job entry subsystem (either JES2 or JES3).

START({YES|NO}) Parameter indicating whether an automatic START command should be issued for the primary subsystem.

Ex: SUBSYS SUBNAME(JES2) PRIMARY(YES)

IFAPRDxx (Product Enablement Policy):

Use the IFAPRDxx parmlib member to define the enablement policy for products or product features that support product enablement. The policy lists the products and features, as well as the system environment in which they are enabled to run.
The system builds the enablement policy from the PRODUCT statements and WHEN statements in the active IFAPRDxx member(s). Each WHEN statement defines a system environment. PRODUCT statements identify products and product features that are enabled or disabled when running in the system environment defined on the preceding WHEN statement. You can use the SET PROD operator command to modify the enablement policy dynamically by specifying which IFAPRDxx member(s) the system is to use. Statements in the member modify, not replace, an existing policy.

Ex:
```
PRODUCT OWNER('IBM CORP')
        NAME(OS/390)
        ID(5645-001)
        FEATURENAME(GDDM-REXX)
        STATE(ENABLED)
        WHEN ( SYSNAME(S) SYSPLEX(SP) )
```

```
D PROD
```

IGDSMSxx (Storage Management Subsystem Definition):

IGDSMSxx contains the parameters that initialize the Storage Management Subsystem (SMS) and specify the names of the active control data set (ACDS) and the communications data set (COMMDS). Defining SMS Through the IEFSSNxx Member, You can start SMS only after you define it to MVS as a valid subsystem. You do this by adding a record for the SMS subsystem to parmlib member IEFSSNxx. IEFSSNxx defines how MVS is to initialize the SMS address space.

```
SUBSYS SUBNAME(SMS) [INITRN(IGDSSIIN)[INITPARM('ID=yy,PROMPT=NO )]]
                   [ YES  ]]   [ DISPLAY ]]
```

IGDSSIIN Identifies the SMS subsystem initialization routine. If you include this field, SMS is automatically started at IPL. If you omit this field, SMS as defined to MVS as a valid subsystem, but is not automatically started at IPL.

- Use the SET SMS command to initialize SMS parameters and start SMS if it was defined, but not started at IPL time, or restart SMS if it is already active.
- Use the SETSMS command to change SMS parameters when SMS is already running.

Assume that you want to define SMS with the following properties:

- An ACDS named SYS1.ACDS9
- A COMMDS named SYS1.COMMDS
- An interval of 15 seconds before synchronizing with any other SMS subsystems in the complex
- SMF type 42 records are to be synchronized with the SMF and RMF intervals.

```
SMS ACDS(SYS1.ACDS9) COMMDS(SYS1.COMMDS) INTERVAL(15)
SMF_TIME(YES)
```

```
D SMS
```

IKJTSOxx (TSO/E Commands and Programs):

You can use IKJTSOxx to identify the commands and programs the system is to use. If RACF 1.9 or later is installed, you can use the MSGPROTECT parameter to meet more stringent security requirements.

The IKJTSOxx member allows you to identify:

- Authorized commands and programs
- Commands that a user cannot issue in the background
- APF-authorized programs that users may call through the TSO/E service facility.

IKJTSOxx also allows you to specify the defaults for the TSO/E ALLOCATE, SEND, RECEIVE, TRANSMIT, CONSOLE, and TEST commands.By defining the SHR option on the ALLOCATE parameter, you can change the system default for the disposition of data sets from OLD to SHARE (SHR). The TRANSREC statement allows you to specify the characteristics for the data to be transmitted or received. The TEST parameter allows you to specify the installation-written test subcommands and the names of TSO/E commands that are to be allowed to execute under the TEST command. These commands are in addition to those allowed by default.

Through the SEND statement, you can specify the data set into which the SEND command is to store messages. The system checks that named data set when the user issues the LISTBC command to retrieve stored messages.

If present, IKJTSO00 is used automatically during IPL. To select another IKJTSOxx member after IPL, you can issue the following TSO/E command:

```
PARMLIB  UPDATE(xx)
```

TSOKEY00 (TSO/VTAM Time-Sharing Parameters):

TSOKEY00 contains TSO/VTAM time-sharing parameters.

Starting TSO/VTAM time sharing activates the terminal control address space (TCAS). The function of TCAS is to accept TSO/VTAM logon requests and to create an address space for each TSO/E user. TCAS builds a TCAS table (TCAST) and inserts the parameter values into it. The VTAM terminal I/O coordinator (VTIOC), which is the interface between TSO/E and VTAM, uses these values to control the time-sharing buffers, the maximum number of users, and other operational variables.

Ex.USERMAX=40,RECONLIM=3,BUFRSIZE=132,HIBFREXT=48000,
LOBFREXT=24000,CHNLEN=4,SCRSIZE=480,ACBPW=password(optional-no default),MODE=NOBREAK,MODESW=NO,RCFBDUMP=xxyyz(optional-no default),CONFTXT=YES,GNAME=(optional - no default),BASENAME=TSO
USERMAX

LNKLSTxx (LNKLST Concatenation):

Use the LNKLSTxx member of parmlib to specify the program libraries that are to be concatenated to SYS1.LINKLIB to form the LNKLST concatenation. In addition to the data sets you specify in LNKLSTxx, the system automatically concatenates data sets SYS1.MIGLIB and SYS1.CSSLIB to SYS1.LINKLIB.

```
  IEASYSxx:   LNK=(00,01,02,03)
  LNKLST00:   SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB
```

```
LNKLST01:   SYS1.LINKLIB,DBLUE.U30LIB(U30PAK),SYS2.U30LIB
LNKLST02:   SYS1.AUXLIB,SYS1.JES3
LNKLST03:   SYS1.TEST
```

As a result of these specifications, the following data sets, in the order specified are concatenated to SYS1.LINKLIB (after SYS1.MIGLIB and SYS1.CSSLIB)

```
SYS1.CMDLIB,SYS1.TSORTNS,SYS1.BTAMLIB,DBLUE.U30LIB,
SYS2.U30LIB,SYS1.AUXLIB,SYS1.JES3,SYS1.TEST
```

CSVLLAxx (Library Lookaside (LLA) List):

Use the CSVLLAxx member to specify which libraries (in addition to the LNKLST concatenation) library lookaside (LLA) is to manage. If you do not supply a CSVLLAxx member, LLA will, by default, manage only the libraries that are accessed through the LNKLST concatenation.

The START LLA,LLA=xx command identifies the CSVLLAxx parmlib member to be used to build the LLA directory. This command is issued by the IBM-supplied IEACMD00 parmlib member during system initialization

COFVLFxx (Virtual Lookaside Facility Parameters):

The virtual lookaside facility (VLF) enables an authorized program to store named objects in virtual storage managed by VLF and to retrieve these objects by name on behalf of users in multiple address spaces. VLF is designed primarily to improve performance by retrieving frequently used objects from virtual storage rather than performing repetitive I/O operations from DASD. Certain IBM products or components such as LLA, TSO/E, CAS, and RACF use VLF as an alternate way to access data.

IBM supplies a default VLF parmlib member (COFVLF00) that contains CLASS statements for the VLF classes used by IBM-supplied products. You might need to tailor some of these CLASS statements to meet your installation's needs. In addition, your installation can write applications that use VLF, and you must include the CLASS statements for those applications.

There are three items VLF requires for each VLF class used. They are:

1. The name of the class, specified on the required NAME parameter. IBM supplies the names   of the classes it uses. These names start with the letters A-I.
2. The maximum amount of virtual storage that your installation wants VLF to use for the objects   in the class.
3. A list of the major names that represent eligible sources of data for objects in the VLFC class.

How you specify the major names depends on whether the VLF class is a PDS class or a non-PDS class. For a PDS class, each major name identifies a unique partitioned data set and consists of a PDS name concatenated to the volume serial number. For a PDS class, use the EDSN and VOL parameters on the CLASS statement to define the major names.

For a non-PDS class, the major name does not correspond to a partitioned data set. To specify the eligible major names for the class, use the EMAJ parameter on the CLASS statement. For an IBM-supplied class, use the product information to determine if anything other than the name(s) specified in the IBM-supplied default COFVLFxx member are eligible.

Starting VLF:

Issue the following command to start VLF:

```
START VLF,SUB=MSTR,NN=xx
```

The two alphanumeric characters (xx) are added to COFVLF to form the name of the COFVLFxx member. If you do not code NN=xx, the system defaults to COFVLF00.

COFDLFxx (Hiperbatch Parameters):

The COFDLFxx parmlib member provides the name of the DLF installation exit and sets limits on the amount of expanded storage that the system can use for Hiperbatch. To activate DLF, the CLASS statement describing the group of shared objects must be present in the active COFDLFxx parmlib member (the member named on the START command for DLF). Only one COFDLFxx member can be active at a time.

Issue the following command to start DLF:

```
START DLF,SUB=MSTR,NN=xx
```

The two alphanumeric characters (xx) are added to COFDLF to form the name of the COFDLFxx member. If you do not code NN=xx, the system defaults to COFDLF00.

```
DISPLAY DLF
```

CLOCKxx (Time of Day Parameters):

CLOCKxx performs the following functions:

- Prompts the operator to initialize the time of day (TOD) clock during system initialization
- Specifies the difference between the local time and Greenwich Mean Time (GMT)

- Controls the utilization of the IBM Sysplex Timer (9037), which is an external time reference (ETR). Having all systems in your complex attached and synchronized to a Sysplex Timer ensures accurate sequencing and serialization of events.

The CLOCKxx member for a system that is a member of a multisystem sysplex must contain a specification of ETRMODE YES. The system then uses the Sysplex Timer to synchronize itself with the other members of the sysplex. The system uses a synchronized time stamp to provide appropriate sequencing and serialization of events within the sysplex.

Displaying the Local and Greenwich Mean Time and Date

```
D T
```

CONSOLxx (Console Configuration Definition):

CONSOLxx is an installation-created member of SYS1.PARMLIB in which you can define a console configuration to meet the particular needs of your installation. In CONSOLxx, you define multiple console support (MCS) consoles. You define the specific characteristics of up to 99 MCS consoles and the system console. While the master console is the principal means of communicating with the system, the other MCS consoles often serve specialized functions and can have master authority to enter all MVS commands.

Through the CONTROL, SET, and VARY commands, you can change some of the characteristics specified in CONSOLxx:

- If the console is not named, the changes are temporary. The changes made through the CONTROL, SET, and VARY commands last only for the current IPL. At the next IPL, the definitions in the selected CONSOLxx member are in effect.
- If the console is named in the CONSOLxx member NAME parameter, the changes made through the CONTROL, SET, and VARY commands can last the life of the sysplex.

You define each console by device number on the CONSOLE statement. IBM recommends that you also associate a system symbol console name with a console.

```
CONSOLE  DEVNUM    {(devnum)  }  {(SUBSYSTEM)} {(SYSCONS)}
           UNIT      {(unittype)}  {(PRT)    } NAME  (conname)
           AUTH      {(MASTER)        }   {(INFO)          }
                     {([SYS][,IO][,CONS])} {(ALL)           }
           USE           {(FC)}  {(MS)} {(SD)}
         ALTERNATE  {(devnum) }  {(conname)}
```

CONSOLE

CONSOLE indicates the beginning of a statement that defines the characteristics of a console.

DEVNUM {(devnum) } | {(SUBSYSTEM) } | {(SYSCONS) }

DEVNUM specifies the device number of the console. DEVNUM is required and must be the first keyword on the CONSOLE statement. devnum must be the same as the number that was specified for the device on the Add Device panel in HCD.
SUBSYSTEM indicates that this console is reserved for subsystem use, such as by NetView.
SYSCONS indicates that this console is the System Console attached to this processor.

ALTERNATE {(devnum) } | {(conname)}

ALTERNATE specifies the device number or the symbolic console name of the alternate console. An alternate console is eligible to backup this console for a console failure. The console specified on the ALTERNATE parameter must be an MCS console defined on a different CONSOLE statement.

<u>IEACMD00 (IBM-Supplied Commands):</u>

IEACMD00 contains IBM-supplied commands, as follows:

- A CHNGDUMP command to add trace table, LSQA and XES information to SVC dumps
- A SET command (SET SLIP=00) to indicate that the system is to use the IEASLP00 parmlib member to issue IBM-supplied SLIP commands.
- A SET command (SET DAE=00) to indicate that the system is to use the ADYSET00 parmlib member to start DAE processing.
- A START command (START LLA, SUB=MSTR) to start the library lookaside (LLA) procedure, which resides in SYS1.PROCLIB. The LLA procedure then starts the library lookaside function.
- A START command (START BLSJPRMI, SUB=MSTR) creates IPCS tables which allows SNAP ABDUMP and IPCS to print formatted control blocks in dumps.

```
COM='CHNGDUMP SET,SDUMP=(LSQA,TRT,XESDATA),ADD'
COM='SET SLIP=00'
COM='SET DAE=00'
```

COM='START LLA,SUB=MSTR'
COM='START BLSJPRMI,SUB=MSTR'
COMMNDxx (Commands Automatically Issued at Initialization):

COMMNDxx is an optional, installation-created list of automatic commands the system internally issues as part of system initialization. COMMNDxx is useful for automatic entry of commands that are frequently issued at system initialization.

You cannot use this member to issue JES commands, because JES is not started when the system issues the COMMNDxx commands.

MSTJCLxx (Master Scheduler JCL):

The MSTJCLxx parmlib member contains the master scheduler job control language (JCL) that controls system initialization and processing. You can place the master scheduler JCL in MSTJCLxx as an alternative to keeping it in the MSTJCLxx module in SYS1.LINKLIB.

The advantage to placing the master JCL in the MSTJCLxx parmlib member is that it is easier to make changes to the master JCL. If you specify the master JCL in the MSTJCLxx module in linklib, you must reassemble the module and link-edit it into the system each time a change is made. Specifying the master JCL in parmlib eliminates the need to reassemble and link-edit the module.

NUCLSTxx (Customizing the Nucleus Region):

The NUCLSTxx member allows you to load installation-supplied modules into the system's DAT-ON nucleus region at IPL-time.

You can use NUCLSTxx to:

- Add your installation's modules to the nucleus region.
- Delete nucleus-resident modules and replace them with alternate versions of the modules.

NUCLSTxx saves you from having to link-edit your installation's nucleus-resident routines (such as installation-written SVCs) into the IEANUC0x member of SYS1.NUCLEUS.

The modules to be added to the nucleus region, or deleted from it, must reside in members of SYS1.NUCLEUS. To add or delete modules, simply specify the members on INCLUDE or EXCLUDE statements in NUCLSTxx. NUCLSTxx resides in SYS1.PARMLIB (SYSn.IPLPARM wherever LOADxx resides)

SMFPRMxx (System Management Facilities (SMF) Parameters):

The SMFPRMxx member allows you to control how system management facilities (SMF) works at your installation.
You can use SMFPRMxx parameters to:

- Identify the system on which SMF is active.
- Specify global values for interval recording and synchronization that SMF, RMF, and other requestors can use to schedule the execution of their interval functions.
- Specify the data sets to be used for SMF recording.
- Specify the system identifier to be used in all SMF records.
- Select the SMF record types and subtypes SMF is to generate.
- Allow the operator to change the SMF parameters established at IPL.
- Specify the job wait time limit.
- Specify whether SMF is to invoke installation-supplied SMF exit routines.
- Specify whether the SMF dump program is to attempt to recover from abends.
- Specify the system response when SMF has used all of the buffered storage in its Address space.
- Specify the system response when the last SMF data set is filled and no other data sets are available for use.

SCHEDxx (PPT, Master Trace Table, and Abend Codes for Automatic Restart):
Use the SCHEDxx member of parmlib to specify the following:

- Size of the master trace table
- Abend codes that are eligible for automatic restart
- Programs that are to be included in the program properties table (PPT) and thus receive special attributes.

The SCHED statements are described as follows:

```
   MT SIZE    {(nnnK)      } {(NONE)      } {(24K)       }
```

NORESTART CODES(code,code...)
RESTART CODES(code,code...)
PPT PGMNAME(xxxxxx)
{CANCEL } | {NOCANCEL} The program specified on PGMNAME can be cancelled (CANCEL) or cannot be cancelled (NOCANCEL).

KEY(n) The program specified on PGMNAME is to have the protection key (n) assigned to it. The range of values for n is 0 through 15.

{SWAP } / {NOSWAP} The program specified on PGMNAME is swappable (SWAP) or non-swappable (NOSWAP).

{PRIV } / {NOPRIV} The program specified on PGMNAME is privileged (PRIV), or not privileged

{PASS } / {NOPASS} The program specified on PGMNAME can or cannot bypass security protection indicates that security protection is in effect, NOPASS indicates that security protection is not required. PASS is the default.

<u>VATLSTxx (Volume Attribute List):</u>

VATLSTxx contains **one or more volume attribute lists that predefine the mount and use attributes** of direct access volumes. The mount attribute determines the **conditions under which a volume can be demounted.** The use attribute controls the **type of requests for which a volume can be allocated.**

Three types of volume "mount" attributes are

- Permanently resident
- Reserved
- Removable

Three types of volume "use" attributes are

- Storage
- Public,
- Private.

Critical direct access volumes can be controlled as the "mount" and "use" attributes determine the type of data sets that can be placed on a volume. During allocation, data sets on volumes marked permanently resident or reserved are selected first because they require no serialization, thus minimizing processing time. A permanently resident volume is either one that cannot be physically demounted (that is, a drum, 3344, or 3350) or one that cannot be demounted until its device is varied offline.

A reserved volume remains mounted until the operator issues an UNLOAD or a VARY OFFLINE command. A volume is marked reserved when it is so designated in a volume attribute list, or when the operator issues a MOUNT command for the volume.
A removable volume can be demounted after its last use in a job, or when the device on which it is mounted is needed for another volume.
The use attribute controls the type of request for which a volume can be assigned:

- A Specific volume request
- A temporary, non-private non-specific volume request
- A non-temporary, non-private, non-specific volume request.

Three use attributes are used for allocating these types of volume requests, as follows:

- A private volume is allocated only to a specific volume request.
- A public volume is allocated to a temporary, non-specific volume request . Thus, a scratch data set would be placed on a public volume.
- A storage volume is allocated primarily to a non-temporary, non-specific volume request.

```
D U,DASD,ONLINE,,999
```

Master Catalog	a VSAM dataset that contains data set and volume information necessary to locate data sets and user catalogs.
Page and Swap	VSAM data sets that contain the paged-out portions of address spaces, data spaces, the Common Service Area (CSA), and the Pageable Link Pack Area (PLPA)
IODF	a VSAM data set called the I/O definition file that contains information about processors, channel paths, control units, and I/O devices
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
SYS1.BROADCAST	a BDAM data set that contains two types of TSO messages – notices and mail
SYS1.CMDLIB	a partitioned data set that contains TSO command processor routines, service routines, and utility programs
SYS1.CSSLIB	an optional partitioned data set that contains TSO/E command processor routines, service routines &utility programs
SYS1.DAE	a sequential data set that contains a permanent record of

	unique dumps identified by Dump Analysis and Elimination (DAE)
SYS1.DGTCLIB	a partitioned data set that contains the CLIST text for the Interactive Storage Management Facility (ISMF)
SYS1.DGTLLIB	a partitioned data set that contains system load modules for the Interactive Storage Management Facility (ISMF)
SYS1.DGTMLIB	a partitioned data set that contains the message text for the Interactive Storage Management Facility (ISMF)
SYS1.DGTPLIB	a partitioned data set that contains the panels for the Interactive Storage Management Facility (ISMF)
SYS1.DGTSLIB	a partitioned data set that contains the skeletons for the Interactive Storage Management Facility (ISMF)
SYS1.DGTTLIB	a partitioned data set that contains the tables for the Interactive Storage Management Facility (ISMF)
SYS1.DUMPnn	(SYS1.DUMP00 through SYS1.DUMP99) sequential data sets that contain SVC dumps
SYS1.FDEFLIB	a partitioned data set that contains forms definitions objects built by the Print Management Facility
SYS1.FONTLIB	a partitioned data set that contains various font objects built by the Print Management Facility
SYS1.HELP	a partitioned data set that contains HELP information regarding the syntax, operands & function of each TSO command
SYS1.IMAGELIB	a partitioned data set that contains the UCS (Universal Character Set) or FCB (Forms Control Buffer) images for printers such as the 3211 and 3800
SYS1.INDMAC	a partitioned data set that contains the macro definitions for the industry subsystems
SYS1.ISAMLPA	a partitioned data set that contains the ISAM component modules loaded into the Pageable Link Pack Area (PLPA)
SYS1.JES3LIB	a partitioned data set that contains all the JES3 code except JES3 modules that reside in SYS1.LPALIB and SYS1.LINKLIB
SYS1.JES3MAC	a partitioned data set that contains the macro definitions for JES3
SYS1.LINKLIB	a partitioned data set that contains non-resident system routines

	as well as the assembler program, the linkage editor, the utility programs, and service aids
SYS1.LOGREC	a sequential data set that contains statistical data about hardware failures and certain system software failures
SYS1.LPALIB	a partitioned data set that contains the modules to be loaded into the Pageable Link Pack Area including system routines, SVC routines, data management access methods, and some TSO modules.
SYS1.MACLIB	a partitioned data set that contains the macro definitions for supervisor and data management macro instructions
SYS1.MANn	(SYS1.MANA through SYS1.MANZ and SYS1.MAN0 through SYS1.MAN9) VSAM data sets that contain information collected by the System Management Facilities (SMF) routines or other measurement facilities
SYS1.MIGLIB	a partitioned data set that is the system load library for the Interactive Problem Control System (IPCS) and all component and subsystem dump exit modules
SYS1.MODGEN	a partitioned data set that contains macro definitions
SYS1.NUCLEUS	a partitioned data set that contains the resident portion of the control program as well as the nucleus initialization program (NIP), a pointer to the master catalog, and the I/O configuration members built by the MVS Configuration Program (MVSCP)
SYS1.OVERLIB	a partitioned data set that contains overlays generated by the Overlay Generation Language program for the IBM 3800 Model 3 Printing Subsystem
SYS1.PARMLIB	a partitioned data set that contains IBM-supplied and installation created lists of system parameter values
SYS1.PDEFLIB	a partitioned data set that contains page definitions built by the Print Management Facility
SYS1.PROCLIB	a partitioned data set that contains the cataloged JCL

	procedures for system tasks or processing program tasks invoked by the operator or the programmer
SYS1.PSEGLIB	a partitioned data set that contains page segments built by the Print Management Facility
SYS1.SAMPLIB	a partitioned data set that contains the installation verification procedure (IVP), the independent utilities, and the IPL text as well as sample exit routines
SYS1.SBLSCL10	a partitioned data set that contains the text of the CLISTs for the Interactive Problem Control System (IPCS)
SYS1.SBLSKEL0	a partitioned data set that contains the file tailoring skeletons for the Interactive Problem Control System (IPCS) dialog programs
SYS1.SBLSMSG0	a partitioned data set that contains the message text for the Interactive Problem Control System (IPCS) dialog programs
SYS1.SBLSPNL0	a partitioned data set that contains the dialog panels for the Interactive Problem Control System (IPCS) dialog programs
SYS1.SBLSTBL0	a partitioned data set that contains the dialog tables for the Interactive Problem Control System (IPCS) dialog programs
SYS1.SCBDCLIST	a partitioned data set that contains the CLIST procedures to invoke and run hardware configuration definition (HCD)
SYS1.SCBDHENU	a partitioned data set that contains the English-language help panels for the hardware configuration definition product (HCD)
SYS1.SCBDHJPN	a partitioned data set that contains the Japanese-language help panels for the hardware configuration definition product (HCD)
SYS1.SCBDMENU	a partitioned data set that contains the English-language messages for the hardware configuration definition product (HCD)
SYS1.SCBDMJPN	a partitioned data set that contains the Japanese-language messages for the hardware configuration definition product (HCD)
SYS1.SCBDPENU	a partitioned data set that contains the English-language panels for the hardware configuration definition product (HCD)

SYS1.SCBDPJPN	a partitioned data set that contains the Japanese-language panels for the hardware configuration definition product (HCD)
SYS1.SCBDTENU	a partitioned data set that contains the English-language key lists for the hardware configuration definition product (HCD)
SYS1.SCBDTJPN	a partitioned data set that contains the Japanese-language key lists for the hardware configuration definition product (HCD)
SYS1.STGINDEX	a VSAM data set that contains auxiliary storage management records for virtual I/O (VIO) data sets that MVS saves across job steps and between IPLs
SYS1.SVCLIB	a partitioned data set that contains some online test xecutive program (OLTEP) and appendage modules
SYS1.TCOMMAC	a partitioned data set that contains ACF/TCAM record API macros
SYS1.TELCMLIB	a partitioned data set that contains telecommunications subroutines in load module form
SYS1.UADS	a partitioned data set that contains a list of authorized TSO users and information about them such as userid, password and LOGON procedure name
SYS1.VTAMLIB	a partitioned data set that contains the ACF/VTAM load modules and related members such as logon exit routines and authorization and accounting exit routines

DAT	Dynamic Address Translation
IOCDS	I/O Configuration Data Set
IODF	I/O Definition File
IPL	Initial Program Load
JES	Job Entry Subsystem
MCS	Multiple Console Support
MSI	Master Scheduler Initialization
POR	Power-on-Reset
RSM	Real Storage Manager
SMS	System managed Storage
SVC	Supervisor Call
TOD	Time of Day Clock
UCW	Unit Control Word
VSM	Virtual Storage Management

. **ASM**	Auixiliary Storage Manager
. **ENF**	Event Notification Facility
. **IOCP**	I/O Configuration Program
. **IOS**	Input/Output Supervisor
. **IRIM**	IPL Resource Initialization Module
. **MCH**	Machine Check Handler
. **MIH**	Missing Interrupt Handler
. **NIP**	Nucleus Initialization Phase
. **RIM**	Resource Initialization Module
. **RTM**	Recovery Termination Manager
. **SRM**	System Resource Manager
. **SYSRES**	System residence Volume
. **UCB**	Unit Control Block
. **UIM**	Unit Information Module

MAINFRAME STARTUP PROCEDURE FOR S/390

STEP 1 SWITCH ON THE CONSOLE MONITOR.

STEP 2 PULL THE RED BUTTON

STEP3 WAIT TILL THE USERID AND PASSWORD SCREEN APPEARS .

STEP 4 **USER ID** **SYSPROG**

PASSWORD **PASSWORD** PRESS LOGON BUTTON

STEP 5 DOUBLE CLICK **"CONSOLE ACTIONS" IN VIEW**

STEP 6 DOUBLE CLICK **"START EMULATOR 3270 EMULATOR**"

STEP 7 MINIMIZE THE SCREEN A,B CLICK THE SMALL BOX IN THE SESSION A

STEP 8 AGAIN MINIMIZE FOR SESSION B

STEP 9 DOUBLE CLICK A LOCAL

STEP 10 GO TO TOP LEFT HAND CORNER & **CLICK FILE --> RUN OTHERS ---> SELECT ZOSOP.WS ---> OK**

STEP 11 MINIMIZE EVERYTHING

STEP 12 DOUBLE CLICK B LOCAL

STEP 13 GOTO LEFT CLICK **FILE ---> RUN OTHERS -->SELECT ZOS .WS---> OK**

STEP 14 MINIMIZE EVERYTHING

STEP 15 DOUBLE CLICK **GROUPS ---> CPC---> P001C61A**(SINGLE CLICK FOR SELECTING)

STEP 16 GOTO TO THE SIDE PANEL **CPC RECOVERY ---> POWER ON RESET** (DOUBLE CLICK)

STEP 17 PRESS (OK)

STEP 18 DOUBLE CLICK **GROUPS --> IMAGES ---> ZOS14**(SINGLE CLICK FOR SELECTING)

STEP 19 GOTO SIDE PANEL **DAILY -->ACTIVATE(DOUBLE CLICK)-->YES--OK**

STEP 20 DOUBLE CLICK C SESSION

ISSUE THE FOLLOWING **COMMANDS**

```
SET SMF=00
02, WARM, NOREQ
```

THEN WAIT TILL YOU GET A MESSAGE **" DYNAMIC COMMAND CAN BE ENTERED".**

THEN TYPE THE FOLLOWING **COMMANDS** AND ENTER

```
S TSO
S TCPIP
```

WAIT TILL YOU GET A MESSAGE FTPD ENDED

```
-START DB2
S CICS1
S TCPIP
S RMF
S VTAM
S SYSLOG
S VLF,SUB=MSTR
S DLF,SUB=MSTR
S LLA
S IRRDPTAB
S RRS
S OMVS
$S I
```

SHUTDOWN PROCEDURE FOR S390

1. ISSUE TH FOLLOWING COMMANDS FROM THE CONSOLE

P TSO	TO STOP TSO
C CICS1	TO STOP CICS
-**STOP DB2**	TO STOP DB2
P TCPIP	TO STOP TCPIP
P RMF	TO STOP RMF

Z NET,QUICK	TO STOP VTAM
C SYSLOGD1	TO STOP SYSLOGD1
&PI	TO CANCEL INITIATORS
F BPXOINIT,SHUTDOWN=FORKINIT	TO STOP OMVS
$P JES2,TERM	TO STOP JES

YOU WILL GET A MESSAGE **"RESOURCE TEMP UNAVAILBLE"**

SET SMF =01

Z EOD

YOU WILL GET A MESSAGE **" HARDCOPY DEVICE UNAVAILABLE"**

2. CLOSE ALL THE CONSOLE A,B,C,D

BY CLINKING THE CROSSED BOX IN THE TOP RIGHT HAND CORNER IN ALL THE SESSION-- A,B,C,D

3.THEN GO TO **GROUPS(DOUBLE CLICK)--> IMAGES (DOUBLE CLICK)—>ZOS14(CLICK ONES FOR SELECTING IT)**

4.GO TO THE SIDE PANEL UNDER **DAILY** SELECT

DEACTIVATE (DOUBLE CLICK)---> YES----->OK THEN GO TO

GROUPS(DOUBLE CLICK) ---->CPC(DOUBLE CLICK)-->P001C61A(CLICK ONCE FOR SELECTING IT)

5.GOTO THE SIDE PANEL UNDER **CPC RECOVERY** SELECT

SHUTDOWN(DOUBLE CLICK) ---> YES--> DONIT PRESS OK, IT WILL GET CLOSE AUTOMATICALLY

ONCE THE SCREEN BLACKS OUT---> RED SWITCH DOWN

JOB ENTRY SUBSYSTEM

Subsystems & its Functions

A subsystem is a set of elements which is a proper subset of the whole system. Subsystem is part of the MVS doing a function on behalf of it and having its own address space. There are two types of subsystem:**Primary & Secondary subsystem.** The primary subsystem (JES) is usually defined first because other subsystems, such as DB2 (secondary subsystem), require the services of the primary subsystem in their initialization routines.

After the primary JES is initialized, the other subsystems are initialized in the order in which the IEFSSNxx members are specified by the SSN parameter.

Command to display all the subsystem

```
D SSI
```

Some of the Subsystems are:

JES: Job Entry Subsystem: For Job management

RACF: Resources access control facility: For Security

DFSMS:Data Facility Storage Management Subsystem: For Data and space management

SMP/E: System Modification Program Extended: For installation

CICS: Customer information Control System. For Online transaction procession

DB2 : Database 2: For RDBMS

THE JOB ENTRY SUBSYSTEM (JES2 OR JES3)

Job Entry Subsystem (JES) is a primary subsystem and does the job management on behalf of the MVS.

- Each job is described to the operating system by system administrators or other users in job control language (JCL).
- The operating system then sends the job to the JES program.
- The JES program receives the job, performs the job based on priority, and then purges the job from the system.

There are two versions, JES2 and JES3. JES3 allows central control of the processing of jobs using a common work queue. Both OS/390 and MVS provide an interactive menu for initiating and managing jobs.

HASP

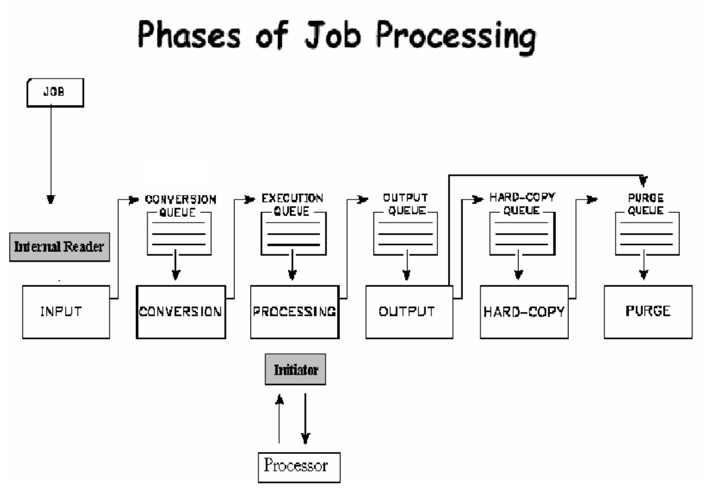
JES2 is descended from **Houston automatic spooling priority (HASP)**. HASP is defined as a computer program that provides supplementary job management, data management, and task management. HASP remains within JES2 as the prefix of most module names and the prefix of messages sent by JES2 to the operator.

Phases of JES

During the life of a job, both JES2 and the base control program of MVS control different phases of the overall processing. Generally speaking, a job goes through the following phases:

- Input
- Conversion
- Processing
- Output
- Print/punch (hard copy)
- Purge

Except for processing, all the job phases are controlled by JES2



INPUT PHASE

- As JES2 reads the input stream and places each job’s JCL, optional JES2 control statements, and SYSIN data onto the SPOOL data sets.
- The queue entry for each job contains the job number, job name, job class, and priority, pointers to JES2 control blocks on the SPOOL, and the next JES2 process for which the job is eligible.
- The job queue entry is used by JES2 to keep track of the job while it is in the system.

CONVERSION PHASE

The job stream is passed to the JES converter program, which:

- assigns a job number to the job (giving a unique identifier to jobs with the same jobname)
- analyzes the JCL statements
- merges in any cataloged procedures which they reference
- converts the JCL into "internal text", which is meaningful to JES and the MVS job scheduler
- checks for syntax errors, and if it finds any, fails the job with a "JCL error", placing it straight onto the output queue without queuing it for execution
- if no errors are found, the converter stores the internal text on the "spool" dataset and adds the job to the input queue

PROCESSING PHASE

- JES2 initiator selects jobs based on the job class(es) and the priority
- Each initiator is associated with one or more job classes in this way allows an installation to control job selection.
- After JES2 selects the job and passes it to the initiator, the initiator invokes the interpreter to build control blocks from the converter / interpreter text that the converted 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.

OUTPUT PHASE

- JES2 controls all SYSOUT processing. SYSOUT is a system-produced output; that is, all output produced by, or for, a job.
- The output includes system messages that must be printed, as well as data sets requested by the user that must be printed or punched.

HARD-COPY PHASE

The output queue can have output that is to be processed locally or output to be processed at a remote location (either an RJE workstation or another NJE node).

- The active devices that are attached locally or through RJE connections select the output data sets with characteristics that best match their selection criteria.
- Job output passing through to another JES2 nodes resides in the network output queue.
- JES2 selects a job’s output from the network output queue for transmission to another node based upon the priority and the desirability of reaching the output processing node over the availability transmission line.

PURGE PHASE

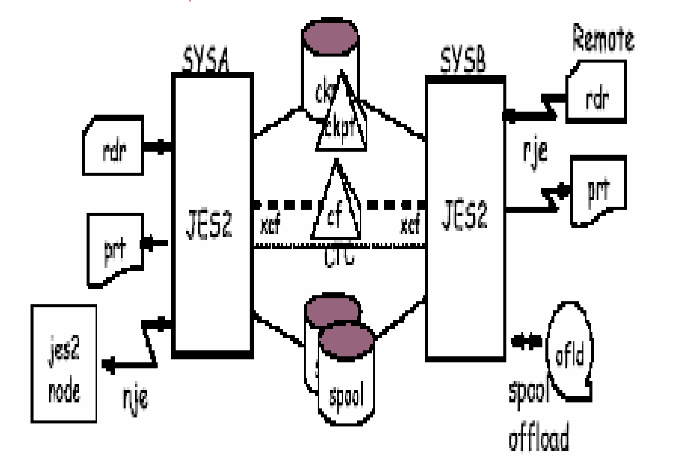
- When all processing for a job completes, JES2 releases the SPOOL space and the job queue space assigned to the job, making the space available for allocation to subsequent jobs.
- JES2 then issues a message to the operator indicating that the job has been purged from the system.

JES2 MULTI-ACCESS SPOOL (MAS)

- A multi-access SPOOL (MAS) configuration of two or more JES2 processors at the same physical location, all sharing the same SPOOL and check-point data sets.

- There is no direct connection between the processors; the shared direct access data sets provide the communication link.
- Each processor in a MAS complex operates independently of the other JES2 processors in the configuration.
- The JES2 processors share a common job queue and a common output queue, which resides on the checkpoint data sets.
- These common queues enable each JES2 processor to share in processing the installation's workload.

MultiAccess Spool (MAS) node w/ 1 - 32 Members:



EXERCISES

<u>JES COMMANDS</u>

- $D INIT - Display all Initiators
- $D I(init-id) - Display one particular Initiators
- $D I(from init-id – To initid) – Display few initiators .Eg: $DI(4-10)
- $T i(init-id), c=class - Modify Class for an initiator Eg: $TI20,C=m
- $DI,LONG,L=A == Long - display of Inits with jobnames and ASIDs
- $DA - Display Active jobs and printers
- $D JOB(jobname),LONG – Display job details in detail

- $D JOB(job-name) – Display job with job-name
- $D Jobid – Display job with job-id
- $DJOBDEF - See free job number counts
- $DSPL,JOBS=<u>001</u> - Show jobs using more than 001% of spool space
- $H JOB(job-name) – Hold a job using Job-name
- $H jobid – Hold a job using Job-id
- $A JOB(job-name) – Release a job using Job-name
- $A jobid – Release a job using Job-id
- $P JOB(job-name) – Purge a job using Job-name
- $P jobid – Purge a job using Job-id
- $D SPOOLDEF – Display Spool Defintion
- $D SPOOL – Display Spool Utilization and Spool Volume
- $DSPL,ALL
- $D JOBQ – Display JOB queue
- $D JOBQ,SPOOL - Display JOB queue with spool details
- $D JOBQ,SPOOL=TGS - Display JOB queue with TGS details
- $D JOBQ,SPOOL=TGS>=2 Display JOB queue with TGS more than 2
- $D JOBQ,SPOOL=PERCENT Display JOB queue with spool percentage
- $DCKPTSPACE - Check BERT available ($HASP050)

<u>DISPLAY COMMAND</u>

• D T	Displays the local time of the system
• D D	Displays the dump datasets Eg:sys1.dump
• D D,T	Displays the dump datasets with titles
• D SMF	Displays the SMF datasets Eg : sys1.MAN
• D C	Display console commands
• D A,L	Display active job list
• D J	Display active jobs
• D XCF	Display systems in the sysplex
• D TS,L	Display the active TSO users
• D GRS,C	Display system contention
• D C,B	Display the console buffer
• D M=CPU	Display the cpu status
• D M=CHP(X)	Display the paths to all devices

• D M=DEV	Display all paths to all devices
• D IPLINFO	Display IPL Information
• D R,L	Display outstanding records
• D R,U	Display outstanding mount machine information
• D PFK	Display pfk structure for the consoles
• D M=STOR	Display storage
• DD Clear,DSN=ALL	Clear all dump dataset
• D U,Tape,Online	Display tape drive online to the system
• D U,Tape,Offline	Display tape drive offline to the system
• D U,,Alloc,XXX,1	Display the job allocated to the device(XXX)
• C Userid	Cancel TSO user.
• $T I(3) class=y	Change an initiator ‘x’ to class’y’.
• $P jes2,term	Forcing jes2 to shutdown
• Set smf = 01	To halt SMF
• Z,EOD	To halt the MVS
• V (volume) offline	To disable a sms-volume for new location.
• D GRS,RES=(sysdsn,datasetname)	Display who has the dataset
• &H job(job-name) (or) $H jobid	Hold all queues

HARDWARE MANAGEMENT CONSOLE (HMC)

The Hardware Management Console provides a single point of control. The hardware Management Console communicates with each central processor complex (CPC) through the CPC's support element (SE). When tasks are performed at the Hardware Management Console, the commands are sent to one or more support elements which then issue commands to their CPCs. CPCs can be grouped at the Hardware Management Console so that a single command can be passed along to as many as all of the CPCs defined to the Hardware Management Console. One Hardware Management Console can control up to 100 support elements and one support element can be controlled by 32 Hardware Management Consoles. Refer to the example below and the example on the next page for typical Hardware Management Console configurations. The Hardware Management Console monitors all the defined support elements for the Following conditions.

- Views area background turns red
- Hardware messages.
- Operating system messages

VERSION OF HMC

- HMC 1.n.n - Open the HMC Settings in the Console Actions Work Area.
- HMC 2.n.n - User Settings in the Console Actions Work Area.

Hardware and Operating System messages.

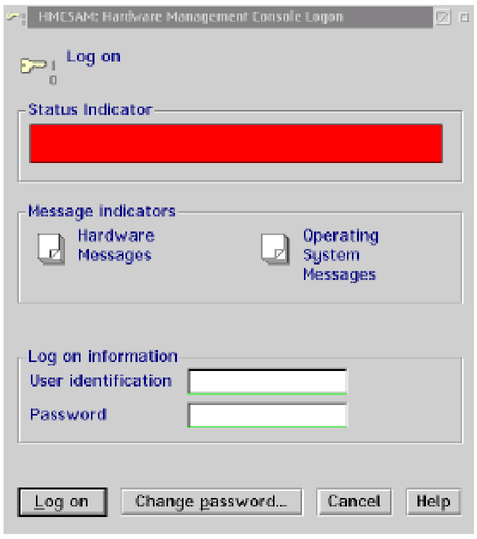
The recommended settings are

- Green - Views area background with no exceptions
- Red - Views area background with exceptions, Blue. Hardware Messages Pending
- Cyan - Operating System Messages Pending

The values may be changed by clicking the radio button and then clicking the desired color. Apply to activate. The slider bar indicates additional colors available to the right.

Starting the Hardware Management Console

When initialization is complete, the Hardware Management Console Logon window is displayed.



Default userids and passwords are established as part of a base Hardware Management Console. The Access Administrator **should assign new userids and** and passwords:

HMC Signon's

Operator	OPERATOR PASSWORD
Advanced Operator	ADVANCED PASSWORD
System Programmer	SYSPROG PASSWORD
Access Administrator	ACSADMIN PASSWORD
Service Representative	SERVICE SERVMODE

Overview of Activation profile

Activation Profiles are required for CPC and CPC image activation. They are used to tailor the operation of a CPC and are stored in the support element associated with the CPC. Activation profiles are used by the Activate task to:

- Perform a power on Reset
- Activate LP AR partitions
- IPL

There are three types of Activation profiles:

- Reset profile is used for POR
- Image profile is used in LPAR mode to initialize LPAR partitions
- Load profile is used to IPL

Default profiles

Three default profiles are supplied by IBM:

Default Reset profile (DEFAULT RESET)

Performs power-up and Power on Reset. Defines mode of operation, storage assignments, dynamic 110. I/O reset. LPAR activation sequence or load profile in basic mode.

- Used as a building block to create new Reset profiles
- Provide a new name and customize the parameters
- Assign and save the new profile
- Multiple Reset profiles may exist

Default Load profile (DEFAULTLOAD LOAD)

Defines the parameters required to IPL. Defines the type of load (load normal or load clear) IPL device number and the load parameters.

- Used as a building block to create new Load profiles
- Provide a new name and customize the parameters
- Assign and save the new profile
- Multiple Load profiles may exist

Default Image profile (DEFAULT IMAGE)

Defines the parameters required to activate each LPAR partitions (one for each partition) Defines mode of operation, numbers of CPs shared or dedicated, Storage assignments, Security options and optional load.

- Copied to EACH LPAR partitions Image profile
- Each Image profile has the name of the LPAR partition
- One and only one Image profile for every LPAR partition defined in the IOCDS

Three default profiles are supplied with the IBM servers.

- The Customize/Delete Activation profiles task is used to customize, delete, or assign profiles.
- Profiles can be assigned and viewed using the Details panel.
- Profiles can also be viewed from the Activate Task Confirmation panel.
- Profile are used to define the parameters to be executed by the activate task. each object in each group is assigned a profile.

Examples of HMC tasks

<u>Changing Load Parms / SYSRES</u>

1. Click on Groups

2. Choose the System
   SYSTEM1 or SYSTEM2. (just example names) click to highlight.
3. Under CPC Recovery, double click on LOAD.
4. Modify the Load address and/or of changes you want to make and click OK.

IPL

1. Click on Group
2. Choose the System SYSTEM1 or SYSTEM2.
3. Click on Load

Set Clock

1. Click on Groups
2. Click on Defined CPC'S
3. Click on Customize Support Element Date/Time. This is where you would update the date and time, When Finished:
   Click on Use New Time

Power-on-reset (POR)

1. Must Signon as SYSPROG
2. Click on Groups
3. Click Defined CPC'S
4. Highlight System name
5. Click Single Object Operations under CPC Recovery
6. Click POR
7. Click on ACTIVATE
8. Click on YES

You must be logged on as SYSPROG and Single Object Operations under CPC Recovery. If one of these are not correct POR will not work. After POR completes - Don't forget to click on ACTIVATE in Daily Menu. (POR disables CPC Recovery - you will not be able to LOAD LPAR or IPL until ACTIVATE is clicked)

Removing / adding Write Protect to the IOCDS

1. Must be signed on as SYSPROG
2. Click on Groups
3. Click Defined CPC'S
4. Highlight System name
5. Double click on Single Object Operations under CPC recovery
6. Click on Yes to continue with change.

You are now under the Support Element Workplace Screen.

7. Support Element CPC should be highlighted
8. Double click on CPC's
9. Highlight System name
10. System name should be highlighted
11. Under CPC Configuration
12. Double click on Input / Output
13. Input / Output Configuration screen should appear
14. Choose the one you want to change and click
15. Click on Options
16. Select Enable or Disable and click
17. Under Options to exit click on exit
18. To LOGOFF click on the box next to the Support Element Workplace.

Changing IOCDS

1. Must be signed on as SYSPROG
2. Click on Groups
3. Click Defined CPC'S
4. Highlight System name
5. Click Single Object Operations under CPC Recovery
6. Click Customize Delete / Activation Profile.
7. Select DEFAULT in Customize Delete / Activation Profile
8. Choose the system Profile name you want and click on customize.
9. Choose the system name to change and click save.
10. Click on YES to CONTINUE change

You must be logged on as SYSPROG and Single Object Operations under CPC Recovery. If one of these are not correct the IOCDS change will not occur.

System Activity

1. Click on Groups
2. Click on Defined CPC's
3. Click on Activity
4. Click on CPC activity bar

This will display system activity, swapping from systems every 10 seconds.

5. Click on Actions
6. Click on Stop

Resource Access Control Facility(RACF)

What is Security?

Making data secure does not mean just making confidential information inaccessible to those who should not see it. Rather, it means preventing the inadvertent destruction of files by people either knowingly or unknowingly.

Why Security?

As the general computer literacy and the number of people using computers has increased, the need for data security has taken on a new level of importance. With the increasing popularity of the World Wide Web, the need for security has gained more importance. z/OS security services comprise a variety of security related products, which have been grouped into the following three elements.

- z/OS security server, an optional feature of z/OS
- Integrated security services, a base element
- Cryptographic Services, a base element

z/OS basic security facilities

Integrity

Program property table (PPT)
Authorized program facility (APF)
Authorized programs
System authorization facility (SAF)

Auditing

Logs (hardcopy, system)
Generalized trace facility (GTF)
System management facility (SMF)

Integrated Security Services components

The basic security functions are shipped as two separate parts. The Security Server, ie RACF and the Integrated Security Services. The Integrated Security Services consists of the following components.

- IBM Tivoli Directory Server (LDAP Server)

- Network Authentication Service (Kerberos)
- Enterpise identity mapping (EIM)
- Open Cryptographic Enhanced Plug-ins (OCEP)
- DCE Security Server

The Firewall Technologies component was removed from the system with z/OS V1R8.

Cryptographic Services

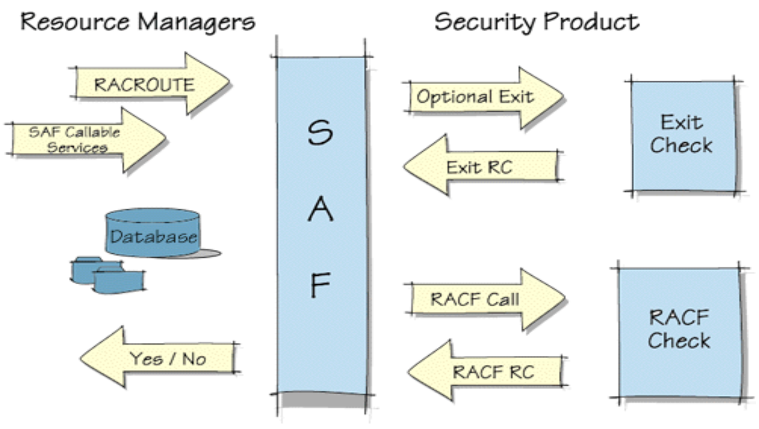
Cryptography is the transformation of data to conceal its meaning. In z/OS, the base element Cryptographic Services provides the following cryptographic functions: data secrecy, data integrity, personal identification, digital signatures, and the management of cryptographic keys.

- Integrated Cryptographic Service Facility (ICSF)
- Open Cryptographic Services Facility (OCSF)
- Public Key Infrastructure (PKI) Services
- System Secure Sockets Layer (SSL)

z/OS Security Server Components

SAF (SYSTEM AUTHORIZATION FACILITY)

SAF is part of the operating system. If you use SAF with RACF, you can enhance and complement your overall system security functions. SAF conditionally directs control to RACF, if RACF is present, or to a user-supplied processing routine, or both, when it gets a request from a resource manager. The key element in SAF is the SAF router, which is always available to you. It is a system service that gives you a common focal point for all products that provide resource control. With a common focal point, it's easier for you to use common control functions shared across products and across systems. The components and subsystems that manage your resources call the MVS router as part of some of their decision making functions, such as access control checking and authorization checking. These functions are called control points.



z/OS Security Server RACF

Prior to z/OS V1R5, the z/OS Security Server consisted of several components, now Resource Access Control Facility (RACF) is the only component. The z/OS Security Server RACF is an optional priced feature that allows an installation to control access to protected resources.
The operating system provides integrity. By using a Security Server, in this case RACF, you can protect resources by defining which resources should be protected and which groups of users or which individual users should have access to the defined resources. The definitions are kept in the RACF database. A RACF administrator defines users, user groups, and resources together with rules for how these resources may be used. RACF is "invisible" for most users if a good security structure is put in place.

RACF helps meet your needs for security by providing the ability to:

- Identify and verify users
- Authorize users to access the protected resources
- Control the means of access to resources
- Log and report attempts to access protected resources
- Administer security to meet an installation’s security goals

RACF provides these functions when the installation defines the users and the resources to be protected.

What is RACF?

RACF is an add-on software product that provides the basic security to a z/OS system. There are other security software products available such as from Computer Associates, ACF2 and Top Secret. RACF is included as part of the base z/OS system but requires a separate licence to be activated.
RACF gives you the ability to implement on your system the security policies you choose.

- RACF, Resource Access Control Facility is an add-on product to implement and control the installation's security policies on z/OS systems.
- Access to protected resources is controlled by rules.
- Access to resources are logged and can easily be monitored by an Auditor
- Users, groups and resources together with access rules are administrated by an Administrator

RACF's Role?

RACF helps meet the need for security by providing:

- Flexible control of access to protected resources
- Protection of installation-defined resources
- Ability to store information for other products
- Choice of centralized or decentralized control of profiles
- An ISPF panel interface
- Transparency to end users
- Exits for installation-written routines

RACF Resources

- RACF Classes & Profiles
- RACF Database
  - Access Control List
  - Class Descriptor Table
  - Global Access Checking Table
  - Started Procedures Table

User

An individual member identified by his unique id and verified using a password.

Group

A group is a collection of RACF users who share common access requirements to protected resources or who have similar attributes within the system.

Resource

Any information stored on a system such as datasets, Terminals, volumes etc.

Profiles

A profile is a record of RACF information that has been defined by the security administrator. When the security administrator or a delegate defines authorized users, groups, and protected resources, RACF builds *profiles.* Profiles contain the information RACF uses to control access to the protected resources. Each profile is owned by a user or group. By default, the owner of a profile is the user who creates it. Profile RACF stores all this information about users, groups, and resources in profiles There are user, group, and resource profiles.

Resources managed

You can use RACF to control access to:

- The system
- Subsystems and applications, including: JES, including job names, JES commands, consoles, JES input devices, spool, writers (printers) and others
- TSO, DB2
- IMS trans, programs, files, journals etc
- CICS files, trans, programs, journals etc
- Storage Management Subsystem (SMS)
- VTAM
- APPC sessions, transactions and resources
- Terminals
- MVS and JES consoles
- Catalogs
- DASD and tape data sets, DASD and tape volumes
- SYSIN and SYSOUT data on the JES spool
- Load modules (programs) for execution only, or copying as well
- Installation-defined resources

RACF functions

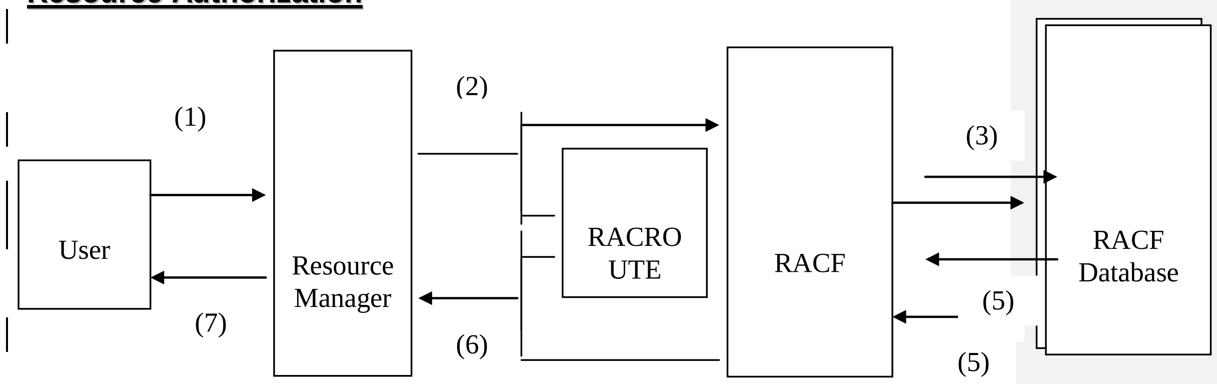
RACF protects resources by granting access only to authorized users of the protected

- Identify and authenticate users
- Resource authorization
- Log and report access to protected resources
- Security administration
- RACF database

User Identification and Verification:

RACF uses a user ID and a system-encrypted password to perform its user identification and verification.
During terminal processing, RACF allows the use of an operator Identification card (OIDCARD) in place of, or in addition to, the password. (The OIDCARD information is also encrypted). By requiring a user to know both the correct password and the correct OIDCARD, you have increased assurance that the proper user has entered the user ID.

Resource Authorization



User requests access to a resource using a resource manager (like TSO/E). Resource Manager gives a RACF request to see if the user has access to the resource (Generally using a RACROUTE Macro). RACF checks the RACF database where the user and resource access profile is stored.
RACF receives the information about the profile and passes it on to the resource manager. The Resource manager grants or denies permission to the requestor.

Log and report access to protected resources

It keeps track of what happens on the system so that an organization can monitor who is logged on the system at any given time. RACF reports if persons have attempted to perform unauthorized actions.

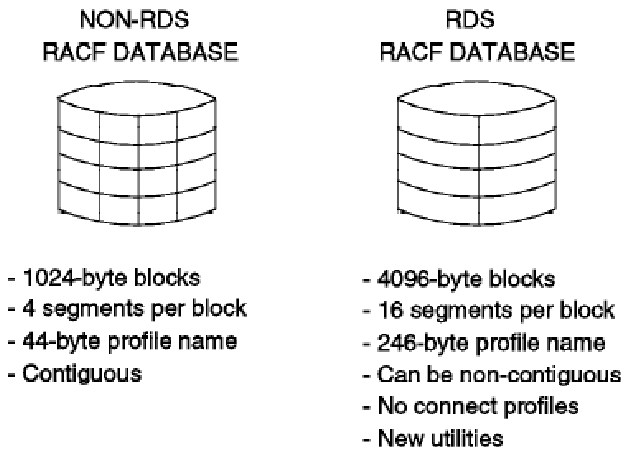
Security Administration:

The user with the RACF SPECIAL Attribute has the authority to control the access of various resources by defining the profiles.

RACF Database:

The RACF database contains information about all the profiles (viz. user, group and dataset) and all other resources defined to RACF. A backup database is maintained to ensure that any. physical errors to the original database do not affect the security definitions of the system. Control can be switched to the backup database by using RVARY command. It is comprised of two or more datasets called the RACF datasets.
Ex.

- SYS1.RACF.PRIM
  SYS1.RACF.BACK



- RACF 1.9.0 is the last release supporting the non-restructured database (NRDS) format.
- Therefore, it is strongly recommended that new users create a restructured RACF database. Also, installations that have RACF installed are encouraged to convert to the restructure database format (RDS) at this time.
- The new restructured RACF database provides more efficient management of information and a flexible base for further enhancements, as follows:
    - The block size is 4096 bytes, which will better utilize storage devices.
    - RACF still allocates space with 256-byte segments, so there are 16 Segments in each 4-KB blocks (K=1024).
    - Profile names can now be up to 246 bytes in length. Longer names may be needed for some profiles, such as JESSPOOL profiles.
    - In a non-restructured database, a profile must occupy contiguous space. Profiles in a restructured database need not be contiguous.

There are no CONNECT profiles in an RDS. The information that is kept in a CONNECT profile in a NRDS is kept in the USER profile in an RDS. This results in increased processing efficiency because the CONNECT and USER profile are commonly used together.

- The RACF database must be cataloged and should reside on a permanently resident volume. The RACF database must be placed on the fastest available device for performance. If the device is generated as a shared device, performance will be degraded, even though the RACF database is not actually being shared with another system.

Access Control List

- A collection of all access rights for one object. A list associated with an object that identifies all the subjects that can access the object and their access rights.For example, a list associated with a file that identifies users who can access the file and identifies their access rights to that file.

Class Descriptor Table

- A table consisting of an entry for each class except the USER, GROUP, and DATASET classes. The CDT contains the classes supplied by IBM and the installation-defined classes.

Global Access Checking Table

The Global Access Checking (GAC) is a table built in storage and contains the names of resources that are shared among all users, and are frequently accessed by users in a common manner. When determining a user's authority to access a resource, RACF consults the GAC table before it looks at any other security profiles. Use of this table results in less I/O to the RACF database and better system performance.
The GAC can only allow access, not deny, to a resource. If a user wishes to access a data set with a higher level than specified in the GAC or the resource is not found in the table, RACF profile checking is performed.

The Global Access Checking Table is strictly a performance and tuning tool. When RACF grants access to a resource because of an entry in the GAC table, RACF does not log the event (even if you request logging) and maintains no statistics.

Administering Groups

Groups

A group can be defined to serve as an anchor point for users who otherwise have no common access requirements. Groups can be defined based on access level. Any users requiring access would be connected, as appropriate, by the group administrator.

Group Naming Conventions
The naming conventions for groups are relatively simple: A group name must be from 1 to 8 characters long, chosen from the letters (A-Z), numbers (0-9), (#, $, @).
No two groups can have the same name. No group name can be the same as a user ID.

Group Structure

RACF conforms naturally to a tree structure of groups. t the top of the RACF group-user structure is a group called SYS1. This group is defined when RACF is installed. The SYS1 group is the highest group in the total RACF group-user structure. It is predefined. Every other group has a superior or owning group. Groups can correspond directly to business entities such as divisions, departments, and projects. Users can be connected to one or more groups

Based on the basic purpose, there are 5 types of groups

- Administrative Groups

- Holding Groups
- Data Control Groups
- Functional Groups
- User Groups

Benefits of Using RACF groups:

- Reducing the Effort of Maintaining Access Lists
- Avoiding the Need to Refresh In-Storage Profiles
- Providing a Form of Timed PERMIT

Delegation-Group Authority:

Each user in a group requires a level of group authority for that group. If a user is connected to several groups, the user has a level of group authority for each group. Group-SPECIAL is the easiest way to do delegation. USE, CREATE, CONNECT and JOIN do not propagate as group-SPECIAL does.

USE: The group authority of USE is not an administrative authority. When a user is connected to a group he/she will atleast have USE authority.

CREATE: The use with CREATE privilege can define dataset profile and also create datasets. Note that the user with group-SPECIAL cannot create datasets.

CONNECT: This privilege allows a user to connect other users to the group.

JOIN: permits the user to create new groups as sub-group of the group. If the user has CLAUTH(USER) then the user can also define new users to RACF.

1. DEFINING Groups

ADDGROUP | AG (group-name .) [DFP SEGMENT] [OMVS SEGMENT] [OWNER(userid or group-name)] [SUPGROUP(group-name)]

EX: ADDGROUP grp1 OWNER (ibmuser) SUPGROUP (sys1)

group-name : specifies the name of the group whose profile is to be added to the RACF database.

DFP: allows for specification of defaults for DFSMS application names and data storage and management class names.

OMVS: specifies OS/390 UNIX System Services information for the group being defined to RACF.

OWNER (**userid or group-name) :** specifies a RACF-defined user or group to be assigned as the owner of the new group.

SUPGROUP (group-name) : specifies the name of an existing RACF-defined group. This group becomes the superior group of the group profile you are defining.

2. ALTERING GROUPS

Use this command to change the parameters of a RACF defined group.

EX: If the group requires access to UNIX resources, alter the profile to include an OMVS segment with a GID.

```
ALTGROUP grp1 OMVS(GID(100))
```

3. LISTING GROUP INFORMATION:

This command lists the information for the specified group.

Command: {LISTGRP | LG} [{(group-name ...) | *}] [DFP] [OMVS]

```
EX:  LG Grp1
     LG Grp2 OMVS
```

4. DELETING GROUPS

DG command deletes a group from the RACF database.

Command:
DELGROUP | DG (group-name ...)
EX: DG Grp1

ADMINISTERING USERS

USERS

A RACF user is identified by an alphanumeric user ID that RACF associates with the user. A RACF user need not be an individual. From the security standpoint equating a user ID with anything other than an individual can be undesirable because individual accountability is lost. The default user is IBMUSER. Groups can be used for user administration, for resource ownership and decentralization of resources. Every user defined to RACF must be a member of atleast one group. The user may also be connected to more than one group.
The profiles can be can be owned by either a user or a group. To improve ease of maintenance, it is better to have the profiles owned by the group. Administration can be done either centralized or delegated.

User Attributes

User attributes can be assigned by specifying operands on RACF commands. User attributes describe various extraordinary privileges, limitations, and processing environments that can be assigned to specified users in a RACF-protected system. They can be assigned at either the system level or at the group level. When assigned at the system level, attributes are effective for the entire RACF-protected system. When assigned at the group level, their effect is limited to profiles that are within the scope of the group. The user attributes are :

SPECIAL,AUDITOR,CLAUTH,OPERATIONS,GRPACC,ADSP,REVOKE.

SPECIAL Attribute:

A user who has the SPECIAL attribute at the system level can issue all RACF commands. The SPECIAL attribute gives the user full control over all of the RACF profiles in the RACF database.
SPECIAL attribute can also be given at the group level. When it is done, the group-SPECIAL user has full control over all of the profiles within the scope of the group.

AUDITOR Attribute:

A user who has the AUDITOR attribute at the system level has the authority to specify logging options on the ALTDSD, ALTUSER, RALTER, and SETROPTS commands. He cannot make changes except to those related to logging. This does not give any access to resources.

OPERATIONS Attribute:

The OPERATIONS attribute is for the user in-charge for maintaining the DASD volumes on the system. The OPERATION attribute gives the ability to DUMP all the MVS datasets in the system.

CLAUTH Attribute

The CLAUTH (class authority) attribute allows the user to define profiles in a specific RACF class. A user can have class authority for the USER class and any of the classes that are defined in the class descriptor table (CDT). Examples of classes that IBM supplies in the CDT are the TERMINAL class (for terminals) and the TAPEVOL class (for tape volumes).

GRPACC Attribute

When a user with the GRPACC attribute creates a data set profile for a group data set, RACF gives UPDATE access authority to other users in the group (if the user defining the profile is a member of that group). A group data set is a data set whose high-level qualifier, or the qualifier derived from the RACF naming convention table, is a RACF-defined group name.

ADSP Attribute

The ADSP attribute establishes an environment in which all permanent DASD data sets created by this user are automatically defined to RACF and protected with a discrete profile. ADSP can be assigned at the group level, in which case it is effective only when the user is connected to that group.

REVOKE Attribute

The REVOKE attribute prevents the RACF-defined user from entering the system. REVOKE can be assigned at the group level, in which case the user cannot enter the system and connect to that group.

RESTRICTED Attribute

The RESTRICTED attribute prevents a user from gaining access to a protected resource, other than a z/OS UNIX file system resource, unless the user is specifically authorized on the access list. Global access checking, the ID(*) entry on the access list, and the UACC will not be used to allow a restricted user to access a protected resource.

User Naming Conventions:

A RACF user ID must be from 1 to 8 characters in length, and may consist of any combination of A-Z, 0-9, (#, $, @). Although RACF permits 8-character user IDs, TSO user IDs and user IDs on JOB cards cannot be more than 7 characters. TSO and MVS also require that the first character of user IDs be A-Z, (#,$,@). No two user IDs can be the same. No user ID can be the same as a group name.

1. ADDING User

Before creating a user, decide on the following attributes. default connect group, User ID, Password , Security, owner of the profile, User attributes, Security Labels, DFP, CICS, OMVS, TSO segment attributes.

{ADDUSER | AU} (userid .) NAME('user-name') USER-ATTRIBUTE DFLTGRP(group-name) OWNER(userid /group-name) OMVS SEGMENT | CICS SEGMENT| TSO SEGMENT PASSWORD(password) WHEN([DAYS(day-info)] [TIME(time-info)])]

EX:

AU User1 OWNER(ibmuser) DFLTGRP(Grp1)   NAME('ARICH') PASSWORD(ABC)

The TSO SEGMENT:

If a TSO segment has been defined in the user's profile, TSO checks the user's authority to use certain TSO resources when the user logs in. If no TSO segment has been defined then TSO checks the SYS1.UADS data set for the information it needs to build a session. If TSO does not find an entry for the user in SYS1.UADS, the user is denied access to the system. When this TSO information is moved into the RACF database, it is stored in the TSO segment of the user's profile. The TSO segment in the user profile consists of the following.

ACCTNUM	Accounting information
DEST	Destination id
MAXSIZE	Maximum region size
SIZE	Default region size
PROC	Logon procedure name
JOBCLASS	Jobclass id
MSGCLASS	Message class name
SYSOUTCLASS	Sysout class name
HOLDCLASS	Hold class name
SECLABEL	Security label name
UNIT	Unit-name
USERDATA	User data

<u>**TSO General Resource Classes**</u>

TSOPROC - Protects TSO/E logon procedures.
ACCTNUM - Protects TSO/E account numbers.
TSOAUTH - Protects TSO/E user attributes, OPER, JCL, ACCT, MOUNT, RECOVER, PARMLIB, TESTAUTH, and CONSOLE.
PERFGRP - **P**rotects TSO/E performance groups

The CICS segment :

The CICS segment of the RACF user profile contains data for CICS users. The information you can specify in the CICS segment is as follows:

OPCLASS({<u>1</u>|number})	Operator Class
OPIDENT({<u>blank</u>|name})	Operator identification code
OPPRTY({<u>0</u>|number})	Operator priority value
TIMEOUT({<u>0000</u>|hhmm})	Time out value for CICS terminal
XRFSOFF({<u>NOFORCE</u>|FORCE})	CICS persistent sessions restart and extended recovery facility (XRF) sign-off option

2. ALTERING Users

ALTUSER command is used to alter the parameters of a defined user. The parameters take the same meaning as in the case of ADDUSER.

EX: ALU USR001 PASSWORD(PASS01) RESUME

3. DELETING Users

DELUSER: Use the DELUSER command to delete a user from RACF.

{DELUSER | DU} (userid ...)

EX: DELUSER USR001

4. LISTING Users:

Use the LISTUSER command to list the details of specific RACF user profiles. A user profile consists of a RACF segment and, optionally, other segments such as TSO and DFP.
{LISTUSER | LU} (userid ...) CICS | DFP | OMVS | TSO

EX: LU USR1 TSO OMVS

<u>CONNECTING USERS to a GROUP:</u>

This command is used to connect a user to a group or to modify a user's connection to a group.

```
{CONNECT | CO}  (userid ...)  [ GROUP(group-name) ]  [ OWNER(userid
                or group-name) ]   USER ATTRIBUTES
```

EX: CONNECT usr1 GROUP (grp1) SPECIAL AUDITOR ADSP OPERATIONS

<u>REMOVING USERS FROM A GROUP:</u>

REMOVE command removes a user from a group and optionally assigns a new owner to any group dataset profile the user owns on behalf of that group.
{REMOVE | RE} (userid ...) [GROUP(group-name)] [OWNER(userid or group-name)]

EX: RE usr1 GROUP(grp1)

Protecting Datasets

Dataset

Datasets can be protected using discrete or generic dataset profiles.
Dataset profiles are of 2 types – Discrete, Generic

Generic and Discrete Profiles:

DATASET NAME	DISCRETE PROFILE	GENERIC PROFILE
USR1.UTIL.SAMP	USR1.UTIL.SAMP	USR1.*.SAMP
USR1.REXX.SAMP	USR1.REXX.SAMP	
USR1.JCL.SAMP	USR1.JCL.SAMP	

Discrete Profile: The profile is matched one-one with the resource. The name of the profile is the same as the name of the dataset. Discrete profiles have a one to one relationship with the resource. Discrete profiles provide very specific levels of control and should be used to protect sensitive resources
Ex. ‘ARICH01. FIRST.PDS’

Generic Profile: The generic profile covers more than one dataset. Generic Profiles have a one to many relationship with the resource. Fully qualified generic dataset profile can also be defined. Fully qualified dataset profile is a generic profile that does not have any generic character in the profile name. The generic characters that can be used are: %, *, **. . A single generic profile can protect many datasets having similar naming structure.
Ex. ‘ARICH01.**’

1. Adding dataset Profiles

The command is used to add RACF protection to data sets with either discrete or generic profiles.

{**ADDSD | AD} (profile-name-1 [/password] ...) [NOTIFY [(userid)]] [OWNER(userid or group-name)] [UACC(access-authority)] [UNIT(type)] [VOLUME(volume-serial ...)]**

Profile-name-1: specifies the name of the discrete or generic profile to be added to the RACF database.

NOTIFY [(userid)]: specifies the user ID of a RACF-defined user to be notified whenever RACF uses this profile to deny access to a data set.

OWNER (userid or group-name): specifies a RACF-defined user or group to be assigned as the owner of the dataset profile.

UACC (access-authority): specifies the universal access authority to be associated with the data sets.

EX: **Generic Profile** : ADDSD ‘abc.**‘ UACC(READ)
Discrete Profile : ADDSD ‘abc.load’ UACC(None)

2. PERMIT: The PERMIT command is used to maintain the lists of users and groups authorized to access a particular resource.

{PERMIT | PE} profile-name-1 [ACCESS(access-authority) | DELETE]

profile-name-1: specifies the name of an existing discrete or generic profile whose access list is to be modified

ACCESS(access-authority): specifies the access authority you want to associate with the names that you identified on the ID operand.

ID : Userid / Group name for whom the access need to be given.

EX: PERMIT ‘ABC.**’ ACCESS(UPDATE) ID(USER1)

3. Altering Dataset Profiles

The ALTDSD command can be used to modify an existing discrete or generic data set profile.

4. Listing Dataset Profiles

The LISTDSD command to list information included in tape and DASD dataset profiles.

{LISTDSD | LD} [ALL] [{DATASET(profile-name ...) | ID(name ...) |

EX: LD DATASET(‘ABC.**’)
LD DATASET(‘ABC.**’) ALL

5. Deleting Dataset Profiles

The DELDSD command is used to remove RACF protection from tape or DASD datasets that are protected by either discrete or generic profiles.

{DELDSD | DD} (Profile-name...)

EX: DD ‘ABC.LOAD’

CATALOG PROTECTION:

Catalog Type	Catalog Administrator	Data Manager	Other Users	Catalog Profile Name
Master Catalog	ALTER	UPDATE	READ	CATALOG.MASTER.**
User Catalog	ALTER	-	UPDATE	CATALOG.USERS.**

General Resources

A general resource profile provides RACF protection for computer resources, other than data sets. General resources are all the resources that are defined in the class descriptor table and include, among others, General resources are grouped in classes. General resources with similar characteristics belong to the same class. When a general resource profile is created, general resource class the profile is in, must be specified.

For Ex. DASD volumes, Tape volumes, Terminals, Load modules (programs),Subsystems & Applications,,TSO,DB2,Catalogs,SMS,CICS Files, Transactions, Programs, ,Journals etc., IMS Files, Transactions, Programs, Journals, MVS & JES Consoles, VTAM.

Like a generic profile, a resource group profile protects several resources with identical security requirements. However, the resources do not have to have similar names. Resource group profiles with similar characteristics belong to the same resource group class.

RACF allows the installation to set its own rules for controlling the access to its resources by defining what is controlled at what level.

Resource profiles contain:

- The owner of the profile
- The auditing parameters
- The Universal Access authority
- An access list with users and groups
- A “Warning” indicator
- A security classification
- A real-time notification information
- An erase-on-scratch indication for data sets
- A volume and a unit (if data set)
- A security retention period (if tape data set)
- Access statistics

RACLIST Processing

When **SETROPTS RACLIST** processing is activated, the sharing of both in-storage discrete and in-storage generic profiles is enabled for the classes specified.

SETROPTS | SETR RACLIST(*classname*)

Ex.

SETR RACLIST(RACFVARS),SETR RACLIST(TERMINAL)

RACFVARS Class

Starting with RACF 1.9, profiles can be created in the **RACFVARS** class whose profile names act like programming variables (indicated by an &). RACFVARS class can be used to create general resource profiles that protect many resources with unlike names. The use of RACFVARS reduces the number of profiles that are needed. Variables cannot be used in data set profile names. All of the **resources must belong to the same class** and must belong to a class that accepts generic profile names.The RACFVARS class requires high performance and hence the profiles must be added to RACLIST.

SETR CLASSACT(RACFVARS) RACLIST(RACFVARS)

Activate the RACFVARS class and RACLIST the profiles.

To create a profile in the RACFVARS class, specify the resource names on the ADDMEM operand:

RDEFINE | RDEF RACFVARS &*profile-name* UACC*(permission)* ADDMEM*(resource name)*

Ex.

RDEF RACFVARS &PAYTAPE UACC(NONE) ADDMEM(TAP111 A22222 B33OLD)

RDEF TAPEVOL &PAYTAPE UACC(NONE)

PE &PAYTAPE CLASS(TAPEVOL) ID(PAYGRP) ACCESS(ALTER)

SETR CLASSACT(TAPEVOL RACFVARS) RACLIST(RACFVARS)

To List all the Classes

SETROPTS LIST (or) SETR LIST

To search the profiles in a Class

SEARCH CLASS(class-name)

To Activate or de-activate a Class

SETR CLASSACT(class-name)
SETR NOCLASSACT(class-name)

To Define a profile in the Class

RDEF class-name profile-name UACC(NONE | READ | UPDATE | EXECUTE) NOTIFY(userid)

To Alter the definition in the Class

RALT class-name profile-name UACC(NONE | READ | UPDATE | EXECUTE) NOTIFY(userid)

To Permit a profile to a user or group

PERMIT profile-name CL(class-name) ID(user or group) ACCESS(NONE | READ | UPDATE | EXECUTE)

To delete the permission to a user or group

PERMIT profile-name CL(class-name) ID(user or group) ACCESS(NONE | READ | UPDATE | EXECUTE) DELETE

To List the profile in a class

RLIST class-name profile-name ALL

To Delete the profile in a class

RDEL class-name profile-name

For Direct Access Storage Device Volume Protection

DASDVOL A user given access via a profile in the DASDVOL class can access datasets only when using DFDSS, ICKDSF or the SCRATCH function. The user can be given the DASDVOL authority for specific volumes or for all volumes. The user can also be restricted to only certain type of volume operations.

Activate the DASDVOL class as

SETROPTS CLASSACT (DASDVOL) GENERIC (DASDVOL)

Then RDEFINE is used to define the volumes to be protected.

RDEFINE DASDVOL <vol-names> OWNER (<owner name>) UACC (NONE)

Administrators or any other users requiring authorization over the DASDVOL class are given access.

PERMIT < vol-names > CLASS (DASDVOL) ID (<user or group id>) ACCESS (ALTER)

For Tape Volume Protection:

TAPEVOL The TAPEVOL general resource class is used to control access to tape volumes. This control access at the volume level and not at the dataset level. A TAPEVOL profile can be defined with the RDEFINE command. The name of the TAPEVOL profile is the volume serial of the tape volume. Access to the tapes can be given using PERMIT command.

For Tape Dataset Protection:

TAPEDSN Tape dataset protection can be activated by using the TAPEDSN operand of the SETROPTS command. When tape dataset protection is activated, RACF refers to profiles in the DATASET class when verifying a user's access authority to a tape data set.

```
SETR CLASSACT (TAPEDSN TAPEVOL)
RDEF TAPEVOL phy UACC (NONE) AUDIT (ALL)
RDEF TAPEDSN phy.bkp UACC (NONE) AUDIT (ALL)
PE phy CLASS (TAPEVOL) ID(USR1) ACCESS(READ)
PE phy.bkp CLASS (TAPEDSN) ID(USR1) ACCESS(READ)
```

For Protecting Terminals

TERMINAL Using the TERMINAL general resource class it is possible to

- Control the Use of Undefined Terminals
- Restrict Specific Groups of Users to Specific Terminals
- Restrict the Times that a Terminal Can Be Used

A security level can be assigned to a terminal. If a user with a higher security level logs on from the terminal, his security level will be pulled down to that of the terminal. If the security level of the terminal is higher than that of the user, he is prevented from accessing sensitive data. Consider ten terminals with addresses TERM0001 to TERM0010.

EX:

Allow only users of group ACCT to use the terminal TERM0004 and TERM0007.

```
RDEF TERMINAL TERM0004 UACC (NONE)
PE TERM004 CLASS (TERMINAL) ID (ACCT) ACCESS (READ)

RDEF TERMINAL TERM0007 UACC (NONE)
PE TERM007 CLASS (TERMINAL) ID (ACCT) ACCESS (READ)

SETROPTS CLASSACT (ACCT) TERMINAL (READ)
```

Any user can logon to the terminal not defined whereas the terminals TERM0004 and TERM0007 can be used by those belonging to the ACCT group only.

Grouping class of terminals:

GTERMINAL Grouping class of terminals GTERMINL can be used to define many terminals with less number of profiles. The profiles must be RACLISTed. When RACLISTing of TERMINAL profiles is done, RACF combines the profiles from the TERMINAL and GTERMINL classes and builds the profiles in memory.

```
RDEF GTERMINL ACCTTERM UACC (NONE) ADDMEM (TERM0004 TERM0007)

PE ACCTTERM CLASS (GTERMINL) ID (ACCT) ACCESS (READ)

SETROPTS CLASSACT (TERMINAL) RACLIST (TERMINAL)
```

Surrogate Job submission :

SURROGAT Surrogate job permission gives the user the rights to submit jobs on behalf of other users. This can be done without knowing the password of the other user whose access authorities are used during the execution of the job.

- The resource class SURROGAT must be activated
- The submitter indicates the user in the job card
- When the submitter and user are different RACF carries out an authorization check
- a SURROGAT profile is required with READ access granted to the submitter

```
RDEF SURROGAT userid.submit  UACC (NONE)
PE  davin3.submit CLASS (SURROGAT) ID (ACCT) ACCESS (READ)
```

For Controlling Sending & Receiving Message:

SMESSAGE Using the SMESSAGE class individual users can be singled out for special treatment. If users are not defined they can receive messages from anyone.

SETROPTS CLASSACT(SMESSAGE DIRAUTH) RACLIST(SMESSAGE)

RDEFINE SMESSAGE USR1 UACC(READ)

PERMIT USR1 CLASS(SMESSAGE) ID(USR2) ACCESS(NONE)

Here the user USR2 is prevented from sending messages to USR1. The checking is done by VTAM. DIRAUTH is activated to provide logging. It is recommended that the SMESSAGE class be RACLISTed. SECLABELs and the MSGPROTECTION (ON) parameter in the SYS1.PARMLIB member IKJTS000 can control viewing messages. The recipient will not be able to view the message if the recipient's SECLABEL does not dominate that of the sender.

For Protecting SDSF:

SDSF will make calls to SAF/RACF for virtually every interaction with its resources.
SDSF resources that can be protected by RACF:

- SDSF panels and commands
- MVS and JES2 commands
- The modification and control of initiators and printers
- The browsing of SYSIN and SYSOUT datasets

SDSF AUTHORIZED COMMANDS:

The unique SDSF commands are defined in the SDSF class. The SDSF commands are covered by profiles with a first qualifier of ISFCMD. The generic name 'ISFCMD.**' covers them all. There are four categories of commands that can be defined individually or covered by a lower generic. The required access for these profiles is READ.

ISFCMD.DSP.** - which covers DA,I,H,O and ST
ISFCMD.FILTER.** - WHICH COVERS ACTION, DEST, FINDLIM, INPUT, OWNER, PREFIX and SYSID.
ISFCMD.ODSP.** - which covers INIT, LOG and PR.

For Protecting CICS & IMS supplied transactions:

TCICSTRN, TIMS
RACF resource profiles can be used to control access to transactions. In both CICS and IMS profiles can be defined in either the member class or the grouping class or both.

Member Class: The member class of IMS transaction is TIMS and that of CICS is TCICSTRN. The generic characters (*, **, %) can be used in the profile name.

CICS

```
RDEFINE TCICSTRN CEMT UACC (NONE)
PERMIT CEMT CLASS (TCICSTRN) ID (GRP1) ACCESS (READ)
SETROPTS CLASSACT (TCICSTRN)
```

IMS

```
RDEFINE TIMS TRA1 UACC (NONE)
PERMIT TRA1 CLASS (TIMS) ID (GRP2) ACCESS (READ)
```

The profiles that are RACLISTed into main memory take up less memory since their access lists are shorter. There is less need to rebuild the in-storage profiles.

Grouping Class

GCICSTRN,GIMS

GIMS and GCICS are the grouping classes for IMS and CICS respectively. As it is for any other class, the profile is define using RDEFINE and privilege granted using PERMIT.

CICS

```
RDEFINE GCICSTRN GRPT1 UACC (NONE) ADDMEM (TRA1, TRA2, TRA3)
PERMIT GRPT1 CLASS (GCICSTRN) ID (GRP1) ACCESS (READ)
SETROPTS CLASSACT (GCICSTRN)
```

IMS

```
RDEFINE GIMS GRPT2 UACC (NONE) ADDMEM (IMS1, IMS2, IMS3)
PERMIT GRPT2 CLASS (GIMS) ID (GRP1) ACCESS (READ)
SETROPTS CLASSACT (GIMS)
```

RACF Utilities

RACF provides several utilities that are helpful for the security administrator, auditor and system programmer to execute their tasks easily and efficiently. The RACF utilities are used for maintaining, modifying, copying, unloading, and monitoring the RACF database.

RACF database unload utility program (IRRDBU00)

This utility unloads the RACF database to a sequential file. The sequential file can then be used to run queries. IRRDBU00 process a copy of the RACF database a backup database or the activate database. UPDATE authority required. The output is used as input for IRRRID00 utility.

RACF remove ID utility (IRRRID00)

This utility processes the output of the RACF database unload utility (IRRDBU00) and creates commands to remove references in the RACF database to user IDs and group IDs that are no longer in the database. Alternatively, it can create commands to delete references in the RACF database to specified user IDs and group IDs.

RACF SMF data unload utility program (IRRADU00)

IRRADU00 enables installations to create a sequential file from the SMF security-relevant audit data. The sequential file can be used in several ways: viewed directly, used as input for installation-written programs, and manipulated with sort/merge utilities. It can also be uploaded to a database manager (for example, DB2) to process complex inquiries and create installation-tailored reports.

RACF database initialization utility program (IRRMIN00)

This formats a non-VSAM DASD data set for use as a RACF database. This utility with PARM=NEW is used during installing a new RACF system. This utility with PARM=UPDATE is used when upgrading to a new release of RACF to upgrade the RACF dataset to the latest version of templates.

Data security monitor (DSMON)

Data security monitor produces reports on the status of the security environment at the installation and, in particular, on the status of resources that RACF control. These reports can be used to audit the current status of the installation's system security environment by comparing the actual system characteristics and resource-protection levels with the intended characteristics and levels.

RACF report writer (RACFRW)

This lists the contents of System Management Facilities (SMF) records in a format that is easy to read. The reports can be tailored to select specific SMF records that contain certain kinds of RACF information.

RACF AUDITING

The purpose of a mainframe audit is to provide assurance that policies are being implemented as required, security is strong, and that procedures in place are working and are updated as needed.
A SPECIAL user can execute any RACF command except those reserved for a user with the AUDITOR attribute. This separation of powers is necessary because it is the

security administrator's job to establish RACF controls; it is the auditor's job to test the adequacy and effectiveness of these controls.

Two types of auditing data exist:

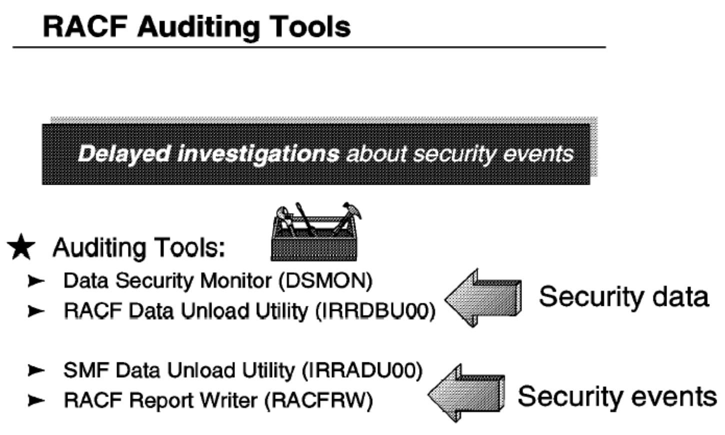
- Security data content from the RACF database, this is a static image or a snapshot of the system parameters at any one time.
- Security events data statistical information, such as the date, time, and the number of times a specific resource was accessed by any one user.

RACF writes security log records when it detects:

- Unauthorized attempts to enter the system
- Authorized or unauthorized attempts to enter RACF commands
- RACF status changes
- Warning mode resource access attempts
- Optional authorized or unauthorized attempts to access RACF-protected resources

There are numerous ways in which to extract information from within the RACF database.

- RACF LIST commands
- RACF SEARCH command
- REXX programs and CLISTs
- RACF Database Unload Utility
- RACF Report Writer
- RACF Data Security Monitor· RACF SMF Data Unload Utility


RACF Auditing Tools
Delayed investigations about security events
Auditing Tools:
Data Security Monitor (DSMON)
RACF Data Unload Utility (IRRDBU00)
Security data
SMF Data Unload Utility (IRRADU00)
RACF Report Writer (RACFRW)
Security events

1. RACF LIST Commands

The profiles in the RACF database contain the information RACF needs to control access to resources. The RACF commands allow you to add, change, delete and list the profiles for users, groups, data sets, and general resources.

The following RACF LIST commands are available:

- **LISTDSD** List the details of one or more discrete or generic DATASET profiles, including the users and groups authorized to access the data set.
- **LISTUSER** List the details of one or more user profiles, including all the groups to which each user is connected.
- **LISTGRP** List the details of one or more group profiles, including the users connected to the group.
- **RLIST** List the details of discrete or generic profiles for one or more resources whose class is defined in the class descriptor table.

2. REXX program

To present this data in a more meaningful way, the RACF administrator had to learn either Assembler and macro programming, or one of the more modern programming/command languages such as REXX (on VM and MVS) or CLISTs (MVS only).

3. RACF Database Unload Utility

- **IRRDBU00**
- This utility reads a RACF database, either the primary or a copy, and creates a sequential data set. This data set can be:
  Sorted or   Loaded into a relational database such as DB2

```
//IRRDBU00 EXEC PGM=IRRDBU00,PARM='NOLOCK'
//SYSPRINT DD  SYSOUT=*
//INDD1      DD  DISP=OLD,DSN=input dsn(database name)
//OUTDD    DD  DISP=OLD,DSN=ouput dsn
//SYSUDUMP DD  SYSOUT=*
```

4. RACF SMF Data Unload Utility

The RACF SMF Data Unload Utility is implemented by SMF Dump Utility (IFASMFDP).

5. RACF Data Security Monitor

DSMON produces the following reports for the following:

- System Report - SYSTEM
- Group Tree Report – RACFGRP
- Program Properties Table Report – SYSPPT
- RACF Class Descriptor Table Report - RACCDT
- RACF Global Access Checking Report - RACGAC

- RACF Started Procedures Table Report - RACSPT
- Selected User Attribute Report – RACUSR
- Selected Data Sets Report – USRDSN

The three DSMON control statements that allow you to control DSMON reporting are:
LINECOUNT,FUNCTION,USEROPT

LINECOUNT number: specifies the number of lines per page for reports. The valid values for number are 0 or a number in the range of 40 through 99.

FUNCTION function-name :specifies the DSMON function or functions you want to include.

The default is ALL, which causes DSMON to generate all reports except USRDSN.

USEROPT : defines user input to be processed by the function you specify.

USRDSN , RACGRP

```
//stepname EXEC PGM=ICHDSM00
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD *
   LINECOUNT 55
   FUNCTION all
```

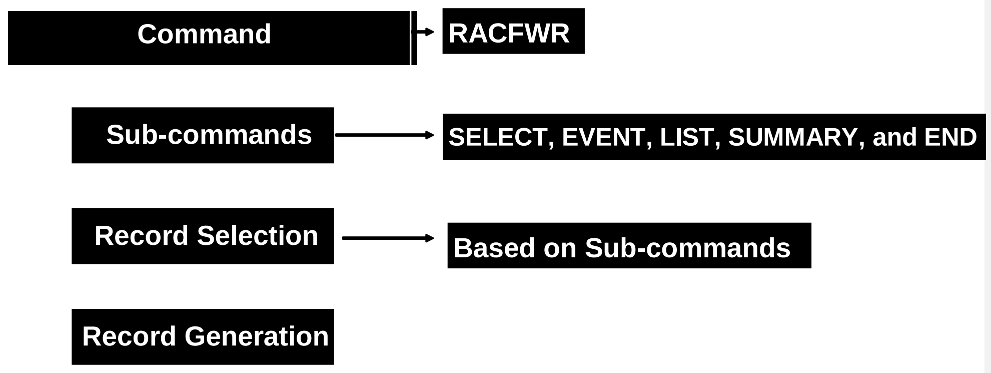
<u>6.RACF Report Writer</u>

The RACF Report Writer (RACFRW) lists the contents of the System Management Facilities (SMF) records in a format that is easy to read.
The report writer is no longer the recommended utility for processing RACF audit records. The RACF SMF data unload utility is the preferred reporting utility.

The RACF report writer consists of three phases:

Command and subcommand processing
Record selection
Report generation

RACF Report Writer



```
//AUDITRAC EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=9
//SYSTSPRT DD SYSOUT=9
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//RSMFIN   DD DSN=your.smf.file, DISP=SHR
//SYSTSIN  DD *,DLM=XX
          RACFRW TITLE ('LIST access for terminal T69000CT')
          SELECT DATE(97001:97351) TERMINAL(T69000CT)
          EVENT LOGON
          LIST SORT(USER DATE TIME)
          END
/*
```

So, these are the methods using which we can extract reports from the mainframe

Exercises

1. To Create User id:

ADDUSER (userid) NAME('123') DFLTGRP(RACFGRP) OWNER(DAVIN3) PASSWORD(ARICH) TSO(ACCTNUM(ACCT#) PROC(IKJDB2) SIZE(4096) MAXSIZE(8212) UNIT(3390))DFLTGRP(grp)

2. To Create a Alias

DEFINE ALIAS(NAME(userid) RELATE(USERCAT.BRANCH))

3.To Alter User Information

To alter the name of the user: **ALU userid name(newname**)
To alter the owner of the user: **ALU userid owner(newowner)**

*To revoke the userid***:** **ALU userid REVOKE**
To resume the user: **ALU userid RESUME**
To resume the user and change pwd: **ALU userid PASSWORD(pwd) RESUME**

3**. To LIst User id :**

LU userid
LU userid CICS

4 To Delete the User id :

DU userid

Profile Group

1.To Create Group:

AG groupname supgrooup(anothergroup) owner(name)

Example: AG racfgrp supgrooup(sys1) OWNER(davin2

2.To alter group:

To alter owner of the groups: Alg groupname owner(newowner)

3. To List group

Lg groupname

4 To Delete Group:

DG groupname

Profile Connect

1.To Connect One User to another Group

*To connect the user to new group***: Connect userid group(newgrp)**
To alter the users default group: **Alu userid DFLTGRP(newgrp)**
To remove user from the old grp: **Remove userid group(oldgrp)**

Profile Dataset

1.To Protect the dataset:

To define dataset to racf with universal access none

ADDSD ‘userid.**’ UACC(NONE)

To permit dataset to other with access read or update

PERMIT ‘userid.**’ ACC(READ/UPDATE) ID(otheruserid)

Example:

PERMIT ‘syspm01.**’ ACC(PDATE) id(mfsp01)

2. Alter dataset protection:

ALTDSD ‘userid.**’ access(none) id(otheruserid)

Example:
ALTDSD ‘syspm01.**’ ACC(READ) id(mfsp01)

3. List dataset protection

LISTDSD ‘userid.**’
LISTDSD ALL ID(userid)

4. TO Delete Dataset Protection

DELDSD ‘userid.**’ NOTIFY( )

Profile General Resources

1.To list Racf DataBase

RVARY

2.To Switch Racf DataBase

RVARY SWITCH

3.To list all classes

SETR LIST

4. To list one particular class and its profile names

RLIST classname *

5.To list a particular profile names

RLIST classname profilename

6.To Define a class to racf i.e change it from Generic to Active class

To protect the class by giving universal access none

RDEFINE classname profilename UACC(NONE)

To permit the class by giving access read

PERMIT profilename CL(classname) ID(userid) ACCESS(READ)

7. To activate the class and not activating the class

SETR CLASSACT ‘classname’
SETR NOCLASSACT ‘classname’

8.To refresh the class after making changes

SETR RACLIST (classname) REFRESH

9. To delete class proctection

RDELETE classname profilename

10.To revoke User ids after 4 attempts of wrong password and updating the statistics about it

```
SETR PASSWORD(REVOKE(4))
SETR INITSTATS
```

11.To revoke User ids after 60 days of no logon & updating the statistics about it

```
SETR PASSWORD(INTERVAL(60))
SETR INITSTATS
```

12. To logon even after unlimited wrong password entry

```
SETR PASSWORD(norevoke)
```

COMMUNICATION SERVER-TCPIP,VTAM & UNIX

What is a Network?

A **computer network** is a system for communication among two or more Computers. These networks may be fixed (cabled, permanent) or temporary (as via modems).

- The need of network evoluted because of some economic and technological reasons i.e. they sum up like if sufficiently large and powerful mainframes were available at acceptable prices, most companies would simply have afforded to keep all the data in them.
- Hence the computer networks became more popular because of its huge price/performance advantage over mainframes.

Network Hardware

The technical issues mainly focuses on

- Transmission Technology
- Geographical Disposition

Transmission technology

Broadly speaking there are two types they are

- Broad-cast network
- Point-to-Point network

Broadcast networks have a single communication channel that is shared by all the machines on the network. Short messages, called packets in certain contexts, sent by any machines are received by all the others.

Contrast point-point networks consist of many connections between individual pairs of machines. To go from the source to the destination, a packet may be routed thru one or more paths.

Geographical Disposition

Broadly classifying this as
LAN(Local Area Network)
MAN(Metropolitan Area Network)
WAN(Wide Area Network)
Internetwork(Network of networks)

Network Software

Coming into the high-end part of the network, lets start out with Protocol hierarchies:

- Protocol generally focuses the rules and conventions used in communication between nodes of a network.
- It determines how a computer node functions during communication with another node, how data is enclosed to reach its destination safely & what path it should follow.

TERMINOLOGY

Host: In the Internet suite of networks, it is the high end system .i.e an individual computer on a network. It can just be any workstation.

Subnet: It is the part of network used to carry messages from host to host.It has two distinct components**,**

- Transmission Lines
- Switching Elements

Transmission lines are simply the channels or circuits.
Switching elements are specialized computers used to connect two or more transmission lines. When data arrive on the incoming lines, the elements must choose an outgoing line to forward it. It can be packet switching or data switching.

Gateway: A functional unit that interconnects two networks or systems with different architecture.

Port: Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number,used by the host-to-host protocol to identify to which higher level protocol or app program it must deliver incoming messages. There are two types of ports

i. Well-known port (standard server)
ii. Ephemeral (clients)

Socket: An endpoint for communication between processes or application programs.

Socket Address: The address of an application program that uses socket interface on the network. In internet format, it consists of the IP address of the socket's host and the port number of the socket.

Router : A router interconnects networks at the internetwork layer level and routes packets between them. The router must understand the addressing structure associated with the networking protocols it supports and make decisions on whether, or how, to forward packets.

Layered structure

- To reduce the design complexity, most networks are organized as a series of Layers or Levels, each built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network.
- The entities comprising the corresponding layers on different machines are called **Peers.**
- Below layer-1 is the **physical medium** through which actual communication occurs.
- Between each pair of adjacent layers there is an **interface**
- A set of layers and protocols is called a **architecture.**
- A list of protocols used by a certain system, one protocol per layer, is called a **protocol stack.**

Layers can offer two different types of service to the layers above them:

- Connection-Oriented
- Connectionless

Connection-Oriented network Service is modeled after the telephone system. i.e. the service user first establishes a connection, uses the connection, and then releases the connection. That is it acts like a tube the sender pushes objects (bits) in at one end, and the receiver takes them out in the same order at the other end.

Connectionless service is modeled after the postal system.Each message carries the full destination address, and each one is routed through the system independent of all the others.

Reference models OSI:
This model is based on a proposal developed by the ISO (International Standards Organization) as a first step toward international standardization of the protocols. It mainly deals with connecting open systems i.e. systems that are open for communication with other systems.

Class A addresses use 7 bits for the networkand 24 bits for the hostportion of the IP address. That allows for 126 (2**7-2) networks with 16777214 (2**24-2) hosts each; a total of over 2 billion addresses.

Class B addresses use 14 bits for the networkand 16 bits for the host portion of the IP address. That allows for 16382 (2**14-2) networks with 65534 (2**16-2) hosts each; a total of over 1 billion addresses.

Class C addresses use 21 bits for the networkand 8 bits for the hostportion of the IP address. That allows for 2097150 (2**21-2) networks with 254 (2**8-2) hosts each; a total of over half a billion addresses.

Class D addresses are reserved for multicasting (a sort of broadcasting, but in a limited area, and only to hosts using the same class D address).

Class E addresses are reserved for future use.

The PING command is very simple:

PING IP-address|symbolic_name

You can issue this command from all TCP/IP hosts, and it will tell you whether you can communicate with the target host.

TCPIP OVERVIEW

TCP/IP is a set of protocols that allows communication between networks and computers of networks. It is a component of eNetwork communication server.Provides interconnection of networks.Independent of the types of computers used in the network.

TCP/IP LAYERS:

Unlike the OSI reference model, this TCP/IP reference model is a four layered structure. They are

- PHYSICAL LAYER
- INTERNET LAYER
- TRANSPORT LAYER
- APPLICATION LAYER

THE PHYSICAL LAYER:

- This is the lowest layer and is not actually specified by TCP/IP.
- This is actually responsible for the physical transmission of real data i.e. 0 as 0 and not 1.
- The user can access any physical transmission approach including WAN & LAN.
- The common architecture would be of Ethernet , X.25 or etc

THE INTERNET LAYER:

- This provides the routing of datagrams(messages) from one computer to another i.e. can be from one local device to another or it could be across a wide area n/w.
- INTERNET PROTOCOL does the major role of this.
- This is performed thru the unique IP address which every TCP/IP host has.
- IP address is a combination of network ID that is assigned centrally & the local address that is locally administrated.

TRANSPORT LAYER:

- This supports reliable transmission of data.
- It allows the peer entities on the source & destination hosts to carry on a conversation.
- The two major protocols here are TCP & UDP

TCP:

- It allows a byte stream originating at one end to be delivered in the other end.
- It does the fragmentation at one end & the reassembling at the other end.
- This mainly is responsible for flow control.

UDP:

- When TCP's flow control has to be overridden by the user's own flow-control, this UDP is used.
- Its like one-shot ,client-server type request and reply applications in which prompt delivery is more important than accurate delivery. Ex: speech or video

APPLICATION LAYER:

- It contains all the higher level protocols that the users can invoke to access network services.
- All are not mandatory, certain applications are implemented in nearly all TCP/IP implementations Ex: TELNET, SMTP & FTP.

FUNCTIONS OF eNetwork Server:

The network for OS/390 which can access OS/390 applications and data over TCP/IP.

- Log-on to remote host
- Transfer datasets
- Email services
- Print on remote printers
- Authenticate new users
- Monitor the network

TCP/IP PROCEDURE:

```
//TCPIP         PROC PARMS = ‘CTRACE(CTIEZB00)’
//TCPIP         EXEC   PGM = EZBTCPIP, REGION = 0M, TIME =1440
//STEPLIB     DD *
//SYSPRINT  DD  SYSOUT=*,DCB =(RECFM =VB,LRECL=80)
//ALGPRINT DD  SYSOUT=*,DCB =(RECFM =VB,LRECL=80)
//CFGPRINT DD  SYSOUT=*,DCB =(RECFM =VB,LRECL=80)
//CEEDUMP DD  SYSOUT=*,DCB =(RECFM =VB,LRECL=80)
//PROFILE    DD  DISP=SHR,DSN=TCPIVP.TCPPARMS(PROFILE1)
//SYSTCPD   DD DISP=SHR,DSN=TCPIVP.TCPPARMS(TCPDATA)
//
```

TCPIP PROC PARMS = 'CTRACE(CTIEZBxx)' This is the first statement of the proc, as in the case of all procedures, it has a procname which should be the same as the member name of the CPAC.PROCLIB, where it resides. PARMS is the parameters to be passed to the procedure which does the CTRACE that is the component tracing of the TCP/IP products in the case of a Sysplex environment. The CTRACE member CTIEZB00 is a parmlib member which contains the info about the components to be traced.

TCPIP EXEC PGM = EZBTCPIP This is the execution step where the load module is called to execute the parameters in the TCP/IP initialization datasets
STEPLIB DD * The C runtime libraries should be in the system's link list or add them via a STEPLIB statement definition & the added libs should be APF authorized
SYSPRINT This contains the run-time diagnostics from TCP/IP.It can be either directed to a dataset or the SPOOL.But for space constraints, this is forwarded to SPOOL.
ALGPRINT This contains run-time diagnostics from the TCP/IP autolog task.Autolog means the start and the time value to start or terminate for the subservers.
CFGPRINT This contains the run-time diagnostics about the TCPIP statistics.
SYSERROR This contains the error messages from TCP/IP that occurred while processing PROFILE dataset.
CEEDUMP This is strongly recommended since the TCP/IP config component is now written in C. If TCP/IP terminates abnormally, a dump occurs.
PROFILE DDDD statements which specifies location of PROFILE ds can could be found
SYSTCPD DD This explicitly identifies the data set used to obtain parameters defined by TCPIP.DATA.

PROFILE datasets: This is used to configure TCP/IP address space.

- During initialization of the TCP/IP stack, system Operation & configuration parameters for the TCP/IP are read from the configure file Profile.TCPIP.
- The location of this profile datasets can be explicity found from DD statement of the PROC.
- In the case of dynamic start up of TCP/IP ,the search has a specific order.
- In order to customise your system,specify system operation parameters & network configuration info in the dataset using the configure statements & save it as another member.
- Some of the commands can also be put in another datsets and can be processed with VARY TCP/IP commands & dynamically change the TCP/IP configure.
- Even system symbols can be used in the Profile datasets ,thus several TCPIP sys can share one PROFILE.TCPIP reducing the number of dataset to be present.

EFFECTS ON OTHER MEMBERS

IEAAPFxx/PROGxx : To add the needed C libraries Ex : hlq.sezalink, hlq,sezatcp in the LNKLSTxx and LPALSTxx.

SCHEDxx : To set PPT entries - PPT PGMNAME(EZBTCPIP) KEY(6) NOCANCEL NOSWAP PRIV SYST

COMMNDxx:	Command to start the SSI
IFAPRDxx:	Enables the TCPIP base by adding the IP
BPXPRMxx:	To activate TCPIP support for transport provider.

TCP/IP applications

TN3720 TELNET is most often used as the primary method of connection between client workstations and the SNA mainframe environment. TELNET terminal emulation needs to simulate actual SNA terminals as closely as possible to make this form of remote connection as seamless as possible to end users. RFC1647, also known as TN3270E, adds the ability to simulate specific terminal LU connection and support printer devices and additional SNA functions.

Telnet The telnet support comes with the TCP/IP OS/390 UNIX feature. It also uses the inetd daemon which must be active and set up to recognize and receive the incoming telnet requests. OS/390 UNIX telnet code is installed in the hierarchical file system (HFS) (path /usr/lpp/tcpip/sbin/otelnetd with a symbolic link to /usr/sbin/otelnetd) and in the MVS data set hlq.SEZALINK. The hlq.SEZALINK data set needs to be a PADS protected data set if you are running with the BPX.DAEMON facility class defined. OS/390 UNIX checks whether the sticky bit is set on in the HFS. If it finds the sticky bit on, it first checks for an executable file in the MVS data set. If it does not find the executable file in a data set in the MVS search order, OS/390 UNIX then uses the executable file in the HFS.

rlogin When the inetd daemon is set up and active, you can rlogin to the shell from a workstation that has rlogin client support and is connected via TCP/IP or Communications Server to the MVS system. To login, use the rlogin (remote log in) command syntax supported at your site. The inetd daemon provides service management for a network. For example it starts the rlogind program whenever there is a remote login request from a workstation.
The rlogind program is the server for the remote login command rlogin, commonly found on UNIX systems.

3270

It is the session protocol used to establish a screen connection to a System/390 mainframe.

It started life as the traditional 24x80 green screen and developed into the more usable, but still essentially dumb screens or terminal emulators we use today:

The 3270 model 2 24x80
The 3270 model 3 32x80
The 3270 model 4 43x80
The 3270 model 5 27x132
Extended Attribute Support (colors, blink, etc.)

TN3270 parms

The TELNET server gets some of its configuration parameters through the TELNETPARMS statements. If you want to specify parameters for the TELNET server, update the TELNETPARMS section of the TCP/IP Profile data set.

```
TELNET parms
TELNETPARMS
PORT 623
WLMCLUSTERNAME TN3270E ENDWLMCLUSTERNAME
ENDTELNETPARMS
```

The TELNETDEVICE statement lets you specify a logmode for a device type anywhere within the BEGINVTAM/ENDVTAM block, instead of just at the beginning of the block. This statement accepts two logmodes: one for 3270 connections and one for 3270E connections.

FTP

FTP File Transfer Protocol (FTP) lets you transfer data sets between the local host and any other host that supports TCP/IP. Using the FTP command and its subcommands, you can sequentially access multiple hosts without leaving the FTP environment.
File Transfer Protocol (FTP) is the simplest way to transfer files between computers connected by a TCP/IP network. The direction of the file transfer can be either PUSH (send a file to the remote host) or PULL (get a file from the remote host). The local and remote hardware and software are largely irrelevant, and you can select to transfer files in ASCII, BINARY, or EBCDIC format from any machine to any other machine. For example, a flat text file can be transferred in ASCII format from a PC workstation to VM/ESA, or OS/390, or UNIX and its contents will still be readable as a flat text file when it arrives at its destination. A compressed (zipped) PC file can be transferred in binary format to an OS/390 machine for storage and then transferred from there to a different PC where it will arrive intact and can be successfully decompressed (or unzipped).FTP is a client-server protocol where the initiating side of the connection is the FTP client and the client connects to an FTP server. FTP provides various commands to connect to a remote host, change the active directory, define the data transfer format, and obviously, send and receive files.
A few of the more useful ones are:

AScii	Set transfer mode to text
Binary	Set transfer mode to binary
CD	Change active Directory on remote side
CLose	Close connection
DIR	Get directory listing of remote files
Get	Retrieve files from remote host
MGet	Multiple Get for multiple files
MPut	Multiple Put for multiple files
Open	Initiate connection to remote site
PASS	Send password to remote host
Put	Send file to remote host
PWD	Query Preset Working Directory on remote host
QUIT	Close connection to remote host, or EXIT ftp if no connection active
SYStem	Query the remote operating system type
USer	Initiate a logon to remote host once connection is established
UDP	User Datagram Protocol
SPX	Sequence Packet Exchange Protocol
IPX	Internetwork Packet Exchange
HDLC	High Level DataLink Control
EIA	Electronci Industries Alliance
TIA	Telecommuncaition Industry Association
PDU	Protocol Data Unit
SNMP	Simple Network Management Protocol
DHCP	Dynamic Host Configuration Protocol

Systems Network Architecture (SNA)

Systems Network Architecture (SNA) is IBM's proprietary network architecture and set of implementing products for network. With SNA , a mainframe computer running Advanced Communications Function/Virtual Telecommunication Access Method (ACF/VTAM).

- (SNA) networking protocol including basic connectivity such as 3270, SDLC(Synchronous Data Link CONTROL)
- PHYSICAL - hosts, communications controllers, establishment controllers, and terminals
- Data link control (DLC)—Defines several protocols, including the Synchronous Data Link Control (SDLC) protocol for hierarchical communication, and the Token Ring Network communication protocol
- Path control—Performs many OSI network layer functions, including routing and datagram segmentation and reassembly.

SNA defines three essential network addressable units,

- Logical units (LUs) function as end-user access ports into an SNA network. LUs provide users with access to network resources, and they manage the transmission of information between end users.
- Physical units (PUs) are used to monitor and control attached to network links
- Control points (CPs) manage SNA nodes and their resources. CPs generally are differentiated from PUs in that CPs determine which actions must be taken, while PUs cause actions to occur.
- Transmission control—Provides a reliable end-to-end connection service, as well as encrypting and decrypting services.
- Data flow control—Manages request and response processing, determines .
- Presentation services —Specifies data-transformation ,coordinate resource sharing, and synchronize transaction operations
- Transaction services —Provides application services in the form of programs that implement distributed processing or management services

VIRTUAL TELECOMMUNICATION ACCESS METHOD (VTAM)

It is the system software which controls all telephone and communication connections to a mainframe computer. VTAM provides a method by which application programs can communicate with telecommunication devices . VTAM was the first IBM program to allow programmers to deal with devices as “logical units” without having to understand the details of line protocols and device operation. Prior to VTAM, programmers used IBM’s Basic Telecommunications Access Method (BTAM) to communicate with devices that used the binary synchronous (BSC) and start-stop line protocols. Every terminal connected to the computer is controlled by VTAM. No terminal can be used by an MVS computer system unless it is defined to VTAM.

The VTAM host feature permits SNA-to-SNA communication over a Transmission Control Protocol/Internet Protocol (TCP/IP) network. The specific configurations it supports include the following:

- VTAM-to-VTAM communication over a single IP network
- VTAM-to-Communications Manager/2 communication over a single IP network
- VTAM-to-AIX SNA server communication over a single IP network
- VTAM as a multiprotocal transport networking (MPTN) gateway between an SNA network and an IP network

The following must be defined for SNA access to the IP network:

- **A TCP/IP major node**
- **A CDRSC definition for the destination LU in the IP network.**

VTAM Functions:

- Monitors and Controls the activation and connection of resources.
- Establishes connections & manages the flow and Pacing of sessions.
- Provides Application Program Interface(API),that allows access to the user written application program's & IBM provided subsystems.
- Provides Interactive Terminal support for TSO & MVS
- Provides support for both locally & remotely attached resources

VTAM maintains definitions of the terminals which can connect to the mainframe, and of the programs which terminals can be connected to. The VTAM system programmer defines all the terminals in a dataset usually called SYS1.VTAMLST and CPAC.VTAMLST.

Library files **– SYS1.VTAMLIB AND CPAC.VTAMLIB** .

UNIX System Services

When OS/390 was renamed to z/OS, the new abbreviation for UNIX System Services became z/OS UNIX.

UNIX Layers

Three main parts of OS Kernel,Shell & File Structure

<u>Kernel</u>

It is Core of the Operating System. It Manages Devices, Memory, Processes & Daemons. It Controls the functions between the system programs and the hardware.

<u>Shell</u>

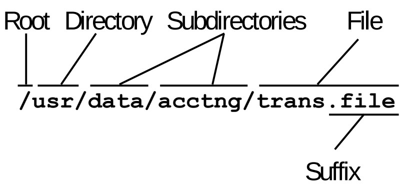
Interface between the user and the kernel, Acts as an interpreter,Accepts commands, interprets and sends the executable to the kernel. Examples Bourne,C,ZSH

All files within a hierarchical file system (HFS) are members of a directory. Each directory is in turn a member of another directory at a higher level in the hierarchy. At the highest level of the hierarchy is the root directory.
The terms hierarchical file system and HFS are used both for the UNIX file structure in general and for the specific facility that OS/390 uses to manage the files on an IBM mainframe.

HFS files can contain data, executable modules or UNIX shell scripts that consist of UNIX commands and scripting commands. They can also be copied to and from OS/390 sequential, partitioned or partitioned extended data sets.
An HFS file name or directory name can contain

- Up to 255 characters
- Uppercase and lowercase A-Z
- Digits 0 to 9
- Periods (.), underscores (_), and hyphens (-)
- No nulls or slashes

Elements of Path name



TSO commands working with HFS files and directories

MKDIR	Creates an HFS directory. Any intermediate directories in the pathname you specify must already exist.
OBROWSE	Browses an HFS file using the ISPF browse facility.
OEDIT	Creates or edits an HFS file using the ISPF editor.
OGET	Copies an HFS file to an OS/390 sequential data set or to a PDS or PDSE member.
OGETX	Copies one or more HFS files from a directory to a PDS, a PDSE, or a sequential data set.
OPUT	Copies an OS/390 sequential file or a PDS or PDSE member to an HFS ‘ file.
OPUTX	Copies one or more members from a PDS, a PDSE, or a sequential data set to a directory. ‘

Z/OS and Unix Operating System

The z/OS support for UNIX System Services (z/OS UNIX) enables two open systems interfaces on the z/OS operating system. They are

1. **An application program interface (API)**
2. **An interactive z/OS shell interface**

API programs can request

Only MVS services, Only z/OS UNIX and Both MVS and z/OS UNIX

Shell interface

It is an execution environment analogous to TSO/E, with a programming language of shell commands.
The shell work consists of:

- Programs run by shell users
- Shell commands and scripts run by shell users
- Shell commands and scripts run as batch jobs

To run a shell command or utility, or any user-provided application program written in C or C++, you need the C/C++ runtime library provided with **the Language Environment (LE).**

z/OS UNIX System Services Application Services Users can request services from the system through shell commands. Write shell scripts to run tasks. Run programs interactively (in the foreground) or in the background.
Application programmers are likely to do the following when creating UNIX-compliant application programs:

- Design, code and test programs on their workstations using XPG4 UNIX-conforming systems.
- Send the source modules from the workstation to z/OS.
- Copy the source modules from the MVS data sets to HFS files.
- Compile the source modules and link-edit them into executable programs.
- Test the application programs.
- Use the application programs.

Following types of applications exist in a z/OS system with z/OS UNIX:

- Applications using only kernel services
- Applications using both kernel and z/OS services
- Applications using only z/OS services

A z/OS program submitted through the job stream or as a job from a TSO/E session can request kernel services through the following:

- C/C++ functions
- Shell commands, after invoking the shell
- Callable services

Product and component support for z/OS UNIX

- TCP/IP, RACF,DFSMS,VLF,SMF,SMP/E,TSO/E,TSM,RMF,WLM

The z/OS UNIX kernel

At system IPL time, kernel services are started automatically. The kernel provides z/OS UNIX System Services in answer to requests from programs and the shell. The kernel manages the file system, the communications, and the program processes. The hierarchical file system is shipped as part of DFSMS. zFS is shipped with the Distributed File Service. A typical UNIX operating system consists of a kernel which interfaces directly with the hardware. Built on the kernel is the shell and utilities layer that defines a command interface. Then there are application programs built on the shell and utilities. In a z/OS UNIX environment the file system is considered part of the kernel because it is allocated to the kernel. The support for the file system is provided by the DFSMS product.

To support the APIs, the z/OS system must provide some system services which are included in the kernel, such as the file system and communication services. Daemons are programs that are typically started when the operating system is initialized and remain active to perform standard services. Some programs are considered daemons that initialize processes for users even though these daemons are not long-running processes. z/OS UNIX supplies daemons that start applications and start a user shell session when requested.

The z/OS UNIX shell and utilities.

An interactive interface to z/OS UNIX services which interprets commands from interactive users and programs. The shell and utilities component can be compared to the TSO function in z/OS. The z/OS UNIX debugger (dbx). It is a tool which application programmers can use to interactively debug a C program. The C/C++ compiler and C run-time libraries are needed to make executables that the kernel understands and can manage.

DFSMS

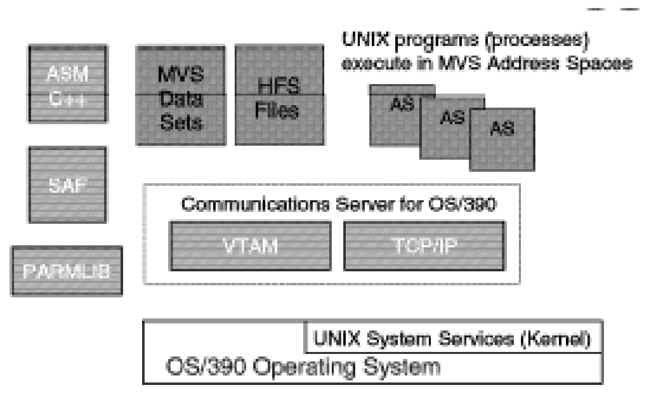
DFSMS manages the hierarchical file system (HFS) data sets that contain the file systems. To use kernel services in full-function mode, SMS must be active. Network File System (NFS) enables users to mount file systems from other systems so that the files appear to be locally mounted.

Language Environment (LE)

The C compiler and Language Environment feature library are changed and extended to include support for the POSIX and XPG4 C function calls. The LE product provides a common run-time environment and language-specific run-time services for compiled programs. To run a shell command or utility, or any user-provided application program written in C or C++, you need the C/C++ runtime library provided with language environment.

zFS

The zSeries File System (zFS) is a new UNIX file system that can be used in addition to the hierarchical file system.


ASM
C++
MVS Data Sets
HFS Files
UNIX programs (processes) execute in MVS Address Spaces
AS
AS
AS
SAF
Communications Server for OS/390
VTAM
TCP/IP
PARMLIB
UNIX System Services (Kernel)
OS/390 Operating System

A process is a program using kernel services. The program can be created by a fork() function, fork callable service or spawn() function or the program can be dubbed because it requested kernel services. The three types of processes are User processes, which are associated with a program or a shell user Daemon processes, which perform continuous or periodic system-wide functions, such as printer spooling Kernel processes, which perform system-wide functions for the kernel such as cleaning up processes (init process).

When you enter a shell command, you start a process that runs in an MVS address space. When you enter that command, the z/OS shell runs it in its own process group. As such, it is considered a separate job and the shell assigns it a job identifier a small number known only to the shell. A shell job identifier identifies a shell job, not an MVS job. When the process completes, the system displays the shell prompt.

BPXPRMxx -This parmlib member determines the number of processes that may be started in the z/OS system. MVS data sets UNIX programs may access MVS data sets. HFS files UNIX programs may access HFS files.

TCP/IP Workstation users can enter the shell environment by using rlogin or telnet in a TCP/IP network. User-written applications can use TCP/IP as a communication vehicle. VTAM Workstation users can access TSO/E through VTAM. z/OS UNIX is then accessed from TSO/E.

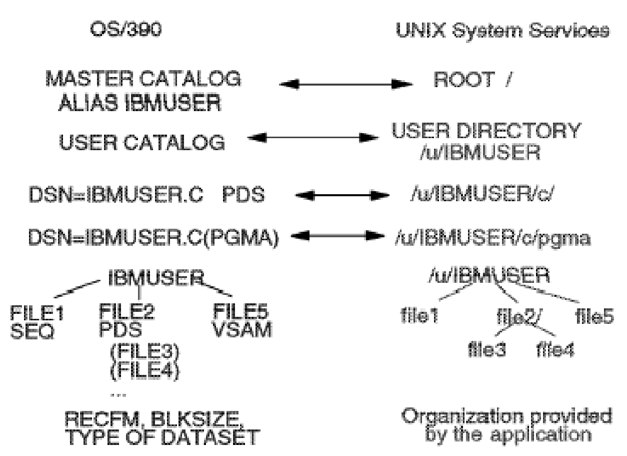
A process will be created by using any of the following methods:

fork() function: Creates a child process which is identical to the parent process in a new address space scheduled by the Workload Manager (WLM).
spawn() function: Creates a child process which will execute a different program than the parent, either in a new address space scheduled by WLM or in the same address space as the parent process.

z/OS UNIX callable services: When a program uses a z/OS UNIX assembler callable service, the z/OS address space will be *dubbed* a z/OS UNIX process. The address space will get a PID. The dub will not result in a new address space. The kernel interfaces to WLM to create the new address space for a fork or spawn. WLM uses an IBM-supplied procedure BPXAS to start up a new address space. This new address space will then initialize the UNIX child process to run in the address space. After the child process completes, this address space can be reused for another fork or spawn. If none is waiting, BPXAS will time out after being idle for 30 minutes.

MVS data sets versus file system files

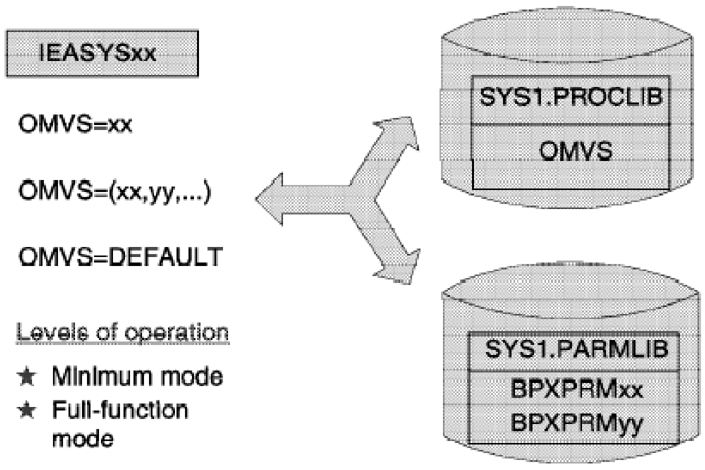
Interactive users of z/OS UNIX have a choice between using a UNIX-like interface (the shell), a TSO interface and an ISPF interface. With these choices, users can choose the interface which they are most familiar with and get a quicker start on z/OS UNIX. The z/OS UNIX shell provides the environment that has the most functions and capabilities.



CPAC.PARMLIB MEMBER

In **IEASysXX** We have to specify **omvs = xx** it invoke BPXPRMxx
omvs = fs it invoke BPXPRMFS

BPXPRMXX specifies system limits by the parameters that control the z/os unix environment. MVS system uses these values when initializing UNIX system services.
BPXPRMFS specifies file system setup and reflects the restored file system.



The initial parmlib settings for the z/OS UNIX kernel are pointed to by the OMVS parameter in the IEASYSxx parmlib member. The OMVS parameter in the IEASYSxx parmlib member lets you specify one or more BPXPRMxx parmlib members to be used to specify the initial parmlib settings for the kernel. If you do not specify the OMVS parameter or if you specify OMVS=DEFAULT, the kernel is started in a minimum configuration mode with all BPXPRMxx parmlib statements taking their default values. OMVS may also be left out or coded as DEFAULT. This allows the z/OS UNIX kernel to start in a minimum configuration. All BPXPRMxx values will take their default values and a temporary root file system will be set up in memory.
Activation of kernel services is available in two modes:

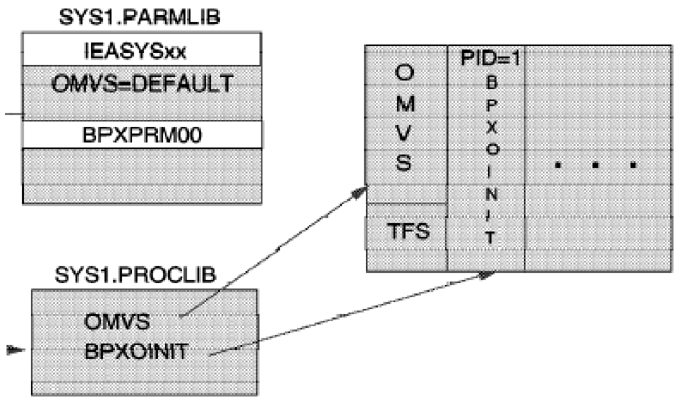
Minimum mode, Full-function mode

You can set up kernel services in either minimum mode or full function mode. If you want to use any z/OS UNIX service, TCP/IP or other functions that require the kernel services, you will need to use full function mode, otherwise, you can use minimum mode. In order to apply service to the HFS, you need at least one system that can run in full function mode. The start and stop commands for the z/OS UNIX kernel are no longer supported

z/OS UNIX minimum mode

If omvs=null or omvs=default,kernel services start in minimum mode.
Minimum mode is intended for installations that do not intend to use z/OS UNIX System Services, but which allow the IPL process to complete. In this mode many services are available to programs. Some that require further customization, such

as a fork(), will fail. If you do not specify OMVS= in the IEASYSxx PARMLIB member or if you specify OMVS=DEFAULT, then kernel services start up in minimum mode when the system is IPLed. This mode is intended for installations that do not plan to use the kernel services. In minimummode, Many services are available, but some functions such as TCP/IP sockets that require other system customization may not work.



TCP/IP sockets (AF_INET) are not available. A temporary file system (TFS) is used. A TFS is an in-storage file system, hence no physical DASD data set needs to exist or be mounted. A temporary file system (kept in memory) is created for the root. The required directories (/bin, /etc, /tmp, /dev, and /dev/null) are built. There are no executables in this file system.

A temporary file system is an in-memory file system which delivers high-speed I/O. If the kernel is started in minimum setup mode, the TFS is automatically mounted.

The system is in minimum mode when: OMVS=Default

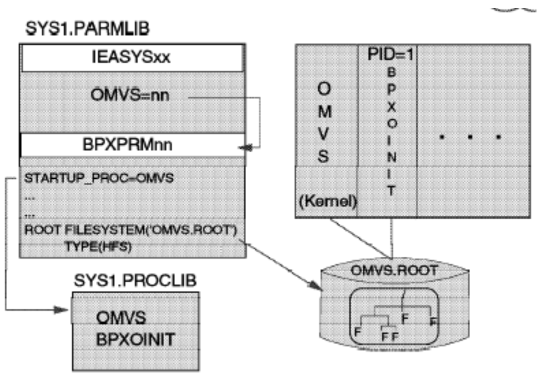
There is no OMVS Statement in IEASYSxx (no BPXPRMxx member in the PARMLIB) A temporary file system is used as the root file system. The temporary file system is initialized and primed with a minimum set of files and directories, specifically the following:

/ (root directory),/bin directory,/etc directory,/tmp directory,/dev directory,/dev/null file

There are no executables in the temporary file system (that is, you will not find the Shell and Utilities). Do not attempt to install z/OS UNIX System Services Application Services in the TFS, since no data will be written to DASD. Because the TFS is a temporary file system, unmounting it causes all data stored in the file system to be discarded. Any data written to this file system is not written to DASD.

z/OS UNIX full-function mode

If omvs=xx or yy of fs,kernel services will start in full-function mode. Full-function mode is started at IPL time when the OMVS parameter in the IEASYSxx parmlib member points to one or more BPXPRMxx parmlib members. There must be a root HFS built and defined in BPXPRMxx for OMVS to initialize correctly, and SMS, WLM, and RACF customization should be completed.



BPXPRMnn

STARTUP_PROC is a 1 to 8 character name of a started procedure JCL that initializes the z/OS UNIX kernel. The default is OMVS. The ROOT statement defines and mounts the root file system. In this example HFS is the TYPE of the file system. OMVS.ROOT is the file system, which is the name of an already defined HFS data set. The root file system has a default of read/write mode. If an active BPXPRMxx PARMLIB member specifies "FILESYSTEM TYPE(HFS)", then the kernel services start up in full-function mode when the system is IPLed. To use the full function, you need to:

- Set up BPXPRMxx PARMLIB definitions
- Set up DFSMS
- Set up the hierarchical file systems
- Set up the security product definitions for z/OS UNIX
- Set up the users' access and their individual file systems

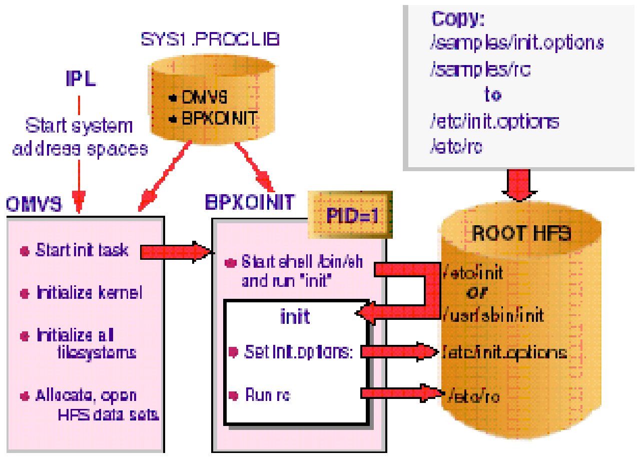
DFSMS manages the HFS data sets that contain the file systems.

BPXOINIT: As of OS/390 V2R3, BPXOINIT is the started procedure that runs the initialization process. BPXOINIT is also the jobname of the initialization process. BPXOINIT is shipped in SYS1.PROCLIB. At system IPL time, kernel services are started automatically. If the OMVS parameter in theIEASYSxx parmlib parameter is not specified, the kernel services are started in minimum mode. If the OMVS parameter specifies one or more BPXPRMxx parmlib members, they are all used to configure the kernel services when the system is IPLed.

The BPXOINIT address space has two categories of functions:

It behaves as PID(1) of a typical UNIX system. This is the parent of /etc/rc, and it inherits orphaned children so that their processes get cleaned up using normal code in the kerne. This task is also the parent of any MVS address space that is dubbed and not created by fork or spawn. Therefore, TSO/E commands, batch jobs, and so forth have a parent PID of

Initializing z/OS UNIX



When z/OS UNIX is started, initialization includes running the /etc/init program. The file/etc/init is referred to as the initialization program that is run when the z/OS UNIX component is started, even though the /usr/sbin/init file may really be run. z/OS UNIX attempts to run the program /etc/init. If no such program is found, z/OS UNIX attempts to run /usr/sbin/init. This file contains the default initialization program shipped with the z/OS UNIX Shell and Utilities.

The /etc/init program invokes a shell to execute an initialization shell script that customizesthe environment for shell users. When this shell script finishes or a time interval established by /etc/init expires, z/OS UNIX becomes available for general batch and interactive use.

Beginning with OS/390 V2R3, an option to use a REXX exec in an MVS data set was provided as an alternative to writing a customized /etc/init initialization program. To activate the REXX exec for initialization, you must specify its name on the STARTUP_EXEC statement in the BPXPRMxx parmlib member.

The /usr/sbin/init program invokes a shell to execute an initialization shell script that customizes the environment for shell users. When this shell script finishes or when a time interval established by /usr/sbin/init expires, kernel services become available for general batch and interactive use.

RACF OMVS segments

All users and programs that need access to OS/390 UNIX System Services must have a RACF user profile defined with an OMVS segment which has as a minimum a UID specified. A user without a UID cannot access OS/390 UNIX. The RACF user profile has a segment called OMVS for OS/390 UNIX support. A user ID must have an OMVS segment defined in order to use OS/390 UNIX System Services, for example access the ISHELL or the shell.

This segment has three fields:

UID A number from 0 to 2147483647 that identifies an OS/390 UNIX user. An OS/390 UNIX user must have a UID defined.

HOME The name of a directory in the file system. This directory is called the home directory and becomes the current directory when the user accesses OS/390 UNIX. This field is optional.

PROGRAM The name of a program. This is the program that will be started for the user when the user begins an OS/390 UNIX session. Usually this is the program name for the OS/390 UNIX shell. This field is optional. The RACF group also has a segment called OMVS to define OS/390 UNIX groups. It contains only one field:

GID A number from 0 to 2147483647 which identifies an OS/390 UNIX group. The home directory is the current directory when a user invokes OS/390 UNIX. During OS/390 UNIX processing this can be changed temporarily by using the cd (change directory) shell command. The command will not change the value in the RACF profile. The directory specified as home directory in the RACF profile must exist (be pre-allocated) before a user can invoke OS/390 UNIX. If a home directory is not specified in RACF, the root (/) directory will be used as default.
z/OS UNIX has no predefined numbering scheme for UIDs and GIDs, with the exception of UID=0. UID=0 is a superuser (also known as ROOT on other UNIX platforms) and has the ability to bypass all forms of security within the UNIX environment. A superuser is a user with a UID of 0 and this UID is in the RACF profile of the user. The procedure must have a UID. However, when a started procedure is trusted or privileged, RACF does not base authority checking on the UID value; the UID can have any value.
A superuser can do the following:

- Pass all security checks, so that the superuser can access any file in the file system.
- Perform the mount function.
- Change identity from one UID to another.
- Manage processes.

- Have an unlimited number of processes running concurrently. For a started procedure, this is true only if it has a UID of 0. It is not true for a trusted or privileged process with a different UID.
- Change identity from one UID to another.
- Use setrlimit to increase any of the system limits for a process.

Define a new user SIVA

ADDUSER SIVA OMVS(UID(15) HOME('/u/smith')PROGRAM('/bin/sh'))

Add OMVS segment to existing group SYSADM

ALTGROUP SYSADM OMVS(GID(0))

Define a new group PROG1

ADDGROUP PROG1 OMVS(GID(25))

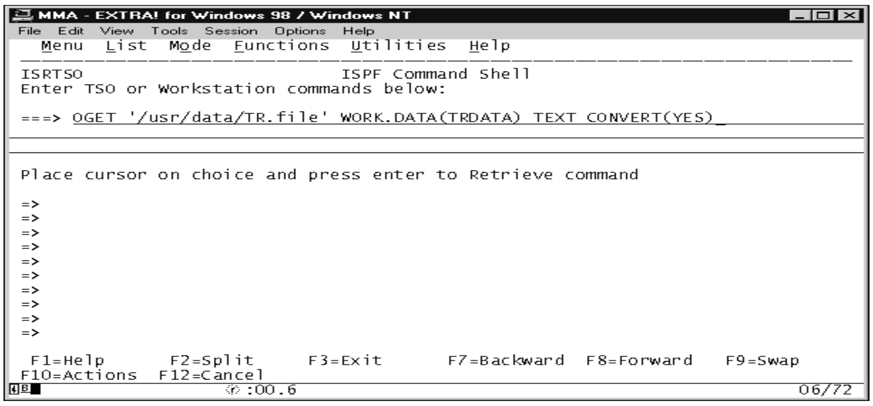
List the OMVS segment of user SIVA

LU SIVA OMVS

Two ways to enter TSO commands from ISPF

- Select option 6 from the Primary Option Panel to access the ISPF Command Shell panel shown in the next slide, and enter the command at the command prompt.
- Enter TSO followed by the command in the command area of any ISPF panel.

To access the OS/390 UNIX command shell, enter the TSO command OMVS at the ISPF Command Shell or TSO OMVS from any other ISPF panel.


MMA - EXTRA! for Windows 98 / Windows NT
File Edit View Tools Session Options Help

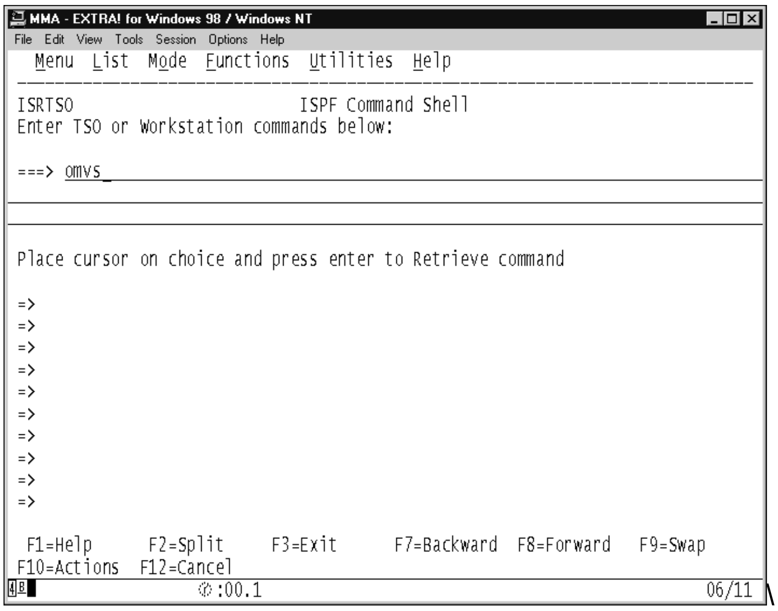
```
  Menu  List  Mode  Functions  Utilities  Help
 ------------------------------------------------------------------------------
 ISRTSO                        ISPF Command Shell
 Enter TSO or Workstation commands below:

 ===> OGET '/usr/data/TR.file' WORK.DATA(TRDATA) TEXT CONVERT(YES)_


 Place cursor on choice and press enter to Retrieve command

 =>
 =>
 =>
 =>
 =>
 =>
 =>
 =>
 =>
 =>

  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
                    :00.6                                                06/72
```



```
MMA - EXTRA! for Windows 98 / Windows NT
File  Edit  View  Tools  Session  Options  Help
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
ISRTSO                        ISPF Command Shell
Enter TSO or Workstation commands below:

===> omvs_

Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
=>
=>
=>
=>
=>
=>

 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
                    :00.1                                             06/11
```

To access the OS/390 UNIX command shell, enter the TSO command OMVS at the ISPF Command Shell or TSO OMVS from any other ISPF panel.

OGET 'pathname' data-set-name [BINARY|TEXT] [CONVERT(YES|NO)]
OPUT data-set-name 'pathname' [BINARY|TEXT] [CONVERT(YES|NO)]

Pathname	The pathname of the HFS file being copied to or from a data set.
data-set-name	The sequential file or partitioned data set member being copied to or from an HFS file.
BINARY	Specifies that the file being copied contains binary data.
TEXT	Specifies that the file being copied contains text data (the default).
CONVERT	For OGET, converts data from ASCII to EBCDIC. For OPUT, converts data from EBCDIC to ASCII.

If a data set name includes lowercase letters or special characters besides the period, hyphen, and parentheses, it must be enclosed in single quotes. Also, if a standard data set name isn't enclosed in single quotes, TSO will add the user-id to it as the first qualifier.
An OGET command that copies an HFS file to a PDS member

```
OGET '/usr/data/TR.file' WORK.DATA(TRDATA) TEXT CONVERT(YES)
```

An OPUT command that copies a PDS member to an HFS file

```
OPUT 'MM01.LOADLIB(TRPRG)' '/bin/trprg' BINARY
```

UNIX COMMANDS

cd	Changes the working directory you are currently in.
cp	Copies the contents of one file to another file. If the file doesn't already exist, the system will create it.
mv	Moves a file to another directory or renames it.
rm	Deletes a file.
grep	Searches a file for occurrences of a specified word.
ls	Lists the files in a directory.
Mkdir	Creates a directory. Any intermediate directories in the pathname you specify must already exist.
Rmdir	Deletes a directory.
pwd	Displays the full pathname of the current directory.
Who am I	Displays your username.
wc	Counts the number of words or lines in a file.

The BPXBATCH utility

BPXBATCH is an OS/390 utility that can be used to run UNIX shell commands, shell scripts, or executable HFS files (programs) through the OS/390 batch environment. BPXBATCH can be executed from JCL,The TSO/E ready prompt CLIST and REXX procedures, A program BPXBATCH is an MVS utility that you can use to run shell commands or shell scripts and to run executable files through the MVS batch environment. You can invoke BPXBATCH:

From the TSO/E READY prompt
From TSO CLISTs and REXX execs
From a program

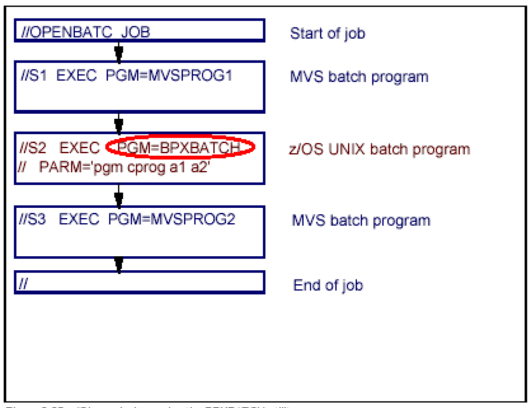


Figure 2-25 JCL used when using the BPXBATCH utility

BPXBATCH has logic in it to detect when it is run from JCL. If the BPXBATC program isrunning as the only program on the job step task level, it sets up the stdin, stdout, and stderr and execs the requested program. If BPXBATCH is not running as the only program at the job step task level, the requested program will run as the second step of a JES batch address space from JCL in batch. If run from any other environment, the requested program will run in a WLM initiator in the OMVS subsys category.

An IKJEFT01 job that copies one HFS file to another

```
//MM01HC   JOB  (36512),'R MENENDEZ',NOTIFY=&SYSUID
//STEP1    EXEC PGM=IKJEFT01
//SYSTSPRT DD   SYSOUT=*
//HFSCP    DD   PATH='/mm01/data/trans.data',PATHOPTS=ORDONLY
//STDOUTL  DD   PATH='/mm01/data/trans.data.copy',
//              PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//              PATHMODE=(SIRWXU,SIXGRP)
//SYSPRINT DD   SYSOUT=*
//SYSTSIN  DD   DATA
 ocopy indd(HFSCP) outdd(STDOUTL)
/*
```

An IKJEFT01 job that copies a PDS member to an HFS file

```
//MM01MC   JOB  (36512),'R MENENDEZ',NOTIFY=&SYSUID
//STEP1    EXEC PGM=IKJEFT01
//SYSTSPRT DD   SYSOUT=*
//MEMDATA  DD   DSN=MM01.WORK.DATA(ORDER),DISP=SHR
//HFSOUTL  DD   PATH='/mm01/data/order.file',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRWXU,SIXGRP)
//SYSPRINT DD   SYSOUT=*
//SYSTSIN  DD   DATA
 ocopy indd(MEMDATA) outdd(HFSOUTL)
/*
```

A BPXBATCH job that executes a Java program

```
//MM01BP3 JOB (36512),'R MENENDEZ',NOTIFY=&SYSUID
//STEP1   EXEC PGM=BPXBATCH,PARM='SH java /mm01/BookOrderApp'
//STDOUT  DD PATH='/mm01/output/mm01bp3.out',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRWXU,SIXGRP)
//STDERR  DD PATH='/mm01/output/mm01bp3.err',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRWXU,SIXGRP)
//STDENV  DD PATH='/mm01/envfil',
//          PATHOPTS=ORDONLY
//
```

A report program that accesses an HFS customer master file

```
//MM01RN   JOB  (36512),'R MENENDEZ',NOTIFY=&SYSUID
//STEP1    EXEC PGM=RPT3000
//STEPLIB  DD   DSN=MM01.TEST.LOADLIB,DISP=SHR
//CUSTMAST DD   PATH='/mm01/data/customer.master',PATHOPTS=ORDONLY
//SALESRPT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
```

System Management Facility(SMF)

System management facilities (SMF) collect and records system and job-related information that your installation can use in:

- Billing users.
- Reporting reliability.
- Analyzing the configuration.
- Scheduling jobs.
- Summarizing direct access volume activity.
- Evaluating data set activity.
- Profiling system resource use.
- Maintaining system security.

SMF formats the information that it gathers into **system-related records or job-related records**. System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information on the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session. An installation can provide its own routines as part of SMF. These routines will receive control either at a particular point as a job moves through the system, or when a specific event occurs. For example, an installation-written routine can receive control when the CPU time limit for a job expires or when an initiator selects the job for processing. The routine can collect additional information, or enforce installation standards.
SMF data is recorded in Man datasets. The naming convention used for SMF data sets

SYS1.MANx, such as **Sys1.man1, Sys1.man2, Sys1.man3…**

You should allocate at least two data sets for SMF use.

How to Start SMF

SET SMF=00

This command invokes the parmlib member SMFPRM00 which has the 'active' parameter which activates the subsystem.

How to Stop SMF

SET SMF=01

This command invokes the parmlib member SMFPRM01 which has the 'noactive' parameter which deactivates the subsystem.

PARMLIB :

SMFPRMxx -- Parameters that define SMF options.

Parameter	**Meaning and Use**	**Value Range**	**Default Value**
JWT=nn	This is a required parameter that initially specifies the number of minutes a job is allowed to wait continuously. When the specified time limit has expired, the time limit exit routine (IEFUTL) is entered, if exits are to be taken. can be changed by IEFUTL.	1-999	10
SID=xxxx	This required parameter specifies the system and model on which SMF is active, provided the installation modifies the default value.	Four alphanumeric and/or special characters	H155

MAN=NON
MAN=USEI
MAN=<u>ALL</u>
	This optional parameter specifies the type of records to be written to the SMF data set(s).
	Note: If MAN=ALL or USER, the BUF parameter is also required, and must not be zero or defaulted.
	NONE - specifies that no records are to be written to the SMF data set(s), regardless of values specified for OPT, DSV, and REC parameters.
	USER - specifies that only user records (types 128 through 255) are to be written to the SMF data set(s).
	ALL - specifies that both SMF-produced and user-produced records are to be written to
		Not applicable	ALL

	the SMF data set(s). All SMF records are created unless suppressed by the OPT, DSV, or REC parameters.		
OPT=	This optional parameter specifies whether system and job information, as opposed to system, job, and job step information is to be recorded. **1** - Specifies that only system and job-related information is to be collected by SMF. The step-initiation exit, IEFUSI, and the job step termination exit IEFACTRT are not taken. **2 -** Specifies that system, job, and job step information is to be collectged by SMF.	1-2	2
DSV=	This optional parameter specifies whether data set information and/or direct access volume information is to be collected by SMF. **0 -** specifies that neither data set information nor direct access volume information is to be collected. **1 -** specifies that direct access volume information is to be collected and data set information is to be suppressed. **2 -** specifies that data set information is to be collected and direct access volume information is to be suppressed. **3 -** specifies that both data set information and direct access volume information are to be collected. If either DSV=2 or 3 is specified and OPT=1 is also	0-3	0

	specified, SMF converts OPT=1 into OPT=2 and issues a warning message.		
REC=	This optional parameter specifies whether record type 17 will be written for temporary data sets. The parameter is not effective unless DSV is also specified as 2 or 3. **0 -** specifies that record type 17 is to be written for non-temporary data sets only, and information is to be suppressed for temporary data sets. **2 -** specifies that record type 17 is to be written for both temporary and non-temporary data sets.	0 or 2	0
BUF=	This parameter specifies the size of the SMF buffer. It must be specified if the MAN parameter equals ALL or USER. Specify a value that is a multiple of 8 bytes (double word). Otherwise, SMF rounds the value to the next *lower* multiple of 8 bytes. Before you reduce the buffer size specified in the previous IPL, you must dump the SMF data set(s). Otherwise, the data set(s) cannot be dumped.	400 to 8,192	None
OPI=	This optional parameter specifies whether the SMFPRMxx parameters are to be presented on the console during IPL for the operator's inspection and/or modification.	YES or NO	NO
EXT=	This optional parameter specifies whether SMF exits are to be taken. The parameter is independent of the value specified for MAN.	YES or NO	YES

What is Exit Routines?

Exit Routines are load Module, capable to inspect/filter the specified records.

Ex. of Exit Routine : IEFU83, IEFU84, IEFACTRT, IEFU29..

Display SMF Data sets:

```
D SMF
```

SWITCHING BETWEEN MAN DATA SETS

```
I SMF
```

The above command shifts the Control from sys1.man1 to sys1.man2..

Dumping the SMF Data Sets:

When notified by the system that a full data set needs to be dumped, the operator uses the SMF dump program (IFASMFDP) to transfer the contents of the full SMF data set to another data set, and to reset the status of the dumped data set to empty so that SMF can use it again for recording data.

The SMF dump program dumps the contents of multiple VSAM or QSAM data sets to sequential data sets on either tape or direct access devices. The SMF dump program allows the installation to route different records to separate files and produce a summary activity report.

```
//DUMPX      JOB   MSGLEVEL=1
//STEP1      EXEC  PGM=IFASMFDP
//DUMPIN     DD    DSN=data set name, DISP=SHR
//DUMPOUT    DD    DSN=SMFDATA,UNIT=unitaddr,
//                 DISP=(NEW,KEEP),LABEL=(,SL),VOL=SER=serial
//SYSPRINT   DD    SYSOUT=A
```

Sample JCL for Running the SMF Dump Program

The IRRADU00 Utility:

Reads the SMF data as exits within the SMF dump procedure and produces a sequential data set where the output can be used to extract information.

```
//DUMPX      JOB    MSGLEVEL=1
//SMFDUMP  EXEC PGM=IFASMFDP
//SYSPRINT DD  SYSOUT=A
//ADUPRINT DD  SYSOUT=A
//OUTDD    DD  DISP=SHR,DSN=ouput dsn
//SMFDATA  DD  DISP=SHR,DSN=input dsn
//SMFOUT   DD  DUMMY
//SYSIN    DD  *
      INDD(SMFDATA,OPTIONS(DUMP))
      OUTDD(SMFOUT,TYPE(000:255))
      ABEND(NORETRY)
      USER2(IRRADU00)
      USER3(IRRADU86)
/*
```

```
//      EXEC PGM=ICETOOL
//TOOLMSG   DD SYSOUT=*
//PRINT    DD      DISP=(NEW,CATLG,DELETE),DSN=ouput dsn ,
//           SPACE=(CYL,(100,5),RLSE),
//           DCB=(DSORG=PS,RECFM=V,LRECL=32700),
//           VOL=SER=VOL04
//DFSMSG    DD SYSOUT=*
//ADUDATA   DD DISP=SHR,DSN=input dsn
//TERM      DD DISP=(,PASS),SPACE=(CYL,(80,10)),DSN=&&TERM
//OUTFCNTL  DD *
  OUTFIL INCLUDE=(14,7,CH,EQ,C'TERM '),FNAMES=TERM
  OPTION   VLSHRT
//TOOLIN    DD *
  COPY    FROM(ADUDATA) USING(OUTF)
  DISPLAY FROM(TERM) LIST(PRINT) WIDTH(255) -
        PAGE -
        TITLE('LOGON DETAILS') -
        DATE(DMY/) -
        TIME(12:)  -
        BLANK -
        ON(32,10,CH)  HEADER('DATE') -
        ON(72,7,CH)   HEADER('USER GROUP')-
        ON(63,8,CH)   HEADER('USERID') -
        ON(14,7,CH)   HEADER('EXPERMENT') -
        ON(193,8,CH)  HEADER('TIME') -
        ON(23,8,CH)   HEADER('TIME') -
        ON(175,8,CH)  HEADER('TCPLOG')
```

HARDWARE CONFIGURATION DEFINITION(HCD)

The channel subsystem (CSS) and the IBM OS/390 operating system need to know what hardware resources are available in the computer system and how these resources are connected. This information is called hardware configuration.
Hardware Configuration Definition (HCD) provides an interactive interface that allows you to define the hardware configuration for both the channel subsystem and the operating system. Before HCD was available, you had to use IOCP (Input Output Configuration Program) to define the hardware tothe channel subsystem and the MVS Configuration Program (MVSCP) to define the hardware to the MVS operating system

What is HCD

Hardware Configuration Definition (HCD) is an z/OS component that supports you in defining both the operating system configuration and the processor hardware configuration of a system.
"CPC + OS"

IODF (Input Output Definition File)

The configuration you define with HCD may consist of multiple processors, each containing multiple partitions. HCD stores the entire configuration data in a central repository, the input/output definition file (IODF). The IODF as single source for all hardware and software definitions for a multi-processor or multi-partition system eliminates the need to maintain several independent MVSCP or IOCP data sets.
What are the datasets used now for definition

Advantage of IODF over MVSCP & IOCP

IODF as single source for all hardware and software definitions for a multi-processor or multi-partition system eliminates the need to maintain several independent MVSCP or IOCP data sets

Objects Managed in the IODF

HCD lets you define the configurations as objects and their connections.
The following objects and their connections are managed by HCD:

1. Operating system configurations	consoles
	EDTs (for MVS alone)
	esoterics (for MVS alone)
	Generic (for MVS alone)
	User-modified generics
2. Switches	
	Ports

Switch configurations
Port matrix

3. Processors

Partitions
Channel paths (chpid)

4. Control units
5. I/O devices

You can use HCD to perform the following major tasks:

- Define new configuration data (an IODF or parts of an IODF)
- Activate configuration data
- View and modify existing configuration data
- Maintain IODFs (such as, copy, import, and export)
- Query and print configuration data
- Migrate configuration data
- Compare the active I/O configuration to a defined I/O configuration in an IODF.

Difference Between IOCDS & IODF

An IOCDS can be created from a configuration definition in an IODF. The IOCDS contains the configuration for a specific processor, while the IODF contains configuration data for multiple processors. (IODF) that contains the definition of your configuration is used during IPL and activation I/O configuration data set (IOCDS) is used at POR.

Types of IODF

An IODF contains information about the I/O configuration. There are 3 types of IODF.

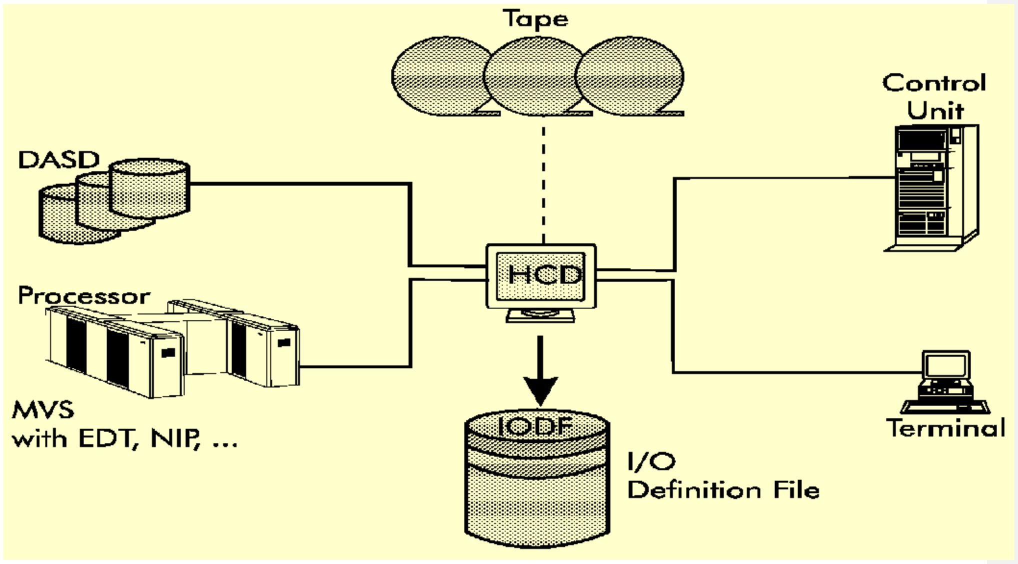
- Work IODF
- Validated work IODF
- Production IODF.

Work IODF

A work IODF allows you to create a new I/O configuration definition or modify an existing I/O configuration definition. A work IODF provides a way to build or modify an IODF before you use it to activate a configuration. It is a working copy, not suitable for IPL selection or activated during dynamic activation. When a work IODF is ready to use, build a production IODF from it.

Validated work IODF

A validated work IODF satifies all validation rules for building production IODFs. It may have at least one physical channel identifier (PCHID). In cooperation with HCD and the CHPID Mapping Tool a validated IODF is required to accept new or updated PCHIDs. From such a validated work IODF, an IOCP input deck suitable for use with the CHPID Mapping Tool is generated. As soon as all PCHIDs are inserted or updated in the validated work IODF, the production IODF can be built. In HCD, you can use various methods to obtain a validated work IODF.



Production IODF

A production IODF defines one or more valid I/O configurations. A configuration in a production IODF can be activated dynamically or selected during IPL. Although you can build multiple production IODFs, only the one that is selected during IPL or activated during dynamic configuration is the active production IODF.

Naming Convention of IODF datasets

Work IODF

The name of the data set for a work IODF has the format of:
'hhhhhhhh.IODFcc.yyyyyyyy.yyyyyyyy'

hhhhhhhh	is the high-level qualifier; up to 8 characters long.
Cc	is any two hexadecimal characters (that is, 0-9 and A-F).

Yyyyyyyy are optional qualifiers; up to 8 characters long.

The following qualifiers must not be used as last qualifier: CLUSTER, ACTLOG or MSGLOG.

You can use any number of optional qualifiers but do not make the total name longer than 35 characters because, in some circumstances, HCD appends an additional qualifier. If you omit the high-level qualifier and the enclosing single quotes, HCD automatically adds, as the high-level qualifier, your user prefix

Production IODF

The name of the data set for a production IODF has the same format as a work IODF. You may specify additional qualifiers to differentiate among IODFs (for example for backup reasons). However, the optional qualifiers must be omitted if the IODF is to be used for IPL or dynamic activation. Thus, the format would be:

'hhhhhhhh.IODFcc'

hhhhhhhh is the high-level qualifier; up to 8 characters long.

Cc is any two hexadecimal characters (that is, 0-9 and A-F).

The IODF is a VSAM LINEAR data set with different names for the cluster and the data component. The name of the data set with a cluster component has the format:

'hhhhhhhh.IODFcc.yyyyyyyy.yyyyyyyy.CLUSTER'

The name of the data set with a data component has the format:

'hhhhhhhh.IODFcc.yyyyyyyy.yyyyyyyy'

What are the Pre- Requisites of HCD

HCD is part of OS/390. It needs a running OS/390 system before it can be used to define a hardware configuration. Therefore, an installation should first load an OS/390 system, using an old IODF or a ServerPac Starter IODF to IPL the OS/390 system for the first time. When we are moving from MVSCP to HCD existing MVSCP configuration data has to be migrated into an IODF and is then used to IPL the system. HCD can be used on that system to define the full configuration. After you complete the input of your configuration data, you have to build a production IODF. The production IODF is used by the MVS operating system.

- To build the configuration data at IPL time. This active production IODF is used for building the IOCDS.

- The production IODF cannot be updated (read-only). This ensures that the data in the production IODF used at IPL remains the same during the run time of that system.

```
├─Control Unit
└─Device
```

UNIT INFORMATION MODULES (UIMs)

Unit information modules (UIMs) contain the information and rules that HCD uses to process I/O device definitions. When you create an IODF, HCD uses UIMs to validate the device definitions you enter. The system invokes UIMs at IPL or during a dynamic configuration change to build UCBs. UIMs contain device dependent information, such as parameters and features of devices. The UIMs must be installed in the OS/390 system before you use HCD to define a configuration. The UIMs are also used at IPL time to build the UCBs. That is why they have to be installed in SYS1.NUCLEUS at IPL time. UIMs are provided with IBM product software for devices that z/OS supports. For a non-supported device, you may be able to use a generic UIM or a UIM from a similar IBM device. Also if you have an MVSCP UIM for a device, you can convert it to an HCD UIM. You can write your own UIMs for non-IBM devices.

UNIT CONTROL BLOCK (UCB)

A unit control block (UCB) holds information about an I/O device, such as State information for the device, Features of the device. You can access information in UCBs using UCB services, such as UCBSCAN and UCBLOOK. At IPL or dynamic configuration, UCBs are built from HCD device definition information in the IODF and UIMs. There is a UCB for each I/O device in a configuration.

ELIGIBLE DEVICE TABLE (EDT)

An eligible device table (EDT) is an installation-defined and named representation of the I/O devices that are eligible for allocation. Using HCD, you define EDT information in an IODF. At IPL or dynamic configuration, information in the IODF and UIMs is used to build the EDT.

Types of Devices

Static
Installation-static
Dynamic

Static Device

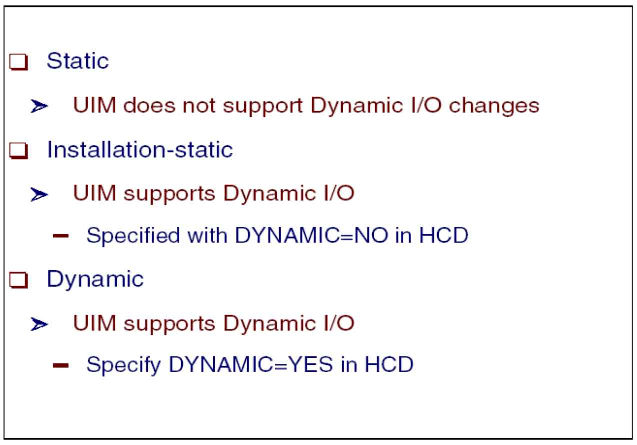
Defining a device as dynamic allows you to dynamically add, modify, and delete the device definition while MVS is running.

Installation Static

That means, that the device might support the dynamic capability, but the installation requests that the device is not treated as dynamic.
Installation-static devices can be dynamically added to the software I/O configuration, but can not be deleted or modified while OS/390 is running.

Dynamic

If your installation includes device types that support dynamic I/O configuration, you can change your I/O configuration without performing a power-on reset or re-IPLing the system.



- Static
  - UIM does not support Dynamic I/O changes
- Installation-static
  - UIM supports Dynamic I/O
    - Specified with DYNAMIC=NO in HCD
- Dynamic
  - UIM supports Dynamic I/O
    - Specify DYNAMIC=YES in HCD

HCD Offers

Single Point of Control

With HCD you have a single source, the IODF, for your configuration data. This means that hardware and software definitions as well as ESCON director definitions can be done from HCD and can be activated with the data stored in the IODF.

Increased System Availability

HCD checks the configuration data when it is entered and therefore reduces the chance of unplanned system outages due to inconsistent definitions.

Changing Hardware Definitions Dynamically

HCD offers dynamic I/O reconfiguration management. This function allows you to change your hardware and software definitions on the fly - you can add devices, or change devices, channel paths, and control units, without performing a POR or an IPL. You may also perform software only changes, even if the hardware is not installed.

Sysplex Wide Activate

HCD offers you a single point of control for systems in a sysplex. You can dynamically activate the hardware and software configuration changes for systems defined in a sysplex.

Migration Support

HCD offers a migration function that allows you to migrate your current configuration data from IOCP, MVSCP datasets into HCD.

Accurate Configuration Documentation

The actual configuration definitions for one or more processors in the IODF are the basis for the reports you can produce with HCD. This means that the reports are accurate and reflect the up-to-date definition of your configuration HCD provides a number of textual reports and graphical reports, that can be either printed or displayed. The printed output can be used for documentation purposes providing the base for further configuration planning tasks. The display function allows you to get a quick overview of your logical hardware configuration.

Guidance through Interactive Interface

HCD provides an interactive user interface, based on ISPF, that supports both the hardware and the software configuration definition functions. The primary way of defining the configuration is through the ISPF dialog. HCD consists of a series of panels that guide you through all aspects of the configuration task. The configuration data is presented in lists. HCD offers extensive online help and prompting facilities. Help includes information about panels, commands, data displayed, available actions, and context-sensitive help for input fields. A fast path for experienced users is also supported.

Batch Utilities

In addition to the interactive interface, HCD also offers a number of batch utilities. You can use these utilities, for instance, to migrate your existing configuration data to Maintain the IODF or to print configuration reports.

Cross Operating System Support

HCD allows you to define both OS/390 and VM/ESA configurations from OS/390.

Support of S/390 Microprocessor Clusters

HCD provides functions for IOCDS and IPL attributes management, which simplify the configuration and operational support for those processors that are configured in an S/390 microprocessor cluster.

How HCD Works

HCD stores the hardware configuration data you defined in the IODF. A single IODF can contain definitions for several processors (or LPARs) and several MVS or VM operating systems. It contains all information used to create IOCDSs and the information necessary to build the UCBs and EDTs. When HCD initiates the write IOCDS function, the IODF is used to build the IOCDS. The IOCDS with the channel subsystem definitions of a processor is then used to perform POR. The same IODF is used by MVS to read the configuration information directly from the IODF during IPL.

Hardware I/O Configuration

In the CPC, the channel subsystem (CSS), which controls channel operations, requires specific data about the hardware I/O configuration. To define the I/O configuration for the channel subsystem, run the Input/Output Configuration Program (IOCP). To run IOCP, you need to specify:

- Channel paths on the CPC
- Control units attached to the channel paths
- I/O devices assigned to the control units

Also, to meet changing I/O requirements, you can replace an existing I/O configuration with a new configuration by running IOCP.All control units and I/O devices that attach to the CPC must be defined to the channel subsystem by using IOCP.

I/O configuration definition process

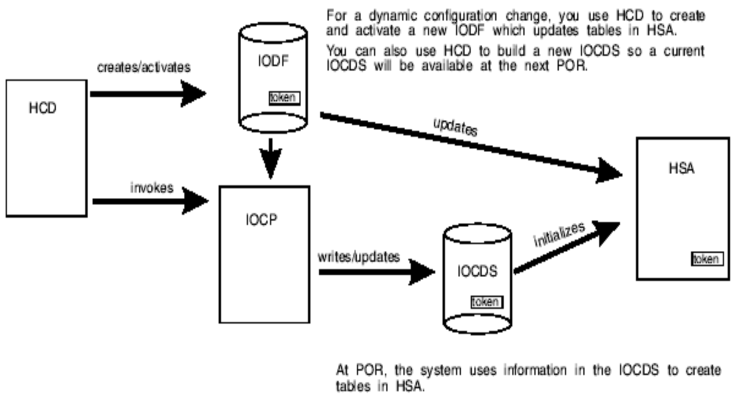
You specify a configuration to the software at IPL or at dynamic configuration. You specify a configuration to the hardware at POR or at dynamic configuration. When you dynamically change a configuration, you can change the I/O configuration definitions to both hardware and software or to software only. A software-only change modifies only the software control structures, such as the unit control blocks (UCBs) and the eligible device table (EDT). A hardware and software change modifies both the hardware and software control structures. In most cases, you will make simultaneous dynamic Configuration changes to both hardware and software configuration definitions.

Defining an I/O Configuration to the Hardware

A configuration is defined to hardware when an IOCDS initializes the IOCDS information in the hardware system area (HSA) during a POR. A configuration is also defined to hardware when information in HSA is updated during a dynamic configuration change.

Defining an I/O configuration to the software

Figure shows the process of defining an I/O configuration to the software. At IPL, the system reads an IODF and constructs UCBs, an EDT, and all device and I/O configuration-related control blocks. When you select and activate a new configuration, the system determines the changes needed to the UCBs and EDT based on comparing the current IODF and the newly selected IODF.


For a dynamic configuration change, you use HCD to create and activate a new IODF which updates tables in HSA.
You can also use HCD to build a new IOCDS so a current IOCDS will be available at the next POR.
creates/activates
HCD
IODF
token
updates
HSA
invokes
IOCP
initializes
writes/updates
IOCDS
token
token
At POR, the system uses information in the IOCDS to create tables in HSA.

I/O configuration program (IOCP)

I/O configuration program (IOCP) creates an IOCDS to define a configuration to the channel subsystem at POR. IOCP is shipped on the z/OS tape, and a stand-alone version is available on a processor. You can specify at any time that you want an IOCDS written for a configuration defined in an IODF. It is a good practice to write an IOCDS when you activate an IODF so a current IOCDS is available for the next POR. Use the HCD Activate and Process Configuration Data panel and its option “Build IOCDS”. HCD invokes IOCP to write an IOCDS. IOCP builds a configuration definition from IODF information and stores the definition in an IOCDS so it is available when a POR is required. Processors can have multiple I/O configuration data sets. On an HCD panel, you can specify the one that you want to update.

IOCDS (Input/Output Configuration Data Set)

An IOCDS contains information to define the I/O configuration to the processor complex’s channel subsystem. The IOCDS is built by the IOCP. A POR loads the IOCDS into HSA to initialize the hardware to reflect the IOCDS. IOCDSs for a processor are stored on the Support element hard disk for:

1. S/390 9672 Parallel Enterprise Server R2 and R3 models, as well as Generation 3 (these processors will run OS/390 Version 2 Release 10, but not z/OS V1R1 or higher)
2. S/390 9672 Parallel Enterprise Server G5 and G6
3. S/390 Application StarterPak (3000 models)
4. S/390 Multiprise 2000 (2003 models)(this processor will run OS/390 Version 2 Release 10, but not z/OS V1R1 or higher)
5. S/390 Multiprise 3000 (7060 models)
6. IBM Eserver zSeries 800
7. IBM Eserver zSeries 900
8. IBM Eserver zSeries 990v Processor controller DASD for other processors

HARDWARE SYSTEM AREA (HSA)

The hardware system area (HSA) contains tables that include information about the current configuration. Ensuring that the software and hardware definitions match Before your first dynamic configuration change, you must use HCD to create an IOCDS from the IODF that will be used for a subsequent IPL, then perform a POR using that IOCDS. The POR places information about the hardware configuration in the hardware system area (HSA).

The same IODF must be used at IPL to define the software configuration. To be able to perform a software and hardware dynamic change, your hardware and software definitions must match. When you use the same IODF to define your software and hardware configurations, the software and hardware definitions match. However, there may be times when you want to change your software definition.
For example to temporarily run a test system to change your software definition, you might perform a software-only dynamic change using a different IODF. In this case, your software and hardware definitions do not match and you cannot perform a full hardware and software dynamic configuration change.
To be able to make a dynamic hardware and software change, you can perform a software-only change using the IODF that was used to define the current hardware configuration. This IODF is the one used to define the software and hardware configuration at POR or at the last full software and hardware dynamic configuration change. You could also perform a POR and the subsequent IPLs using the IODF with the current software configuration so the software and hardware definitions match.

z/OS delivery options

You can install z/OS using one of several IBM packages. These two packages are available at no additional charge when you license z/OS:

1. **ServerPac:** This software delivery package consists of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E

installation steps. You use the CustomPac Installation Dialog to install your system and complete the installation of the software it includes.

2. **CBPDO:** The Custom Built Product Delivery Option (CBPDO) is a delivery package that consists of uninstalled products and unintegrated service. You must use SMP/E to install the individual z/OS elements and features, and their service, before you can IPL.

Several fee-based options are available

3. **SystemPac:** SystemPac offers the capability to build a system with integrated subsystems in either full volume dump/restore format or data set copy format. The full volume dump/restore format enables you to install z/OS without using the dialog. Installation is done via pack restore using DFSMSdss or FDR.

SystemPac is designed for those who have limited skill or time to install or upgrade z/OS, but who want to install or upgrade to exploit z/OS functions in e-commerce or other areas. SystemPac tailors z/OS to your environment (such as DASD layout, migration of MSVCP/IOCP to IODF, and naming conventions) based on information provided to IBM. With this offering, selected non-IBM products can be integrated.

4. **SoftwareXcel:** SoftwareXcel Installation Express (SIE) is available in the U.S. only, and provides pre-built z/OS system packages in full volume dump format, tailored to customer hardware and software configurations. SIE includes on-site planning, installation, and package testing.

5. **Entry Server Offering:** The Entry Server Offering, only available in selected countries, is a packaged solution that includes hardware, software, installation services, maintenance, and financing to help customers get current technology.

System and installation requirements

Hardware and software requirements, including non-IBM software compatibility.

- Virtual storage mapping
- Application performance
- Building a minimum number of system software configurations
- Reducing installation and migration time
- Reducing the opportunities for error during migration
- Making it easier to manage the system after it is in production
- Minimizing migration actions for the people who use the system

The *driving system* is the system image (both the hardware and software) that you use to install your new system image, the *target system*. You log on to the driving system and run jobs there to create or update the target system. Once the target system is built, it can be IPLed on the same hardware (same LPAR or different LPAR on the same processor) or on different hardware than that used for the driving system.

- To prepare the driving system *before* building the target system, you need to perform the following tasks:
- Identify the software requirements for the driving system according to the delivery package you are using, for example, ServerPac.
- Identify the hardware requirements for the driving system.

You are also required to do some preparations for the target system:

- Choose the software products to install and identify requisites.
- Order z/OS and related IBM products.
- Identify the hardware requirements for the target system.
- Identify the service needed for the target system.
- Decide whether to use the existing JES2 or JES3 with the new z/OS release.

ServerPac:

This software delivery package consists of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E installation steps. You use the CustomPac Installation Dialog to install your system and complete the installation of the software it includes.

ServerPac service level

For ServerPac orders, service is *integrated* with product code according to the following **time-line:**

ServerPac is refreshed every month to pick up the most current Recommended Service Upgrades (RSUs). All products incorporate HIPER and PTF-in-error (PE) fixes that are available approximately one week before your order is received. Your ServerPac order also contains a service tape containing all unintegrated service that was available at the time your order was processed.

z/OS installation using ServerPac

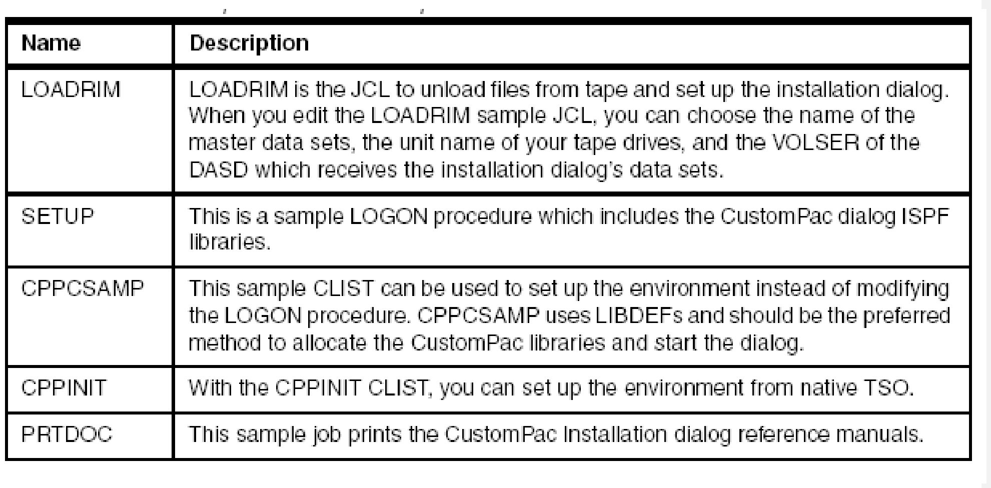
Your z/OS ServerPac order contains an ISPF dialog that you use to install z/OS. This dialog is called the CustomPac installation dialog because it is used to install all of IBM's CustomPac offering.

Before you begin the installation, you should

Review the contents of the ServerPac shipment that you received from IBM by checking the packing slip to make sure that you have a complete set of installation tapes and documentation. Make sure your user ID has ALTER authority for the following high-level qualifiers:

- CPAC
- SYS1
- All product-specific high-level qualifiers for products that come with your package.

During of the job phase of the installation process you may need RACF SPECIAL authority or equivalent if you use other security software.
If you decided to use SMS to manage data sets in your order, your user ID needs READ access to FACILITY class, profile STGADMIN.IGG.DIRCAT.
The RIM tape contains sample procedures, JCL, jobs, and CLISTs. They are in SYS1.orderid.DOCLIB.



Name	Description
LOADRIM	LOADRIM is the JCL to unload files from tape and set up the installation dialog. When you edit the LOADRIM sample JCL, you can choose the name of the master data sets, the unit name of your tape drives, and the VOLSER of the DASD which receives the installation dialog's data sets.
SETUP	This is a sample LOGON procedure which includes the CustomPac dialog ISPF libraries.
CPPCSAMP	This sample CLIST can be used to set up the environment instead of modifying the LOGON procedure. CPPCSAMP uses LIBDEFs and should be the preferred method to allocate the CustomPac libraries and start the dialog.
CPPINIT	With the CPPINIT CLIST, you can set up the environment from native TSO.
PRTDOC	This sample job prints the CustomPac Installation dialog reference manuals.

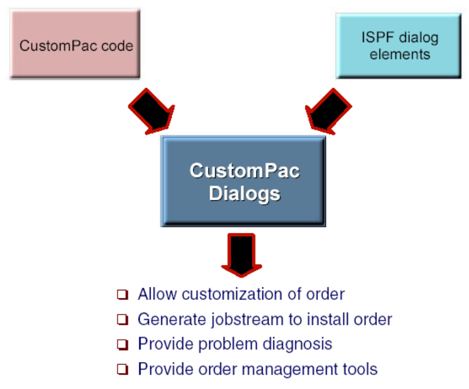
Installing z/OS

ServerPac is a software delivery package consisting of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E installation steps.
To install the package on your system and complete the installation of the software it includes, you use the CustomPac Installation Dialog.
A z/OS ServerPac order contains an Interactive System Productivity Facility (ISPF) dialog that you use to install z/OS. This dialog is called the CustomPac Installation Dialog because it is used to install all of IBM's CustomPac offerings, for example, ServerPac, SystemPac, FunctionPac, ProductPac and ServicePac.

When ordering a ServerPac:

1. IBM selects the products that were ordered.
2. IBM integrates products and selected service into target and distribution libraries and their SMP/E zones.
3. IBM enables the features that you ordered that use dynamic enablement.
4. IBM verifies the resulting system for the specific package by doing an IPL, submitting a job, logging on to TSO/E, and checking the job's output. IBM performs this test using the operational data sets supplied with the ServerPac. If you use the software upgrade path when installing a ServerPac, you will use your own existing operational data sets, so this test will not assure that the system will IPL in your environment. However, it does make sure that the software itself can be used to IPL given a usable set of operational data sets.
5. IBM selects all unintegrated service for products ordered and includes it on a service tape.

Starting the CustomPac dialogs


CustomPac code
ISPF dialog elements
CustomPac Dialogs
Allow customization of order
Generate jobstream to install order
Provide problem diagnosis
Provide order management tools

The CustomPac dialogs are now installed. To access the dialogs, we recommend that you:

1. Customize the CPPCSAMP CLIST, changing the *custompac.qualifier* according to the qualifiers of the master dialogs data sets.
2. Save the customized clist.
3. Go to TSO/ISPF, option 6 and execute the new CLIST:
   exec 'DATA.SET.NAME(CPPCSAMP)'

   Where DATA.SET.NAME is the partitioned data set where you saved the CLIST and CPPCSAMPis the member name where you saved the customized CPPCSAMP CLIST.

Custom Built Product Delivery Option (CBPDO)

The Custom Built Product Delivery Option (CBPDO) is a delivery package that consists of uninstalled products and un integrated service. You must use SMP/E to install the individual z/OS elements and features, and their service, before you can IPL.

CBPDO service level

Remember, the service in the CBPDO orders is *not integrated*. You must receive and apply the service during the installation process. The service that is shipped with every CBPDO order is as follows:

Service for all products previously ordered, as reflected in the customer profile for the customer number used when the order was placed. You can specify the starting date for these PTFs at order time. The default starting date is the last time that customer number was used to order a CBPDO (product or service) for that system release identifier (SREL).
Service for all products, elements, and features that you have ordered. This includes all PTFs that became available between product general availability and the time of your order that have not been incorporated in the product FMIDs.

Installing z/OS: Custom-built product delivery option

The custom-built product delivery option (CPBDO) is a software delivery package consisting of uninstalled products and unintegrated service. You must use SMP/E to install the individual z/OS elements and features, and their service, before you can IPL.
To order CBPDO, use an order checklist (available from an IBM representative, the z/OS Web site, or IBMLink via the configurator). The order is for a unique system release identifier (SREL). It is recommended that you order all products that you maintain in the same z/OS product set.
It is further recommended that you order only z/OS elements and features (or their equivalent stand-alone products) in your CBPDO order. (Ordering equivalent levels of non-exclusive elements does not increase the size of the CBPDO because the FMIDs are the same, but it does enable the IFAPRD00 PARMLIB member to be built correctly.) If you need to update other products, place a separate CBPDO order for these products.

When ordering a CBPDO:

1. IBM selects the products that were ordered.
2. IBM selects service for the products you order and for products that are already licensed under the same customer number you use to place your order.
3. IBM builds a customized job stream to enable the features that you ordered that use dynamic enablement.
4. IBM builds a customized job stream to enable selected features that use the registration service.

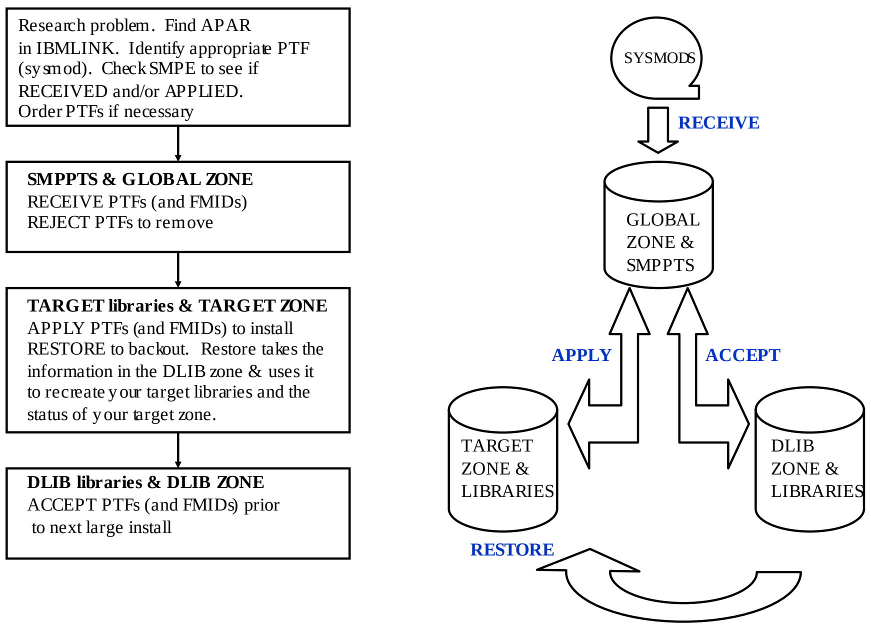
System Modification Program/Extended(SMP/E)

What Is SMP/E?

- A tool designed to manage the installation of software products on your OS/390 system
- And track the modifications you make to those products.

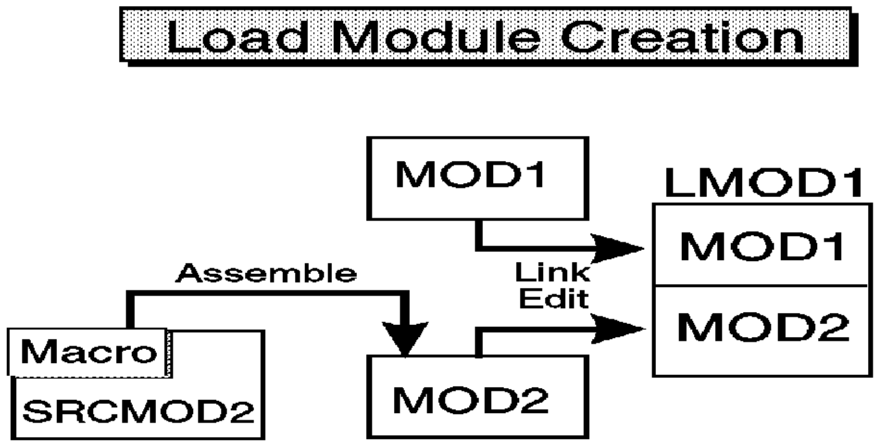
Why Should We Use SMP/E?

- system programmer's responsibility to ensure that all software products and their modifications are properly installed on the system.
- Due to complexity of the SO/390


Research problem. Find APAR in IBMLINK. Identify appropriate PTF (sysmod). Check SMPE to see if RECEIVED and/or APPLIED. Order PTFs if necessary
SMPPTS & GLOBAL ZONE
RECEIVE PTFs (and FMIDs)
REJECT PTFs to remove
TARGET libraries & TARGET ZONE
APPLY PTFs (and FMIDs) to install
RESTORE to backout. Restore takes the information in the DLIB zone & uses it to recreate your target libraries and the status of your target zone.
DLIB libraries & DLIB ZONE
ACCEPT PTFs (and FMIDs) prior to next large install
SYSMODS
RECEIVE
GLOBAL ZONE & SMPPTS
APPLY
ACCEPT
TARGET ZONE & LIBRARIES
DLIB ZONE & LIBRARIES
RESTORE

Load Module

- Each system function is composed of one or more load modules.
- A load module represents the basic unit of machine-readable, executable code.
- Load modules are created by combining one or more object modules and processing them with a link-edit utility.
- Link-editing of modules is a process that resolves external references and addresses.



System Element

- These modules, macros, and other types of data and code are the basic building blocks of your system. All of these building blocks are called elements.

SYSMOD

- You may want to add some new functions to your system, upgrade some of the elements of your system, or modify some elements - system modifications
- A SYSMOD is the actual package containing information SMP/E needs to install and track system modifications.
- SYSMODs are composed of two parts
  Modification control statements (MCS)
  Modification text

```
000001 ++ PTF (UQ67403)      /*  SYSMOD Name
000003 ++ VER (Z038)             indicated it's a mvs based product
000004      FMID(HIP6140)          function modification identifier
```

```
 000005     PRE  (UQ67699)
 000006     SUP  (AQ62094)
/*
 000052  E    B   aH  -j   `    d   !  d(¢H;    /           K ¢T      d    <  (
 000053  c K:   oB2 & c    F         5@     a   /   {S    ^+   cu  I   a i   0  E
 000054  & u(   q    c 8  B   a    I   B      k cu  I   a   `> {     d      -L
 000055   uX   ak - c  u2      & c   F        /   {I   f=  /   {S YK  &- c   B
 000056   I   ab    cu   I    b| Dm  {E   ak ,,c  \Z       /I   B       `> {
 000057  X   ak -  * uX   ak -     *         /   R    AK   c  Om     d  )   2   +-
 000058       +VF0  -       {     B       F H ' RP< 5R   _  Q5xfAW
 000059  +  ;      { _ 6;  ? 8 {  I?       {      7F yy_@  aR      ¢
 000060       > -     y /    N&     0        ¦~      /  Pba 7l 6    ¦    + +     \
 000061     fn   {K  8       N     kI  I c  \ e     pK       }  ;   f} \     d
 000062        5 K !>  r  sOz w   A    bc     m   h Q `5 <    m     K\   E +   s
 000063   }&  5>' 0  q     B     .-Aa# + t  m    m      u \        z  zdi k  q>!z }
 000064   @ w      i ^  t      k. fo          :             >   0  {  e
 000065         -A   n    s% K      b Z_  H             / b wB   &
```

Modification control statements (MCS) is an 80 byte record designated by ++ as the first two characters, that tell SMP/E:

- What elements are being updated or replaced
- How the SYSMOD relates to product software and other SYSMODs
- Other specific installation information

Modification text, which is the object modules, macros, and other elements supplied by the SYSMOD

Modification Control S	**Nature of associated text**
++MAC	a macro replacement
++MACUPD	a macro update (IEBUPDTE control statements)
++SRC	a source replacement
++SRCUPD	a source update (IEBUPDTE control statements)
++MOD	an object module
++ZAP	IMASPZAP control statements
Data element MCS statements, ex. ++CLIST or ++PROC	Data of various types, e.g. TSO clists (++CLIST) and procedures (++PROC)

The first record in any SYSMOD must be a header modification control statement, which identifies the type of SYSMOD and its individual name.
The header will be a ++FUNCTION, ++PTF, ++APAR, or ++USERMOD statement, Depending on the type of SYSMOD.

Modification identifier

- FMID (function modification identifier)
- RMID (Replacement modification identifier)
- UMID (User modification identifier)

Any prerequisite /corequisite/superseded PTFs. Conditional requisites are identified at this stage with ++IF statements. Any hold reasons are represented with ++HOLD statements

INFILE - IF MCS and Modification text are together
RELFILE- MCS and Modification text are separate

Four different categories of SYSMODs

Function SYSMODs--Introduce the elements for a product.

PTF SYSMODs (program temporary fix) - Prevent or fix problems with an element, or introduce new elements.

APAR SYSMODs (authorized program analysis reports) - Fix problems with an element.

USERMOD SYSMODs (user modifications) -Customize an element.

1.The Function SYSMOD

The function SYMOD introduces a new product, a new version or release of a product, or updated functions for an existing product into the system.

When we refer to installing a function SYSMOD, we are referring to the placing of all product's elements in the system data sets, or libraries. Examples of these libraries are SYS1.LINKLIB, SYS1.LPALIB and SVCLIB. The figure below

There are two types of function SYSMODs:

- A **base function** SYSMOD adds or replaces the entire system function

- A **dependent function** SYSMOD provides an addition to an existing system function. It is called dependent because its installation depends upon a base function already installed.

2. The PTF SYSMOD

When a problem with a software element is discovered, IBM supplies its customers with a tested fix for that problem. This will come in a form of a program temporary fix (PTF).

Most PTFs are designed to update or replace one or more complete elements of a system function. For ex. In the above diagram we see a previously installed load module, LMOD2. If we want to replace the element MOD1, we should install a PTF SYSMOD that contains the module MOD1. That PTF SYSMOD replaces the element in error with the corrected element.

3. The APAR SYSMOD

At times, you will find it necessary to correct a serious problem that occurs on your system before a PTF is ready for distribution. Therefore, in this circumstance, IBM supplies you with an authorized program analysis report (APAR). An APAR is a fix designed to correct a specific area of an element or replace an element in error.

The APAR SYSMOD always has the installation of a function SYSMOD as a prerequisite and can also be dependent upon the installation of other PTFs or APAR SYSMODs.

4.The USERMOD SYSMOD

If you have a requirement for a product to perform differently than what it was intended to, you can consider a customized element for that system. IBM provides certain modules allowing you to adapt IBM code for your specific requirements. After you make the desired modifications, you place these modules into your system by installing a USERMOD SYSMOD. This SYSMOD can be used to replace or update an element or to introduce a totally new written element into your system. In all circumstances, the USERMOD SYSMOD is created by you either to modify IBM code or to add your own code to the system. Here we see that MOD3 has changed through a USERMOD SYSMOD

SYSMOD Prerequisites

- PTF, APAR, and USERMOD SYSMODs all have the function SYSMOD as a prerequisite.
- PTF SYSMODs may be dependent upon other PTF SYSMODs. APAR SYSMODs may be dependent upon PTF SYSMODs and other APAR SYSMODs.

- USERMOD SYSMODs may be dependent upon PTF SYSMODs, APAR SYSMODs, and other USERMOD SYSMODs.

Tracking and controlling element

The means for SMP/E to track and control elements successfully is to identify them uniquely. This is where we use the modification identifiers and there are three types.

- **Function Modification Identifiers (FMIDs)** identify the function SYSMOD that introduces the element into the system.
- **Replacement Modification Identifiers (RMIDs)** identify the last SYSMOD (in most cases a PTF SYSMOD) to replace an element.
- **Update Modification Identifiers (UMIDs)** which identifies the SYSMOD that an update to an element since it was last replaced.

HOLD DATA

HOLDDATA is often supplied for a product to indicate a specified SYSMOD should be held from installation. Reasons for holding a SYSMOD can be:

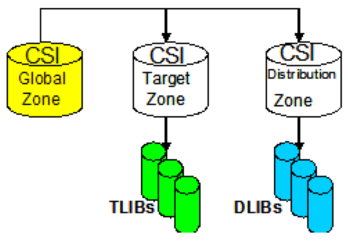
ERROR HOLD A PTF is in error and should not be installed until the error is corrected.
SYSTEM HOLD Certain system actions may be required before SYSMOD installation.
USER HOLD The user may want to perform some actions before installing the SYSMOD.

How Does SMP/E Work?

- The Distribution and Target Libraries
- The Consolidated Software Inventory (CSI)

Consolidated Software Inventory (CSI)

- 'Card catalog' Contain all the information SMP/E needs to track the distribution and target libraries.
- Contain an entry for each element in its libraries. element name, type, History

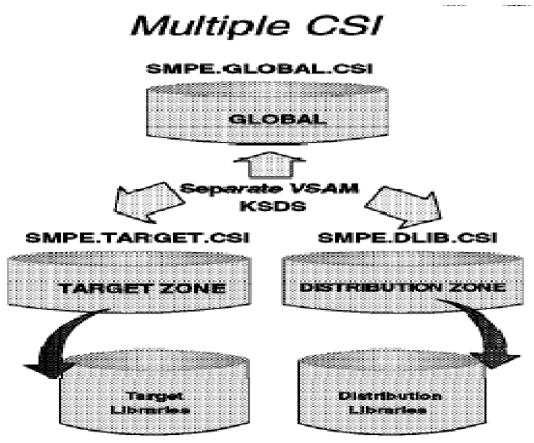


- How the element was introduced into the system
- Pointer to the element in the distribution and target libraries

Global zone Contains entries needed to identify and describe each target and distribution zone to SMP/E and stores information about SMP/E processing options. Contains status information for all SYSMODs SMP/E has begun to process and holds exception data for SYSMODs requiring special handling or that are in error.

Target zone Contains information that describes the content, structure, and status of the target libraries. It also contains a pointer to the related distribution zone, which can be used in APPLY, RESTORE, and LINK when SMP/E is processing a SYSMOD and needs to check the level of the elements in the distribution libraries.

Distribution zone Contains information that describe the content, structure, and status of the distribution libraries. Each distribution zone also points to the related target zone, which is used when SMP/E is accepting a SYSMOD and needs to check if the SYSMOD has already been applied.



There can be more than one zone in an SMPCSI data set (in fact, there can be up to 32766 Zones per data set). For example, an SMPCSI data set can contain a global zone, several target zones, and several distribution zones. The zones can also be in separate SMPCSI data sets.
One SMPCSI data set can contain just the global zone, a second SMPCSI data set the target zones, and a third SMPCSI data set the distribution zones.

```
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//CSIVOL    DD UNIT=SYSALLDA,
  //         VOL=SER=volser,
  //         DISP=SHR
//SYSIN  DD  *
            DEFINE CLUSTER  (NAME(#globalcsi)  FREESPACE(10,5)  KEYS(24 0)
            RECORDSIZE(24 143)  SHAREOPTIONS(2 3) UNIQUE    -
            VOLUMES(volser) - )     DATA (NAME(#globalcsi.DATA)   -
            CONTROLINTERVALSIZE(4096)   CYLINDERS(25 5)  INDEX   -
            (NAME(#globalcsi.INDEX)     CYLINDERS(2 1)   IMBED   )
```

How to initialize a CSI data set

Before you use a CSI, you must initialize it with the GIMPOOL record, which is in SYS1.MACLIB. If you use the access method services REPRO command to copy an entire CSI data set to a newly created CSI data set, you do not have to initialize your newly allocated CSI with a GIMZPOOL record. A GIMZPOOL record is copied in when the existing CSI is copied into your new CSI. Make sure you use the correct SMP/E release GIMZPOOL record to initialize your CSI created with that SMP/E release.

```
//PRIMEIT EXEC PGM=IDCAMS,COND=(9,LT)
//SYSPRINT  DD SYSOUT=*
//SMPCSI   DD DSN=#globalcsi, DISP=OLD
//ZPOOL    DD DSN=SYS1.MACLIB(GIMZPOOL), DISP=SHR
 //SYSIN    DD *
      REPRO OUTFILE(SMPCSI) INFILE(ZPOOL)
```

There are two types of libraries : **Distribution and Target Libraries**

- Distribution libraries contain all the elements, such as modules and macros, that are used as input for running your system.
- Target libraries contain all the executable code needed to run the system.

SMP/E PROCESS Using Commands:

- RECEIVE
- APPLY
- ACCEPT
- RESTORE
- REJECT
- LIST

<u>SMP/E PROCESSING</u>

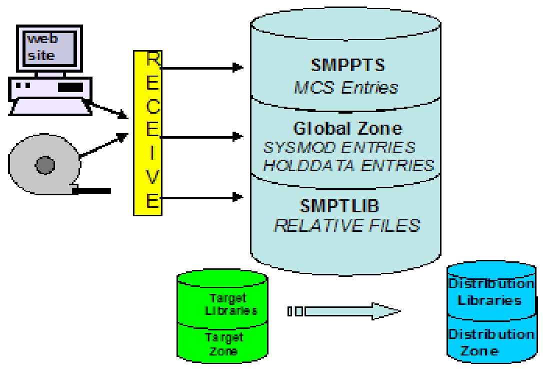
The SMP/E process is performed by three simple basic commands, (RECEIVE, APPLY, ACCCEPT).

1.**RECEIVE Processing:**

Loads the SYSMOD information from the distribution medium (tape files or DASD dataset) into the global zone , and temporary data sets (SMPTLIBs) for later SMP/E processing.

The RECEIVE command processes data from two sources:

- The SMPPTFIN data set, which contains the modification control statements (MCSs)
- The SMPHOLD data set, which contains exception SYSMOD data

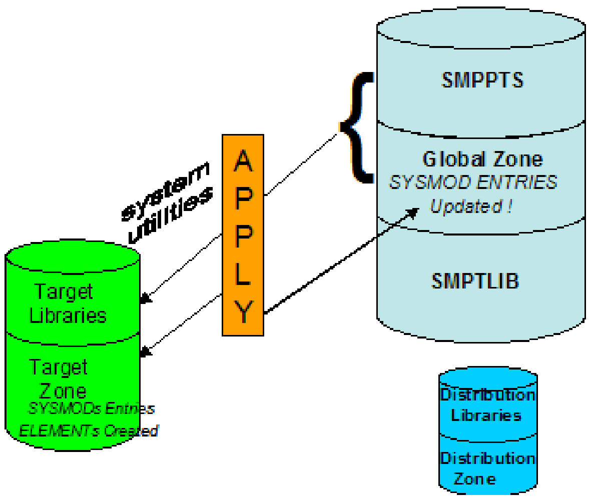


SYSMOD processed by SMP/E contains two types of information

- Instructions telling SMP/E what elements are in the SYSMOD and how to install them (MCS)
- The actual element replacements or updates contained in the SYSMOD(RELFILE, INLINE and Indirect)

2. APPLY Processing

- Installs the SYSMODEs into the appropriate target system libraries.



What happens during APPLY processing ?

- Select which received SYSMOD has to be applied

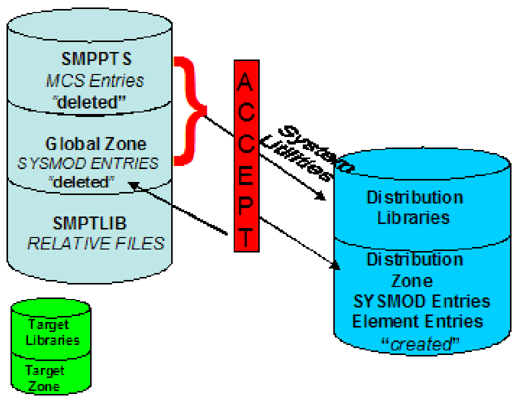
- Checks to make sure all other required SYSMODs (prerequisites) have been installed
- Determine which elements should be installed in the target libraries.
- Checks the Apply
- Update Target Libraries

What happens After APPLY processing?

- A SYSMOD entry is created in the target zone for each SYSMOD that has been applied.
- Element entries (such as MOD and LMOD) are also created in the target zone for those elements that have been installed in the target libraries.
- SYSMOD entries in the global zone are updated to reflect that the SYSMOD has been applied to the target zone.
- BACKUP entries are created in the SMPSCDS data set so the SYSMOD can later be restored, if necessary.

3. The ACCEPT Command

The ACCEPT command is used to cause SMP/E to install the elements supplied by a SYSMOD into the distribution libraries (or DLIBs).



The ACCEPT process:

Selects SYSMODs present in the global zone that are applicable to the specified distribution libraries. Makes sure all other required SYSMODs have been accepted or are being cepted concurrently. Calls system utilities to install the elements into the distribution libraries. Records the functional and service levels of the new elements in the distribution zone. Records the installation of the SYSMOD in the distribution zone. Deletes the global zone SYSMOD and PTS MCS entries for those   SYSMODs successfully processed

4. The REJECT Command

The REJECT command allows you to clean up the global zone, SMPPTS, and associated entries and data sets. REJECT is helpful if the SMPPTS is being used as a permanent database for all SYSMODs, including those that have been installed. You can also use it to purge old data from the global zone and SMPPTS.
To use the REJECT command, you must first determine which processing mode of the command you want to use. The mode you choose depends on the data you want to delete.

Mass mode: SMP/E rejects all SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied.

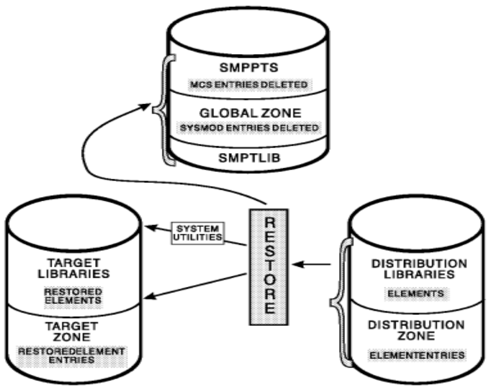
Select mode: SMP/E rejects specific SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied.

PURGE mode: SMP/E rejects all SYSMODs that have been accepted into the specified distribution zones. PURGE mode can be used when SYSMODs were not automatically deleted once they were accepted. This is the case if NOPURGE was coded in the OPTIONS entry used to process the distribution zone.

NOFMID mode: SMP/E rejects all SYSMODs applicable to functions that are not part of the system. NOFMID mode can be used to delete service for all functions that have been deleted from the global zone. NOFMID mode can also be used to delete FEATURE and PRODUCT entries for all functions that have been deleted from the global zone.

5. The Restore Command

At times, you may want to remove a SYSMOD that has been applied to the target libraries. For example, perhaps you installed a fix for a problem and got unexpected results. If you have not yet accepted the SYSMOD into the distribution libraries, you can use the RESTORE command to remove it from the target libraries. The RESTORE command replaces the affected elements in the target libraries with the unchanged versions from the distribution libraries. (As a result, once you have accepted a SYSMOD into the distribution libraries, you cannot use RESTORE to remove it from the target libraries.)



5. The LIST Command

The SMP/E data sets contain a great deal of information--the global zone, target zones, distribution zones, SMPPTS, SMPLOG, and SMPSCDS--that you may find useful when installing a new function, preparing a user modification, or debugging a problem. You can use the SMP/E LIST command to display that information.

SMP/E can display all the entries of a specified type (such as MOD, MAC, SYSMOD, and so on), or it can display information for selected entries.

DDNAME	Function of Dataset	Dataset Attributes
SMPCSI	Consolidated Software Inventory - a database of SYSMODs, datasets, etc, under SMP's control	One or more VSAM clusters
zonename	Target or distribution zone of the CSI – zone name is defined by the user	Either a VSAM cluste or a logical subset of one cluster for the whole CSI
tlib	Target library - contains the code of one or more of the software products under	Partitioned dataset

	SMP's control - DDNAME generally correspond s to low-level qualifier of dataset name	
dlib	Distribution library - contains components required to rebuild members of target libraries if a change to the target library needs to be backed out - DDNAME generally corresponds to low-level qualifier of dataset name	Partitioned dataset
SMPCNTL	Input dataset containing SMP commands to be executed	Card-image dataset. Always required
SMPPTFIN	Input dataset containing SYSMODS to be RECEIVEd	Card-image dataset
SMPHOLD	Input dataset containing HOLDDATA to be RECEIVEd	Card-image dataset
SMPOUT	SMP message printfile (RPT and LIST output also comes here if SMPRPT and SMPLIST are not seperately defined)	SYSOUT dataset
SMPRPT	SMP report print file	SYSOUT dataset
SMPLIST	Printfile for output from the LIST command	SYSOUT dataset
SYSPRINT	Printfile for output from utilities invoked by SMP	SYSOUT dataset
SMPPUNCH	SMP command file punched out by the UNLOAD, GENERATE, and REPORT commands`	Cardimage output file

SMPLOG	History log	Sequentialdataset. Always required.
SMPLOGA	Alternate history log	Sequential dataset. Always required.
SMPPTS	PTF temporary store - used to hold RECEIVEd SYSMODs	Card-image partitioned dataset
SMPMTS	Macro temporary store - intermediate library for storage of macros which are updated or replaced by an APPLY but do not reside in a target library	Card-image partitioned dataset
SMPSTS	Source temporary store - like SMPMTS, but for source which is updated/replaced by an APPLY and does not reside in a target library	Card-image partitioned dataset
SMPTLIBs	Temporary libraries used to hold SYSMODs RECEIVEd from RELFILEs on tape	Partitioned datasets
SYSUT1-4 SMPWRK1-5	Work and scratch datasets used by utilities invoked by SMP	Temporary files

Storage Management Subsystem

Data Facility Storage management Subsystem(DFSMS)

The DFSMS/MVS software product, together with IBM hardware products, comprises the key to manage system storage in an MVS environment. The components of DFSMS/MVS automate and centralize storage management, based on installation policies which define the availability, performance, space, and security.

Components of DFSMS are

- DFSMSdfp
- DFSMSdss
- DFSMShsm
- DFSMSrmm
- DFSMStvs

DFSMSdfp

DFSMSdfp provides the foundation for:
Storage management : DFSMSdfp includes ISMF, an interactive facility that allows you to define and maintain policies to manage your storage resources. These policies help to improve the utilization of storage devices, and to increase levels of service for user data, with minimal effort required from users.

Tape mount management : DFSMSdfp's Storage Management Subsystem provides a means for implementing tape mount management (TMM), a methodology for improving tape utilization and reducing tape costs. This methodology involves intercepting selected tape data set allocations through the SMS automatic class selection (ACS) process, and redirecting them to a DASD buffer. Once on DASD, these data sets can be migrated to a single tape or small set of tapes, thereby reducing the overhead associated with multiple tape mounts.

Data management : DFSMSdfp helps you store and catalog information on DASD, optical, and tape devices, so that it can be quickly identified and retrieved from the system.

Program management : DFSMSdfp combines programs into executable modules, prepares them to run on the operating system, stores them into program libraries, and reads them into storage for execution.

Device management : DFSMSdfp is involved in defining your input and output devices to the system, and in controlling the operation of those devices in the MVS/ESA environment.

DFSMSdss

A reliable utility to quickly move, copy and backup data. Data Set Services (DFSMSdss) is a component of DFSMS (Data Facility Storage Management Subsystem), and is used to:

Move and replicate data : DFSMSdss offers powerful, user-friendly functions that let you move or copy data between volumes of like and unlike device types. It can also copy data that has been backed up.

Manage storage space efficiently : DFSMSdss can increase performance by reducing or eliminating DASD free-space fragmentation.

Backup and recover data : DFSMSdss provides you with host system backup and recovery functions at both the data set and volume levels. It also includes an enhanced Stand-alone Restore Program that lets you restore vital system packs during disaster recovery - without a host operating system.

Convert data sets and volumes :DFSMSdss can convert your data sets and volumes to system-managed storage, or return your data to a non-system-managed state as part of a recovery procedure - all without data movement

Concurrent Copy (CC) : Concurrent Copy (CC) is both an extended function in cached IBM Storage Controls and a component of DFSMSdss. CC enables you to copy or dump data while applications are updating the data. Both IMS/ESA and DB2 can use CC for their backups.

FlashCopy : FlashCopy provides a fast data duplication capability. This option helps eliminate the need to stop applications for extended periods of time in order to perform backups and restores. FlashCopy is automatically initiated by DFSMSdss on IBM Enterprise Storage Servers (ESS) with the FlashCopy feature.

Tape Initialization : Before tapes can be used on most mainframe applications, they must be initialized. This involves writing a volume label (typically 80 bytes) and sometimes some header labels at the front of the tape. The serial number in the volume label is the same as is physically added to the tape. This enables manual operators or robots to physically locate and select a tape.

Backup and Restoration : One of the most important responsibilities for any storage administrators is to take backup of all necessary data as the data is prone to be physically or logically be destroyed.

DFSMSdss Stand-Alone Services
DFSMSdss Stand-Alone Services was previously introduced as an APAR enhancement to all releases of DFSMS. The stand-alone restore function is a single-purpose program designed to allow the system programmer to restore vital system

packs during disaster recovery without needing to rely on an OS/390 or z/OS environment.

DFSMShsm provides functions for:

Storage management : DFSMShsm uses a hierarchy of storage devices in its automatic management of data, relieving end-users from manual storage management tasks.

Space management: DFSMShsm improves DASD space utilization by keeping only active data on fast-access storage devices. It automatically frees space on user volumes by deleting eligible data sets, releasing over allocated space & moving low activity data to lower-cost-per-byte devices.

Tape mount management: DFSMShsm can write multiple output data sets to a single tape, making it a useful tool for implementing tape mount management (TMM) under SMS. When you redirect tape data set allocations to DASD, DFSMShsm can move those data sets to tape, as a group, during interval migration.
This greatly reduces the number of tape mounts initiated on the system. DFSMShsm uses single file format, which also improves your tape usage and search capabilities.

Availability management : DFSMShsm backs up your data--automatically or by command to ensure availability in the event of accidental loss of data sets or physical loss of volumes. DFSMShsm also allows a storage administrator to copy backup and migration tapes. These copies can be stored on site as protection from media damage, or off site as protection from site damage. Disaster backup and recovery is also provided for user-defined groups of data sets, so that critical applications can be restored at the same location or at an off-site location.

DFSMShsm

DFSMShsm is a DASD storage management and productivity tool for managing low-activity and inactive data. DFSMSHSM performs two functions

Space management

Move low-activity data sets from user-accessible volumes to DFSMShsm volumes.
Reduce the space occupied by data on both the user-accessible volumes and the DFSMShsm Volumes.

Availability management:

Copy all the data sets on DASD volumes to tape volumes
Copy the changed data sets on DASD volumes either to other DASD volumes or to tape volumes

Functions of space management

1. Automatic primary space management of DFSMShsm- managed volumes, which includes:
    - Deletion of temporary data sets and expired datasets
    - Release of unused, over-allocated space
    - Migration to DFSMShsm-owned migration volumes

2. Automatic secondary space management of DFSMShsm-owned volumes, which includes:
    - Migration level cleanup, including deletion of expired migrated data sets and some migration control data set (MCDS) records
    - Moving migration copies from migration level 1 (ML1) to migration level 2 (ML2) volumes

3. Automatic interval migration, initiated when a DFSMShsm-managed volume exceeds a specified threshold.

4. Automatic recall of user data sets back to DASD volumes

5. Space management by command

6. Space-saving functions

Functions of availability management:

1. Automatic physical full-volume dump and incremental backup
2. Automatic control data set backup
3. Command dump and backup and recovery
4. Expiration of backup versions
5. Disaster backup
6. Aggregate backup and recovery

Space management: Releasing of unused space, deletion of datasets depending upon the amount of time since the creation date and last referenced date, migrating datasets to ML1 or directly to ML2, migration from ML1 to ML2, migration of older generation groups, expiring or migrating the rolled off generations. The tasks for controlling automatic space management of SMS-managed storage require adding DFSMShsm commands to the ARCCMDxx member and specifying attributes in the SMS storage groups and management classes Storage attributes, Management attributes, Expiration attributes, Partial release, Migration attributes, GDG management attributes.

Availability management:

To ensure copies of datasets which can be used in case of any lost or damage.

Storage Device Hierarchy

- Level 0: DFSMShsm-managed storage devices at the highest level of the hierarchy these devices contain data directly accessible to your application.
- Level 1 and Level 2: Storage devices at the lower levels of the hierarchy; level 1 and '] level 2 contain data that DFSMShsm has compressed and optionally compacted into a format that you cannot use. Devices at this level provide lower cost per byte storage and usually slower response time. Usually L1 is in a cheaper DASD (or the same cost, but with the gain of compression) and L2 is on tape.

Backup and dump

Backup is the process of copying a data set from a level 0 or an ML1 volume to daily backup volume. This copy is called a backup version. The purpose of backup is to have copies of data sets in case something happens to the original data sets. The difference between dump and backup is that the dump function backs up the entire allocated space on a volume, whereas the DFSMShsm backup function backs up individual data sets.

Dump Tasks

- Specifying which volumes to dump and the dump classes to use
- Specifying when automatic dump processing starts
- Specifying the DFSMSdss DASD I/O buffering technique to use for dump
- Defining dump volumes to DFSMShsm

Backup Tasks

- Specifying what storage group has automatic backup performed
- Specifying whether to use concurrent copy
- Specifying frequency of backup
- Specifying the maximum number of backup versions to keep
- Specifying whether to perform automatic backup
- Specifying the start time for automatic backup
- Specifying the first qualifier for names of backup versions
- Specifying whether to back up RACF discrete data set profiles
- Specifying whether DFSMShsm performs spill processing
- Specifying whether to back up only changed data sets
- Specifying the days on which backup occurs
- Defining backup volumes to DFSMShsm

Spill process

A DFSMShsm process that moves all but the latest backup version of a data set from a DASD daily backup volume to a spill backup volume

ABARS:

- Aggregate backup is the command-driven function that backs up a user-defined group (called an aggregate group) of data sets for recovery at another computer site or at the same site.
- Aggregate recovery is the command-driven function that recovers the data sets that were previously backed up by aggregate backup.Aggregate backup and recovery support can be used to back up and recover both SMS and non-SMS-managed data.

DFSMShsm support activities

DFSMShsm maintains three control data sets (CDSs):

- **Backup control data set (BCDS)** - The backup control data set (BCDS) provides DFSMShsm with information defining the backup and dump environment. DFSMShsm needs this information to manage its backup data sets and volumes.
- **Migration control data set (MCDS)** - The migration control data set (MCDS) provides information about migrated data sets and the volumes they migrate to and from.
- **Offline control data set (OCDS)** - The offline control data set (OCDS) provides DFSMShsm with information about each migration and backup tape and about each data set residing on these tapes.
- **Journal Data Set** - The journal data set provides DFSMShsm with a record of each critical change made to a control data set from any host since the last time that CDS was successfully backed up.

DFSMShsm provides for automatic backup of the control data sets and the journal each time that automatic backup is performed for the system. It can also be backed up by command.

DFSMSrmm Concepts and facilities

DFSMSrmm (RMM) is the IBM Tape Management System (RMM stands for “Removable Media Manager”). DFSMSrmm can manage all of your tape volumes and the data sets on those volumes. It protects tape data sets from being accidentally overwritten, manages the movement of tape volumes between libraries and vaults over the life of the tape data sets, and handles expired and scratch tapes, all according to policies that you define. DFSMSrmm runs in its own subsystem address space inside the z/OS operating system. It communicates directly with the Subsystem Interface (SSI), allowing DFSMSrmm to fully control the tape usage in the system. DFSMSrmm supplies ISPF panels as well as TSO commands. Thus, we can access in online, you can also do in batch through TSO command execution.

Locations are places where RMM manages tapes, either within your local tape library, or in a remote location, such as an archival or disaster recovery site.
RMM uses three built-in storage locations, LOCAL, DISTANT, and REMOTE, but you can define your own locations to use instead.

Racks are six-character strings defining shelf spaces in your local tape library - they usually match the volser stored in the rack, but you can use different rack and volser names if you wish.

Bins are six-character strings defining shelf spaces at a remote location - these are usually numbered sequentially from a predetermined number, since a bin at a remote location will hold many different tape volsers as tapes cycle through the remote location
The RMM tape catalog is a VSAM KSDS known as the ***Control Data Set*** (CDS). The CDS records all information about RMM, including tapes, volsers, locations, etc.
The CDS is dynamic, allowing all information about RMM to be dynamically defined or deleted, without requiring an RMM outage, or in many cases without impacting RMM operation.

To ensure CDS integrity, RMM uses a ***journal*** file to record all CDS updates between backups of the CDS. If the CDS fails for any reason, it can be recovered by restoring the last CDS backup, and then using the journal file to apply all updates with a forward recovery.

Removable media library

A removable media library contains all the tape and optical volumes that are available for immediate use, including the shelves where they reside. A removable media library usually includes other libraries: system-managed libraries such as automated or manual tape libraries; and non-system-managed libraries, containing the volumes, shelves, and drives not in an automated or a manual tape library.

Automated tape library

An automated tape library consists of robotics components, cartridge storage areas, tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives.

EDGRMMxx is the parmlib member in which the parameters are supplied when RMM procedure is started.

Vital Record Specifications (VRS)

Vital Record Specifications (VRS) is a component of RMM that sets policies for retention and movement of tape volumes amongst local and remote locations.
Three types of VRS, dataset, volume, and name.

A ***dataset*** VRS defines dataset retention by days or cycles, using a dataset name, a dataset mask, a management class, or a VRS management value.

A ***volume*** VRS defines volume retention by days or the total number of volumes to be retained, using a specific or generic volser.

A ***name*** VRS defines additional move and count criteria for each location, but cannot exceed retention criteria specified in the initial VRS.

VRS is processed as part of RMM housekeeping, EDGHSKP, which also performs other housekeeping chores, such as backing up the RMM control dataset and journal, clearing the journal, expiration (scratch) processing, storage location management, and creating an extract dataset for reports.

EDGRPTD is typically run to create reports from the extract dataset, such as inventory reports and volume movement reports.

DFSMSrmm

DFSMSrmm manages your removable media resources, including tape cartridges and reels. It provides functions for:

Library Management : To create tape libraries, or collections of tape media associated with tape drives, to balance the workload of your tape drives and operators.
DFSMSrmm can be used to manage system-managed tape libraries such as the IBM 3494 and IBM 3495 Automated Tape Library Dataservers or the manual IBM 3495 Tape Library Dataserver Model M10. DFSMSrmm can also manage the more traditional non-system-managed tape libraries.

Shelf Management : DFSMSrmm groups information about removable media by shelves into a central online inventory and keeps track of the volumes residing on those shelves. DFSMSrmm can also record the shelf location for optical disks and track their vital records status.

Volume management : DFSM Srmm helps manage the movement and retention of tape volumes throughout their life cycle.
Data set management : DFSMSrmm records information about the data sets on tape volumes to validate volume and data set information and to help maintain data integrity. It can also control the retention of those data sets.

ISMF (Interactive System Management Facility)

The **I**nteractive **S**ystem **M**anagement **F**acility is an ISPF menu driven application used to control SMS. ISMF is designed to use the storage management, data management, space management and availability management (backup/recovery) functions provided by DFSMSdfp, DFSMShsm, DFSMSdss, DFSMSrmm. ISMF works with the following products that you should be familiar with:

Interactive System Productivity Facility/Program Development Facility (ISPF/PDF), which provides the edit, browse, Data Set and Library utility functions TSO/Extensions (TSO/E), TSO CLISTs and commands.

- Data Facility SORT (DFSORT), which provides the record-level functions.

- RACF, a component of the SecureWay Security Server for OS/390, which provides access control function for data and services.

- Device Support Facilities (ICKDSF) to provide the storage device support and analysis functions.

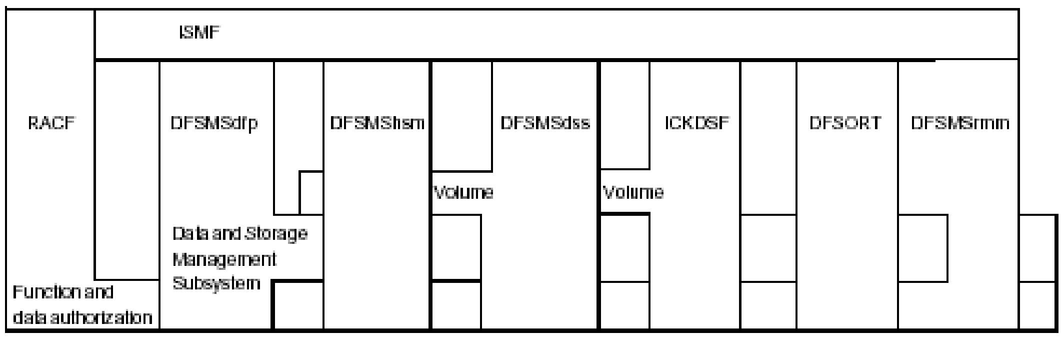

ISMF
RACF
Function and data authorization
DFSMSdfp
Data and Storage Management Subsystem
DFSMShsm
Volume
DFSMSdss
Volume
ICKDSF
DFSORT
DFSMSrmm

Figure 1. Storage Management Product Relationship. ISMF provides access to the functions of the DFSMS family of products.

ISMF PANEL

You can either go into User mode or Storage administrator mode

```
Select one of the following options and press Enter:

 0  ISMF Profile              -  Change ISMF User Profile
 1  Data Set                  -  Perform Functions Against Data Sets
 2  Volume                    -  Perform Functions Against Volumes
 3  Management Class          -  Specify Data Set Backup and Migration Criteria
 4  Data Class                -  Specify Data Set Allocation Parameters
 5  Storage Class             -  Specify Data Set Performance and Availability
 9  Aggregate Group           -  Specify Data Set Recovery Parameters
 L  List                      -  Perform Functions Against Saved ISMF Lists
 R  Removable Media Manager   -  Perform Functions Against Removable Media
 X  Exit                      -  Terminate ISMF
```

Specify the following for that: Go into 0   - ISMF profile

```
User Mode.    (To specify your choice of session, type in a:

                 1 For an End User (EU)
                 2 For a Storage Administrator (SA)
                 in the User Mode Field and Press Enter to Verify Your Selection.)
```

<u>Purpose of ISMF</u>

ISMF is a panel-driven interface application to:

Display lists of information about specific data sets, DASD volumes, mountable optical volumes, and mountable tape volumes. Generate lists of data, storage, and management classes to find out how data sets are being managed. Display and manage lists saved from various ISMF applications.

SMS configuration

An SMS configuration is composed of a set of data class, management class, storage class, storage group, optical library and drive definitions, tape library definitions, and ACS routines to assign the classes and groups. It also includes the aggregate group definitions and the SMS base configuration.

Control dataset : Types of control data sets:

Source Control Data Set (SCDS)
Active Control Data Set (ACDS)
Communications Data Set (COMMDS)

Source Control Data Set (SCDS)

The SCDS contains a set of SMS classes and groups and translated ACS routines that implement a specific set of storage management policies.This is the source control data set, the SMS configuration that you develop and test. You can have several SCDSs. One should correspond to your current policies. Retain at least one prior configuration should you need to regress to it because of error. The SCDS is never used to manage allocations.

1. Allocating an SCDS:

```
//STEP      EXEC PGM=IDCAMS
//SYSUDUMP       DD   SYSOUT=*
//SYSPRINT       DD   SYSOUT=*
//SYSIN     DD   *
DEFINE    CLUSTER(NAME(SMS.SCDS1.SCDS)     LINEAR     -
VOL(VOLID)     TRK(66)     SHAREOPTIONS(2,3)REUSE)-
DATA      (NAME(SMS.SCDS1.SCDS.DATA))
/*
```

Active Control Data Set (ACDS)

The ACDS is the systems active copy of the current SCDS. When you activate a configuration, SMS copies the existing configuration from the specified SCDS into the ACDS. By using copies of the SMS classes, groups, volumes, optical libraries, optical drives, tape libraries, and ACS routines rather than the originals, you can change the current storage management policy without disrupting it.

Communications Data Set (COMMDS)

The COMMDS data set contains the name of the ACDS and storage group volume statistics. It enables communication between SMS systems in a multisystem environment. COMMDS contains space statistics, SMS status, and MVS status for each system-managed volume. Only one COMMDS is used at a time for an SMS installation, we have more COMMDSs on different volumes for recovery purposes. The COMMDS must reside on a shared device accessible to all systems and allocated on the same device as the ACDS. Create a spare COMMDS in case of a hardware failure or accidental data loss. SMS activation fails if the COMMDS is unavailable.

SMS CLASS

The SMS classes and groups are customized by the storage administrator based on the installation environment and storage policies. The SMS classes and group are:

Data class
A list of allocation attributes for data sets. The data class simplifies and standardizes data set creation.

Storage class
A list of storage performance and availability service requests. The storage class contains space, availability, and performance attributes, such as response time and cacher equirements,fordatasets.

Management class
A list of backup, retention, and migration attributes for data sets.

Storage group
A group of one or more DASD volumes that SMS uses for data set allocation.

ACS routines

Automatic class selection (ACS) routines are written by the storage administrator. They automatically assign SMS classes and groups to data sets, database data, and objects. Data allocations are processed through ACS routines and to enforce installation standards, for data allocation and override user specifications, for data, storage, and management classes and requests for specific DASD volumes.
Four ACS routines in an SMS configuration can be created, one for each type of SMS class and one for storage groups. For SMS data sets the storage group ACS routine is required because it is not possible to explicitly specify storage groups. The other ACS routines are optional. For objects, the storage group, storage class, and management class ACS routines are required. For tape, the storage group, storage class, and data class ACS routines are required.

Order of ACS routines process for new dataset allocation:

1. **Data Class**
2. **Storage Class**
3. **Management Class**
4. **Storage Group**

DASD (Direct access storage device)

A type of storage device, such as a magnetic disk, in which bits of data are stored at precise locations, enabling the computer to retrieve information directly without having to scan a series of records. It’s a high capacity high speed auxiliary storage unit. The storage capacity varies from 10 GB to several TBs. The Total space

is spread across different volumes. Each volume can be identified by a unique volume Serial number, assigned to it.
(E.g.: VOL01)

Tracks	A Track is a circular ring on either side of the disk
Cylinders	A cylinders is a set of matched tracks
Seek	Moving the head to correct track
Rotate	Rotate disk under the head to correct sector
Settle	Head lowers to the disk wait for vibrations from moving to stop
Data transfer	Copy data to main memory

Magnetic tape uses a method similar to that of VCR tape for storing data. The speed of access can be quite slow, however, when the tape is long and what you want is not near the start. So this method is used primarily for major backups of large amounts of data. Businesses especially might do a backup of the day's transactions every day and a backup of the whole system once a week or so. Keeping sets of backups like this minimizes the amount of data loss when the computer system goes down.

Traditional DASD

In the era of traditional DASD, the hardware consisted of controllers like 3880 and 3990, Which contained the necessary functions to operate a storage subsystem. The controllers were connected to S/390 systems via parallel or ESCON channels. Behind a controller you had several model groups of the 3390 that contained the disk drives.
Based on the models, these disk drives had different capacities per device. Within each model group, the different models provide either four, eight, or twelve devices. All A-units come with four controllers, providing a total of four paths to the 3990 Storage Control. At that time, you were not able to change the characteristics of a given DASD device.

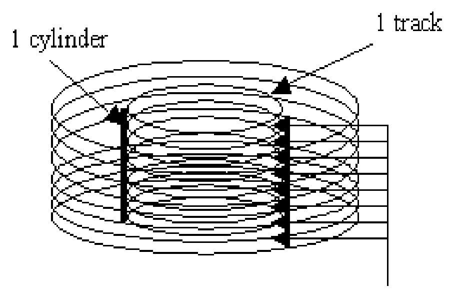
- 3380 devices were used in the 1980s. Capacity went from 885 to 2,655 cylinders per volume.
- When storage density increased, new device types were introduced at the end of the 1980s.
- This types were called 3390, Capacity per volume ranged from 1,113 to 3,339 cylinders.
- A special device type, model 3390-9 was introduced to store large amounts of data that needed very fast access.
- The track geometry within one device category was (and is) always the same; this means that 3380 volumes have 47,476 bytes per track, and 3390 volumes have 56,664 bytes per track.

Large volume 3390

- The IBM TotalStorage Enterprise Storage Server (ESS) initially supported custom volumes of up to 10017 cylinders, the size of the largest standard volume, the 3390 model 9.
- The IBM TotalStorage ESS large volume support (LVS) enhancement, announced in November 2001, has now increased the upper limit to 32760 cylinders, approximately 27.8 GB.
- The enhancement is provided as a combination of IBM TotalStorage ESS licensed internal code (LIC) changes and system software changes, available for z/OS, OS/390, and z/VM.
- Large Volume Support (LVS) is available on z/OS and OS/390 operating systems, and the ICKDSF and DFSORT utilities.
- Large Volume Support must be installed on all systems in a sysplex prior to sharing data sets on large volumes. Shared system/application data sets cannot be placed on large volumes until all system images in a sysplex have Large Volume Support installed.

3390 disk information

Every 3390 disk volume contains 56,664 bytes per track, 15 tracks per cylinder and 849,960 Bytes per Cylinder. A 'track' was the amount of data which could be read from a single surface in one revolution, without moving the heads. A 'cylinder' could be read from all 15 surfaces without moving the heads. This was quite important, as a cylinder could be read quite quickly, without any mechanical movement. The diagram might help explain



Modern disks use FBA storage, but they have to emulate 3390 CKD format, so the terms 'track' and 'cylinder' are still used.

3390 disk numbers

Model	Model 1	Model 2	Model 3	Model 9	Model 27
Tracks per Volume	16,695	33,390	50,085	150,255	450,765
Cylinders per Volume	1,113	2,226	3,339	10,017	30,051
Bytes per Volume	946 MB	1.89 GB	2.84 GB	8.51 GB	25.53 GB

ESS technology

The IBM TotalStorage Enterprise Storage Server (ESS) is IBM’s most powerful disk storage server, developed using IBM Seascape architecture. The ESS provides unmatched functionality to the family of e-business servers, and also to non-IBM (that is, Intel-based and UNIX-based) families of servers. Across all of these environments, the ESS features unique capabilities that allow it to meet the most demanding requirements of performance, capacity, and data availability that the computing business may require.

Volume table of contents (VTOC)

- The VTOC lists the data sets that reside on its volume, along with information about the location and size of each data set, and other data set attributes.
- The VTOC is a contiguous data set; that is, it resides in a single extent on the volume. It is pointed to by a record in the first track of the volume, and starts after cylinder 0, track 0 and before track 65,535.
- A VTOC's address is located in the VOLVTOC field of the standard volume label. Data is organized in physical blocks preceded by the highest record key in the block (that is, a count-key-data format).
- The VTOC has six types of control blocks; they are called data set control blocks (DSCB) and they describe data set characteristics, free space, and other functions
- A set of macros called the Common VTOC Access Facility (CVAF) allows a program to access VTOC information.

Data set control block (DSCB)

- The VTOC is composed of 140-byte data set control blocks (DSCBs) that correspond either to a data set or virtual storage access method (VSAM) data space currently residing on the volume, or to contiguous, unassigned tracks on the volume.
- DSCB is the name of the logical record within the VTOC. DSCBs describe data sets allocated on that volume, and also describe the VTOC itself. The system automatically constructs a DSCB when space is requested for a data set on a direct access volume.

- Each data set on a DASD volume has one or more DSCBs to describe its characteristics. The DSCB appears in the VTOC and, in addition to space allocation and other control information, contains operating system data, device-dependent information, and data set characteristics. There are seven kinds of DSCBs, each with a different purpose and a different format number.
- The first record in every VTOC is the VTOC DSCB (format-4).
- The record describes the device, the volume the data set resides on, the volume attributes, and the size and contents of the VTOC data set. The next DSCB in the VTOC data set is a free-space DSCB (format-5), even if the free space is described by format-7 DSCBs. The third and subsequent DSCBs in the VTOC can occur in any order.

VTOC index

- The VTOC index enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC. It consists of an index of data set names in format-1 DSCBs contained in the VTOC and volume free space information.
- If the system detects a logical or physical error in a VTOC index, the system disables further access to the index from all systems that might be sharing the volume. If a VTOC index becomes disabled, the VTOC remains usable but with possibly degraded performance.
- If a VTOC index becomes disabled, you can rebuild the index without taking the volume offline to any system.
- All systems can continue to use that volume without interruption to other applications, except for a brief pause during the index rebuild. After the system rebuilds the VTOC index, it automatically re-enables the index on each system that has access to it.

VTOC index record (VIR)

The Device Support Facilities (ICKDSF) initializes a VTOC index into 2048-byte physical blocks called VTOC index records (VIRs). VIRs are used in several ways. A VTOC index contains the following kinds of VIRs:

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

VTOC index format-1 DSCB

The name of the index must be 'SYS1.VTOCIX.*xxxxxxxx*', where *xxxxxxxx* conforms to standard data set naming conventions and is usually the serial number of the volume containing the VTOC and its index. The name must be unique within the system to avoid ENQ contention.

Creating the VTOC and VTOC index

To initialize a volume (preparing for I/O activity), use the Device Support Facilities (ICKDSF) utility to initially build the VTOC. You can create a VTOC index at that time by using the ICKDSF INIT command and specifying the INDEX keyword.You can use ICKDSF to convert a non-indexed VTOC to an indexed VTOC by using the BUILDIX command and specifying the IXVTOC keyword. The reverse operation can be performed by using the BUILDIX command and specifying the OSVTOC keyword.

Device Support Facilities (ICKDSF)

ICKDSF is a program you can use to perform functions needed for the initialization, installation, use, and maintenance of DASD volumes. You can also use it to perform service functions, error detection, and media maintenance.

Initializing a DASD volume

After you have completed the installation of a device, you must initialize and format the volume so that it can be used by MVS. The INIT command writes a volume label (on cylinder 0, track 0) and a VTOC on the device for use by MVS. It reserves and formats tracks for the VTOC at the location specified by the user and for the number of tracks specified. If no location is specified, tracks are reserved at the default location.
If the volume is SMS-managed, the *STORAGEGROUP* option must be declared.

```
//  jobname   job    ……….
// step1        exec  pgm=ickdsf
// myvol        dd     unit= devicetype,disp=old,vol=ser=col123
// sysprint     dd      sysout=*
// sysin        dd      *
```

You define a VTOC and Index when you initialise a volume with ICKDSF. A typical command would look like

```
   INIT UNIT(D615) VOLID(43D615) VTOC(1,0,270) INDEX(0,1,14) VFY(OLDVOL)
```

The VFY statement is a verify to check the old label on the volume, this makes it less likely to initialize the wrong volume. You need the volume off-line to all systems before initializing it. Note that the ICKDSF statement does not make the disk a 3390. This happens when the disks are logically configured in the hardware.

Allocating a VVDS

The system will allocate a VVDS for you the first time a VSAM file is allocated on a disk, but the default system VVDS is usually too small. You can allocate a VVDS yourself using IDCAMS commands

```
   DEFINE CLUSTER  (NAME(SYS1.VVDS.Vvolser)  VOLUMES(volser)
NONINDEXED  TRACKS(60 15))   CATALOG(CATALOG.master.name)
```

RAID architecture

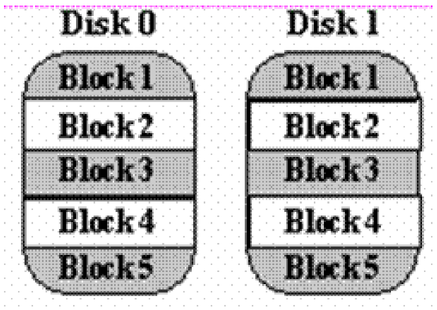
Redundant array of independent disks (RAID) is a direct access storage architecture where data is recorded across multiple physical disks with parity separately recorded, so that no loss of access to data results from the loss of any one disk in the array.
The RAID concept involves many small computer system interface (SCSI) disks replacing a big one. The major RAID advantages are:

- Performance
- Cost
- zSeries compatibility
- Environment

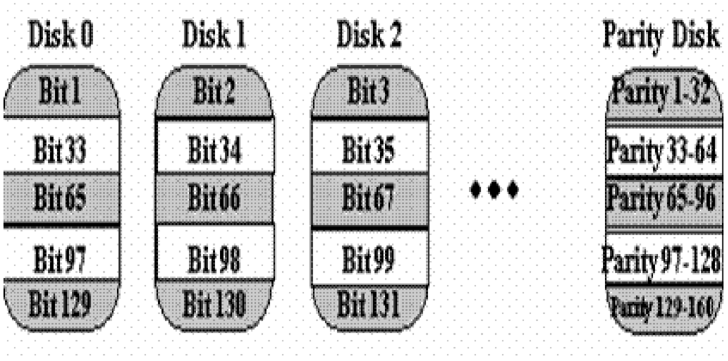
However, RAID increased the chances of malfunction due to media and disk failures and the fact that the logical device is now residing on many physical disks. The solution was redundancy, which wastes space and causes performance problems as "write penalty" and "free space reclamation." To address this performance issue, large caches are implemented. Except for RAID-1, each manufacturer sets the number of disks in an array. An *array* is a set of logically related disks, where a parity applies. Multiple disk drives provides **reliability** via **redundancy**.

Various implementations of RAID Architecture are:

RAID-1 This has just disk mirroring, like dual copy.



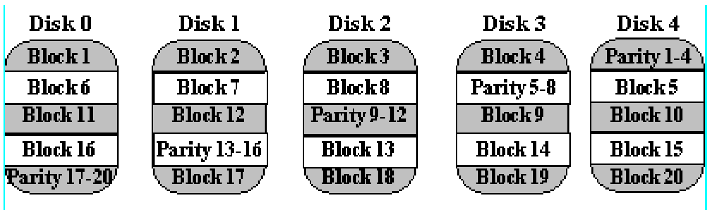
RAID-3 This has an array with one dedicated parity disk and just one I/O request at a time, with intra-record striping. It means that the written physical block is striped and each piece (together with the parity) is written in parallel in each disk of the array. The access arms move together. It has a high data rate and a low I/O rate.



RAID-5 This has an array with one distributed parity (there is no dedicated disk for parities). It does I/O requests in parallel with extra-record striping, meaning each physical block is written in each disk. The access arms move independently. It has strong caching to avoid write penalties; that is, four disk I/Os per write.RAID-5 has a high I/O rate and a medium data rate. RAID-5 is used by the IBM 2105 controller with 8-disk arrays in the majority of configurations.

RAID-5 does the following:

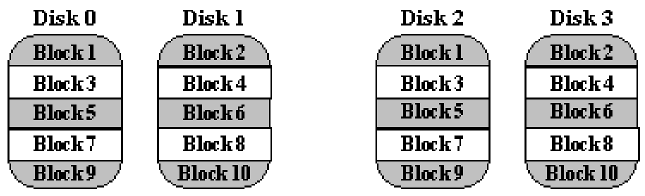
- It reads data from an undamaged disk. This is just one, single disk I/O operation.
- It reads data from a damaged disk, which implies (n-1) disk I/Os, to recreate the lost data where n is the number of disks in the array.
- For every write to an undamaged disk, RAID-5 does four I/O operations in order to store a correct parity block; this is called a write penalty.
- This penalty can be relieved with strong caching and a slice triggered algorithm (coalescing disks updates from cache into a single parallel I/O).
- For every write to a damaged disk, RAID-5 does n-1 reads and one parity write.



RAID-6 This has an array with two distributed parity and I/O requests in parallel with extra-record striping. Its access arms move independently (Reed/Salomon P-Q parity). The write penalty is greater than RAID-5, with six I/Os per write.

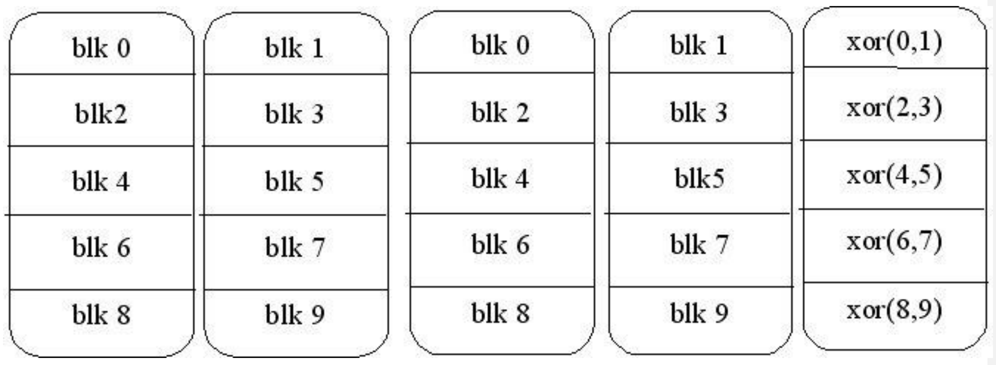
RAID-6+ This is without write penalty (due to log-structured file, or LFS), and has background free-space reclamation. The access arms all move together for writes. It is used by the RVA controller.

RAID-10 RAID-10 has a new RAID architecture designed to give performance for striping and has redundancy for mirroring. RAID-10.



RAID 5 and RAID 10

RAID 5 and RAID 10 are managed by the SSA device adapters. Each loop supports up to 48 disk drives, and each adapter pair supports up to 96 disk drives.
 There are four adapter pairs supporting up to 384 disk drives in total. (RAID ranks are actually split across two eight-packs for optimum performance). You can see there are six RAID arrays: four RAID 5 designated A to D, and two RAID 10 (one 3+3+2 spare and one 4+4).



RAID-10

RAID-10 is also known as RAID 0+1 because it is a combination of RAID 0 (striping) and RAID 1 (mirroring). The striping optimizes the performance by striping volumes across several disk drives (in the ESS Model 800 implementation, three or four DDMs). RAID 1 is the protection against a disk failure provided by having a mirrored copy of each disk. By combining the two, RAID 10 provides data protection and I/O performance.

DASD administration

As a system programmer you make have to handle 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
- Effective Space Management in DASD

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.
3. Using **ICKDSF, initialize the new volume** with a volume serial label, a volume table of contents (VTOC), and a VTOC index.
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.

Example of VATLSTxx

VATDEF IPLUSE (PRIVATE), SYSUSE (PRIVATE)

<u>UTILITIES</u>

1. COMPRESS:

The COMPRESS command compresses **partitioned data sets** on a specified volume. Compressing (degassing) removes unused space between members in a partitioned data set.

Depending on the filtering criteria you specify, you can compress either all or some of the partitioned data sets. This command is useful for compressing system partitioned data sets before you apply maintenance (thus avoiding certain space-related abends).

The following example compresses a selected partitioned data set.

```
//JOB1     JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
 COMPRESS              -
   DYNAM(VOL01) -      /* DYNAM ALLOC VOL 338000      */
   EXCLUDE(SYS1.**)    /* EXCL 'SYS1....' DATA SETS         */ -
                       /* IF THEY MEET THIS CRITERION  */ -
   BY((DSCHA EQ 0))    /* DATA SET WAS BACKED UP      */
 /*
```

Compress partitioned data sets on volume VOL01 if: System data sets are exclude using EXCLUDE(SYS1.**).

1**. <u>ADMINISTRATOR</u>**

ADMINISTRATOR lets you act as a DFSMSdss-authorized storage administrator for the COMPRESS command**.** If you are not authorized to use the ADMINISTRATOR keyword, the command is ended with an error message. To use the ADMINISTRATOR keyword you must have privilege in facility class.

<u>2. DYNAM</u>

DYNAM specifies a dynamically allocated volume whose partitioned data sets, if selected, are to be compressed.

- The volume must be mounted and online.
- You cannot specify a nonspecific volume serial number using an asterisk (*).
- Consider using DYNAM instead of DD statements to allocate DASD volumes. This does not appreciably increase run time and permits easier coding of JCL and command input.

VolSer

Specifies the volume serial number of a DASD volume to be processed. Unit specifies the device type of a DASD volume to be processed. This parameter is optional.

INClude

Dataset name (DSN) specifies the name of a data set eligible to be compressed. Either a fully or a partially qualified data set name can be used. If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(**) is specified, all partitioned data sets are eligible to be selected for compressing.

EXClude

DSN specifies the name of a data set to be excluded from the data sets selected by the INCLUDE keyword. Either a fully or a partially qualified data set name can be used. You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

DDName

DDName Specifies the name of the DD statement that identifies a volume whose partitioned data sets, if selected, are to be compressed. To assure correct processing, each of the DD statements corresponding to a DDNAME (ddn) must identify only one volume serial number.

Password

PASSWORD specifies the passwords DFSMSdss uses for selected password-protected data sets. (Password checking is bypassed for RACF-protected data sets.)
This must be specified only if: You do not have the required RACF DASDVOL or RACF DATASET access. The installation authorization exit does not bypass the checks.

2. RELEASE

Once the compress utility is used then Release is executed for releasing the unused space for use so the space is made available in the volume. If release is not used after the compress command is used then the free space is not available in the volume except for the partition dataset. Releasing Unused Space in Data Sets. The

RELEASE command releases allocated but unused space from all sequential, partitioned dataset.

```
//ARICH4A JOB REGION=5M,NOTIFY=&SYSUID
       //VVPRINT  EXEC PGM=ADRDSSU
       //SYSPRINT DD SYSOUT=*
       //SYSIN    DD *
       RELEASE              -
       DYNAM(VOL07)     -
       INCLUDE(ARICH4.**)
      /*
```

COPY Command

Use the DFSMSdss COPY command to perform data set, volume, and track movement. You can copy data from one DASD volume to another, or move data sets from one DASD volume to another volume of like or unlike device type. You can copy data sets between DASD of like or unlike device type or a full volume or ranges of tracks between DASD of like device type.
However, if you copy a full volume or ranges of tracks, the DASD must be of like device type. The user must specify both the source and target volumes. Only one source volume and one target volume are allowed.

DFSMSdss offers two ways to process COPY commands:

Logical processing is data-set oriented, which means it operates against data sets and volumes independently of physical device format.

Physical processing operates against volumes and tracks, but is oriented toward moving data at the track-image level.

Explanation of COPY Command Keywords

DSN specifies the fully qualified name of a data set whose data set organization is PS, PSU, PO, POU, or null. The data set is processed as follows:

- For a full-volume copy, all of the allocated space for the source data set is copied to the target volume.

- For a data set copy, the function of this parameter is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified.
- For a full-volume copy, all of the allocated space for the source data set is copied to the target volume.
- For a data set copy, the function of this parameter is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified.

ALLExcp

ALLEXCP specifies all data sets whose data set organization is PS, (5) PSU, PO, POU, or null and are empty (the last used block pointer in the data set's VTOC entry is zero).   The data sets are processed as follows:
For a full-volume copy, all of the allocated space for the source data set are copied to the target volume.
For a data set copy, the function of this keyword is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified.

<u>AUTORELBlockaddress</u>

AUTORELBLOCKADDRESS specifies that direct access data sets are to be Automatically processed as being organized by relative block address provided that they were accessed with an OPTCD setting indicating relative block addressing.

BYPASSACS

BYPASSACS specifies that the automatic class selection (ACS) routines are not invoked to determine the target data set class names. To specify BYPASSACS, RACF authorization may be required. If a data set is being renamed, the old name must be specified.

CANCELERROR

CANCELERROR specifies that the copy task be ended for a permanent read error, or that the copy of a data set is ended for a write error.

CATALOG

CATALOG specifies that on a data set copy operation, DFSMSdss is to catalog data sets that it allocates. CATALOG catalogs the target data set as determined by the standard catalog search order. This is the default for VSAM, multivolume data sets, and SMS-managed data sets.
RECATALOG(newcatname) catalogs the target data set in the newcatname catalog.
RECATALOG(*) catalogs the target data set in the same catalog that points to the source data set. If the source data set was not cataloged, the new data set is not cataloged either. After DFSMSdss determines the catalog status of the data set and is changed by other means outside of DFSMSdss, the original catalog status is used.

CONVERT

CONVERT(PDSE(dsn)) specifies that the PDSs listed in the dsn be converted to PDSE
CONVERT (PDS(dsn)) specifies that the PDSE listed in the dsn be converted to PDSs.
1. If the target data set is a PDSE, it must be SMS-managed.
2. The CONVERT keyword is not supported for extended-sequential data sets.

COPYVOLID

COPYVOLID specifies that the volume serial number (VOLID) from the input DASD volume is to be copied to the output DASD volume. This applies to full copy operations and to tracks copy operations if track 0 (zero) is copied.

1. COPYVOLID is required for a full-volume copy operation of an SMS-managed input volume.
2. When the volume serial number is changed by using a COPYVOLID keyword, profiles are not built for the RACF-protected data sets on the target volume or for the RACF DASDVOL for the RACF-protected DASD volume. When the volume serial number on a DASD volume is changed, the operator is notified. The operating system then initiates a demount of the volume.
3. Exercise caution using COPYVOLID in a multiple task job step when two or more of the tasks are using the same output volume. If the output volume is made unavailable by the first task, all succeeding tasks that use the same output volume fail.
4. COPYVOLID cannot be performed if there are permanent I/O errors or if CANCELERROR is specified. If TOLERATE(IOERROR) is honored, however,

COPYVOLID is performed.

CPVOLUME

CPVOLUME specifies that the input and output volumes are VM-format volumes and that the OS-compatible VTOCs must begin on track zero, record five. You must specify the track range to be copied with the TRACKS keyword, as the OS-compatible VTOCs do not describe the extents of any data on the volume. You must also specify the ADMINISTRATOR keyword with CPVOLUME because DFSMSdss cannot check access authorization for VM data.

DATASET

DATASET specifies a data set copy operation using filtering

DELETE

DELETE specifies that for a data set copy DFSMSdss deletes VSAM and non-VSAM data sets from the source volume after a successful copy. This moves, in effect, a data set from one volume to another. The data sets are scratched and uncataloged.

DYNALLOC

DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of data sets. The data sets whose extents are to be relocated are serialized throughout the copy operation..

You can also include keywords like: EXClude , FILterdd , Include, PASsword

FORCE

FORCE specifies that DFSMSdss copy one or more unmovable data sets to a like or unlike device type. Unmovable data sets are those allocated as absolute track (ABSTR) or as unmovable (PSU, POU, DAU, or ISU). FORCE applies to ISAM data sets only when they are allocated with the unmovable attribute. The allocation attribute, unmovable or ABSTR, is carried over to the output volume.

When copying to like devices, DFSMSdss copies the data sets to the same track locations on the target volume. In this case, FORCE is not required if the target volume uses an indexed VTOC, and the space where the unmovable data set is to reside is available. If any of these conditions is not true, DFSMSdss does not copy any unmovable data sets unless FORCE is specified. In this case, DFSMSdss places the unmovable data sets in any available location. When copying to unlike devices, you must specify FORCE. The unmovable data sets are placed in any available location.

FORCECP

FORCECP specifies that checkpointed data sets resident on the SMS volume or volumes can be copied. Checkpoint indications are removed from the target data set. Days is a one- to three-digit number in the range of zero to 255, and specifies the number of days that must have elapsed since the last referenced date before the data set can be copied.

FREESPACE

FREESPACE specifies free space values for DFSMSdss-allocated target VSAM data sets. If this keyword is omitted, the control interval and control area free space are the same as the source data set. CI specifies the percentage of free space to be kept in each control interval during allocation of the data set. CA specifies the percentage of free space to be kept in each control area during allocation of the data set. When omitted, the control area free space is the same as the source data set.

FULL

FULL specifies that an entire DASD volume is to be copied. This is the default. Unallocated tracks are not copied. Unless specified by ALLDATA or ALLEXCP, only the used tracks are copied for sequential data sets, partitioned data sets, and for data sets with unknown data set organization. If the VTOC has errors, all tracks are copied. Used tracks consist of the tracks from the beginning of the data set to the last-used track.

INCAT

INCAT specifies one or more user catalogs to be searched before the standard order of search for locating data sets. STEPCAT and JOBCAT are not allowed for processing SMS-managed data sets. This keyword allows you to identify specific source catalogs. To specify INCAT, RACF authorization may be required. catname specifies a fully qualified catalog name. ONLYINCAT specifies that DFSMSdss is to only search catalogs specified in the INCAT catalog name list. DFSMSdss does not process an SMS-managed data set that is cataloged outside the standard order of search, even if it is cataloged in one of the catalogs specified with the INCAT keyword.

INDDname

ddn specifies the name of the DD statement that identifies a volume to be copied. For a data set copy operation, you can specify multiple names separated by commas. For single-volume data sets, each of the DD statements corresponding to a DDNAME (ddn) must identify only one volume serial number.

When DFSMSdss invokes IEHMOVE to copy multivolume non-VSAM data sets, IEHMOVE requires that the first DD statement in the job stream must identify all input volumes. DFSMSdss requires separate DD statements for the input volumes. To accommodate both requirements, you must code your JCL as follows:

```
 //INDD1  DD  UNIT=(SYSDA,2),VOL=SER=(VOL1,VOL2),DISP=SHR
//INDD2  DD  UNIT=SYSDA,VOL=SER=VOL2,DISP=SHR
```

and code your DFSMSdss control statement:

```
 COPY  LOGINDD(INDD1,INDD2) ...
```

INDYnam

INDYNAM specifies that the volumes to be copied are to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). Only one volume is allowed for a full or tracks copy; one or more volumes are allowed for a data set copy operation. Consider using INDYNAM instead of DD statements to allocate DASD volumes. This does not

appreciably increase run time and permits easier coding of JCL and command input.
Volser specifies the volume serial number of a DASD volume to be copied.
unit specifies the device type of a DASD volume to be copied. This parameter is optional.

OUTDDname

ddn specifies the name of the DD statement that identifies the output DASD

OUTDYnam

OUTDYNAM specifies that the output DASD volume is to be dynamically allocated. The volume must be mounted and online.

OUTTRacks

OUTTRACKS specifies, for a tracks copy operation, the cylinder (cc) and head number (hh) on the output volume to which the tracks from the input volume are to be copied.

PERCENTUtilized

PERCENTUTILIZED specifies that DFSMSdss must stop allocating data sets to the target volumes when the allocated space reaches n percent of the total space on the target volume.

PROCESS

SYS1 specifies that DFSMSdss is to allow data sets with a high-level qualifier of SYS1 to be copied to a preallocated target and that SYS1 data sets can be deleted and uncataloged. SYS1.VVDS and SYS1.VTOCIX data sets cannot be copied, deleted, or uncataloged. To specify PROCESS(SYS1), RACF authorization may be required.

PURge

PURGE specifies, for a full or tracks copy operation, that unexpired data sets on the target volume can be overlaid. If PURGE is not specified and unexpired data sets reside on the target tracks, the copy operation is ended.
For a data set copy operation, PURGE specifies that unexpired source data sets are to be deleted after they are successfully copied. This keyword is only valid when the DELETE keyword is also specified.

READIOPacing

READIOPACING specifies the pacing (that is, I/O delay) to be used for DFSMSdss DASD read channel programs. You can use this keyword to allow more time for other applications to complete I/O processing.

REBLock

REBLOCK specifies that DFSMSdss is to reblock one or more of the selected sequential or partitioned data sets.

REPlace

REPLACE specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If no preallocated target is found, DFSMSdss attempts to allocate a data set.

SHAre

SHARE specifies that DFSMSdss is to share the data sets to be copied for read access with other programs.SHARE and FULL are mutually exclusive; you cannot specify these keywords together.

SPHERE

SPHERE specifies that, for any VSAM cluster copied, all associated AIX clusters and paths are to be copied. Individual names of sphere components do not need to be specified. Only the base cluster name is required. If output volumes are specified, the volumes on which the AIX clusters reside do not need to be specified.

STORCLAS

STORCLAS specifies the storage class that you want to replace the source storage class as input to the ACS routines.

STORGRP

STORGRP specifies that all of the online volumes in the storage group be dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing. Up to 255 storage group names may be specified. Specifying STORGRP with a storage group name is equivalent to specifying LOGINDYNAM with all the online volumes in the storage group included in the list.

TGTAlloc

TGTALLOC	Specifies how DFSMSdss is to allocate the target data set.
BLK	By blocks
CYL	By cylinders
TRK	By tracks
SOURCE/SRC	With the same space allocation type as that of the source data set

TGTGDS

TGTGDS specifies in what status, during a data set operation, that DFSMSdss is to place non-pre-allocated SMS-managed GDG data sets:

DEFERRED specifies that the target data set is to be assigned the DEFERRED status.

ACTIVE specifies that the target data set is to be assigned the ACTIVE status, for ex. rolled into the GDG base.

ROLLEDOFF specifies that the target data set is to be assigned the rolled-off status.

SOURCE/SRC specifies that the target data set is to be assigned the same status as that of the source data set.

TOLerate

TOLERATE specifies that DFSMSdss tolerates certain error conditions. For a copy operation (except of a user catalog or loadlib) in which a utility performs the copy, this keyword is ignored. ENQFailure specifies that source and target data sets are to be processed even though shared or exclusive access fails.

TRACKS

TRACKS specifies ranges of tracks to be copied (that is, a tracks copy).

TTRAddress

TTRADDRESS identifies the direct access data sets whose names match the fully or partially Qualified names specified (dsn).

UNCATalog

UNCATALOG specifies that DFSMSdss is to uncatalog but not scratch successfully copied non-VSAM data sets that are currently cataloged on the source volume. Any non-SMS, non-VSAM data set that has a high-level qualifier of SYS1 cannot be uncataloged unless PROCESS(SYS1) is specified. UNCATALOG is ignored for VSAM data sets and SMS-managed non-VSAM data sets.

VOLcount

VOLCOUNT specifies the method DFSMSdss uses to determine the number of volumes (volume count) for allocating the SMS target data set for a copy operation of VSAM or non-VSAM data sets.

WAIT

WAIT specifies to DFSMSdss the length of the wait in seconds, and the number of passes to be made through the list of selected data sets to obtain control of a data set for COPY DATASET command.

WRItecheck

WRITECHECK specifies that the data copied is to be verified for successful completion. This keyword increases the overall elapsed time.

Data Integrity Considerations for Full or Tracks Copy Operation

For a full or tracks copy operation, DFSMSdss serializes the VTOC to preclude DADSM functions (such as ALLOCATE, EXTEND, RENAME, and SCRATCH) from changing the contents of the VTOC on the volume during the copy operation. Data sets are not serialized on these full or tracks operations. Therefore, some data sets might be opened by other jobs during the copy, resulting in copies of partially updated data sets. You can minimize this possibility by performing the copy when there is low system activity. Full data integrity can only be guaranteed by performing copy operations by data set when TOL(ENQF) or SHARE are not specified.

Example 1:

```
//JOB1       JOB          accounting information,REGION=nnnnK
//STEP1      EXEC         PGM=ADRDSSU
//SYSPRINT          DD    SYSOUT=A
//DASD1      DD    UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2      DD    UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN      DD    *
   command input (see Examples 1A and 1B below)
/*
```

Example 1A: A Full Copy Operation

```
  COPY  INDDNAME(DASD1) OUTDDNAME(DASD2) -
        ALLDATA(*) ALLEXCP CANCELERROR
```

Example 1B: A Tracks Copy Operation

```
  COPY  TRACKS(1,0,1,15) INDDNAME(DASD1) -
        OUTDDNAME(DASD2) CANCELERROR
```

The data from DASD volume 111111 is to be copied to DASD volume 222222. For the full copy operation, all allocated space in sequential or partitioned data sets and data sets with a data set organization that is null is copied (ALLDATA(*)). The preceding applies only to data sets that are not empty. For data sets that are empty, DFSMSdss copies all data within the allocated space (ALLEXCP). The copy operation is to be ended if a permanent read error occurs (CANCELERROR).

Example : A Tracks Copy with Track Relocation

```
//JOB2     JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
   COPY TRACKS(1,0,1,14) /* SOURCE TRACKS  */ -
   OUTTRACKS(3,0)      /* TARGET TRACKS  */ -
   INDYNAM(338000)     /* ALLOC VOL 338000 DYNAMICALLY  */ -
   OUTDYNAM(338001)          /* ALLOC VOL 338001 DYNAMICALLY  */ -
   CANCELERROR                 /* STOP ON INPUT ERROR      */ -
   WRITECHECK           /* VERIFY DATA WRITTEN TO OUT VOL */
/*
```

Example : A Data Set Move--Only Single Volume Data Sets

```
//JOB3     JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    COPY data set(         -
    INCLUDE(USER1.**)          /* FILTER ON DS W/1ST LEV Q USER1 */ -
    BY(MULTI,=,NO))      /* FILTER ON SINGLE VOLUME        */ -
    INDYNAM (338000)           /* ALLOC VOL 338000 DYNAMICALLY   */ -
    OUTDYNAM(338001)           /* ALLOC VOL 338001 DYNAMICALLY   */ -
    DELETE
/*
```

Example : Using the COPY Command to Convert to SMS

```
//JOB9      JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
    COPY  -
      DS(INC(**))  -
      LOGINDYNAM ( -
      (338001) -
      (338002) -
       ) -
       STORCLAS(DB2PERF) -
       MGMTCLAS(DBBACKUP) -
       BYPASSACS(**) -
       DELETE -
       PURGE
/*
```

In the first step of this example, all of the data sets on the non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. DELETE and PURGE are used to avoid duplicate catalog entries. The ACS routines are not invoked to determine the target data set classes for this copy operation. The storage and management classes are provided by the users using the STORCLAS and MGMTCLAS keywords. In addition, BYPASSACS(**) is specified in order to suppress calls made to the ACS routines. All data sets that are supported by SMS are given the new storage and management classes

```
//JOB1      JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
   COPY  -
    DS(INC(**))  -
    LOGINDYNAM ( -
       (338001) -
       (338002) )  -
    RENUNC(AUG0387)
/*
```

In this next step all data sets on the non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. The RENUNC command is used to avoid duplicate catalog entries. The ACS routines select a target storage and management class for each data set. Those data sets that cannot be SMS-managed (storage class ACS routine returns a null storage class) are not copied because no output volume is specified. Each data set that is copied is given a new high-level qualifier (AUG0387) and automatically cataloged.

Example : A Data Set Copy Using CONVERT PDSE

```
//JOB2      JOB  accounting information,REGION=nnnnK
//STEP1    EXEC PGM=ADRDSSU
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
      COPY  -
      DS(INC(USER.PDS.**))  -
      LOGINDYNAM ( -
         (338001) -
         (338002) -
```

```
        ) -
    CONVERT (PDSE(**)) -
    RENUNC (USER.PDS.**, USER.PDSE.**)
/*
```

In this step all data sets with the first two qualifiers of USER.PDS on non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. CONVERT PDSE is used to convert data sets to PDSE. The RENUNC keyword is used to avoid duplicate catalog entries. The ACS routines select a target storage and management class for each data set. Each data set that is copied and converted is given a new secondary qualifier (PDSE) and automatically cataloged.

DEFRAG

With the DEFRAG command, you can relocate data set extents on a DASD volume to reduce or eliminate free-space fragmentation and print a report about free space and other volume statistics. Also, you can specify which data sets, if any, are to be excluded from data-set-extent relocation.

DYNAM

DYNAM specifies that the volume to be processed is to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).

EXClude

dsn specifies a fully or partially qualified name of a data set to be excluded from the DEFRAG operation. You can specify either a cluster or component name for VSAM data sets.

FORCECP

FORCECP specifies that extents of checkpointed data sets resident on the SMS volume or volumes can be moved. The checkpoint indication is left in place for IMS GSAM data sets, as the data set is still usable for a restart.

FRAGMENTATIONIndex

FRAGMENTATIONINDEX specifies that the DEFRAG operation is to end if the fragmentation index is less than n, where n is a 1- to 3-digit number. DFSMSdss prefixes your number, n, with a decimal point. For ex. 1 becomes .1, 999 becomes .999, 001 becomes .001, and so forth.

MAXmove

N is a 1- to 6-digit number that specifies that DFSMSdss is to attempt to assemble up to n free tracks in a contiguous area. If n is greater than the total number of free tracks on the volume, for example MAXMOVE(999999), a message is issued and the DEFRAG operation adjusts n to the number of free tracks.

PASsword

PASSWORD specifies the passwords DFSMSdss is to use for password-protected data sets. (Password checking is bypassed for RACF-protected data sets.)

WRITECHECK

WRITECHECK specifies that the data moved by the DEFRAG operation is to be verified for successful completion. This keyword increases the overall elapsed time.

Example : A DEFRAG Operation with Excluded Data Sets

```
//JOB1     JOB   accounting information,REGION=nnnnK
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD    SYSOUT=A
//DASD     DD    UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//A1       DD    DSN=USER2.EXCLUDE,DISP=SHR
//SYSIN    DD    *
  command input (see examples 1A and 1B below)
/*
```

Example 1A: With the Names of Excluded Data Sets in the Input Stream

```
   DEFRAG  DDNAME(DASD) -
         EXCLUDE(LIST(USER2.**.LIST,*.LOAD))
```

Example 1B: With the Names of Excluded Data Sets in a Data Set

```
   DEFRAG  DDNAME(DASD) -
         EXCLUDE(DDNAME(A1))
```

In examples 1A and 1B, DASD volume 111111 is defragmented. All data sets whose first and last qualifiers are USER2 and LIST, respectively, are to be excluded from this operation, as are data sets with two qualifiers whose second qualifier is LOAD. In example 1B, cataloged data set USER2.EXCLUDE contains a single card-image record with the following in columns 2 through 72:

```
USER2.**.LIST,*.LOAD
```

DUMP

With the DUMP command, you can dump DASD data to a sequential data set. The storage medium for the sequential data set can be a tape or DASD. You can dump data sets, an entire volume, or ranges of tracks.

DFSMSdss offers two ways to process DUMP commands:

- Logical processing is data-set oriented, which means it operates against data sets independently of physical device format.
- Physical processing can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.

The processing method is determined by the keywords specified on the command. DFSMSdss logical dump processing cannot be used to process partitioned data sets containing location-dependent information that does not reside in note lists or in the directory. Furthermore, DFSMSdss cannot be used to dump migrated data sets.

Integrated catalog facility catalogs should not have a high-level qualifier of SYSCTLG because this causes DFSMSdss to treat them as CVOLs.

Special Considerations for Dump

The following special considerations apply when you are performing a dump operation:

- Extended-format VSAM data sets are not supported for physical data set processing.
- You cannot do a physical data set dump of a KSDS with keyranges.

CONVERTING DATA TO AND FROM SMS MANAGEMENT

DFSMSdss is the primary tool for converting data to and from SMS management. Conversion can be done with or without data movement.

Conversion to SMS Management

When you convert data to SMS management, the first thing to consider is whether to convert data sets with or without data movement. If you have SMS-managed volumes with sufficient free space, you can convert data sets by simply moving them from non-SMS-managed volumes to SMS-managed volumes. The same is also true if you are converting data from SMS-management. If, however, you do not have sufficient free space on your SMS-managed volumes to convert by data movement, you might have to convert data sets without data movement.
Regardless of how you convert to SMS management, you must determine the eligibility for conversion of your data sets and volumes prior to conversion.

Data Sets Ineligible for Conversion to SMS

Using the CONVERTV command with the SMS and TEST keywords identifies ineligible data sets without actually converting any data.

Data Sets Ineligible for Conversion from SMS

The following data sets cannot be converted from SMS management:

1. Partitioned data set extended (PDSE)
2. HFS data sets

Using the CONVERTV command with the NONSMS and TEST keywords identifies ineligible data sets without actually converting them.

Volume Eligibility for Conversion to SMS

A volume is eligible for conversion if it:

Is a DASD volume
Is online
Has a VTOC
Conversion by Data Movement

By using the logical data set COPY or RESTORE command, you can move data sets between non-SMS-managed and SMS-managed volumes.When moving data sets to SMS-managed volumes, COPY and RESTORE commands invoke ACS to assign classes to the data sets.

This type of conversion to SMS allows the data sets to be placed on the most appropriate SMS-managed volume.

Converting to SMS Management by Data Movement

When moving data sets to SMS-managed volumes, you can use the COPY or RESTORE command. You can specify storage and management class names with the STORCLAS and MGMTCLAS keywords. You can also specify output volumes with OUTDDNAME and OUTDYNAM

Conversion from SMS Management by Data Movement

To take a data set out of SMS management with the COPY or DUMP/RESTORE command, you should specify the BYPASSACS and NULLSTORCLAS keywords. This forces DFSMSdss to make the data set non-SMS-managed

Conversion without Data Movement

Convert data sets and the volumes they reside on without moving data by using the DFSMSdss CONVERTV command

Simulating Conversion

Before you convert a volume to SMS management, you should simulate the conversion to ensure that all the data sets on the volume are eligible for conversion to SMS. In addition, simulating conversion shows you the classes ACS would assign to the data sets eligible for conversion we can simulate conversion by using the CONVERTV command with the SMS and TEST keywords. Simulated conversion creates a report that identifies data sets ineligible for conversion

Preparing a Volume for Conversion

Before you convert a volume to SMS management, you should reduce the amount of activity to the volume being converted. The CONVERTV command with the

SMS keyword automatically places the volume in a state of reduced activity before doing the actual conversion.

```
//ARICH4A  JOB CLASS=A,
//MSGLEVEL=(1,1),NOTIFY=&SYSUID,REGION=0M
//*JCL TO CONVERT /DEFRAG     EXEC PGM=ADRDSSU
//SYSPRINT   DD   SYSOUT=*
//SYSIN      DD *
          CONVERT                    -
          DYNAM(VOL20) –
           NONSMS
/*
//
```

LAB Exercises

1. COMPRESS

Example for compressing volume VOL20

```
//ARICH9Q JOB COMPRESS', NOTIFY=ARICH9,REGION=0M, //USER=ARICH3
//COMPRESS EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
     COMPRESS -
     DYNAM( -
     (VOL20)) -
     INCLUDE(*.**)
/*
```

2. RELEASE

Releasing the unused space in volume VOL20

```
//ARICH9N  JOB NOTIFY=ARICH9,REGION=0M,USER=ARICH3
//COMPRESS   EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     RELEASE -
```

```
    DYNAM(-
    (VOL20)) -
    ADMIN -
    INCLUDE(*.**)
/*
```

3. DEFRAG (Dfragmentation of multiple volumes.)

```
//ARICH9S JOB ,,CLASS=A,REGION=0M,USER=ARICH3,
//  MSGLEVEL=(1,1),NOTIFY=ARICH9,TIME=(1)
//************DEFRAGGING VOL01**************
//DEFRAG EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//VOL  DD VOL=SER=DEV003,UNIT=3390,DISP=OLD
//VOL1 DD VOL=SER=VOL03,UNIT=3390,DISP=OLD
//VOL2 DD VOL=SER=VOL04,UNIT=3390,DISP=OLD
//VOL3 DD VOL=SER=VOL05,UNIT=3390,DISP=OLD
//VOL4 DD VOL=SER=VOL06,UNIT=3390,DISP=OLD
//SYSIN DD *
//  DEFRAG DDNAME(VOL.VOL1,VOL2,VOL3,VOL4)
//
```

4. DELETION OF Datasets Deleting unwanted datasets using IDCAMS

```
//ARICH08A JOB ,NOTIFY=&SYSUID,
//   MSGCLASS=X
//STEP1    EXEC  PGM=IDCAMS
//DD1      DD VOL=SER=VOL05,UNIT=3390,DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD   *
   DELETE -
   MV02.SPFLOG1.LIST -
   FILE(DD1) -
   NVR
/*
//
```

What is a Catalog

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. 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.

Types of Catalogs

- Integrated Catalog Facility Catalogs
- VSAM Catalogs
- OS CVOL (Control volume) Catalogs

Advantages of ICF Catalogs:

- ICF catalogs can be updated faster than VSAM catalogs or OS CVOLs.
- An ICF catalog can have data sets cataloged on any number of volumes.
- The catalog information that requires the most frequent updates is physically located in the VVDS on the same volume as the data sets, allowing faster access.
- When defining an ICF catalog, you have more control because you can specify parameters that cannot be specified in a VSAM catalog.
- Maintainability is improved by simpler backup and recovery procedures.

Introduction to ICF Catalogs

An ICF catalog consists of two separate kinds of data sets: a basic catalog structure (BCS); and a VSAM volume data set (VVDS).
The BCS can be considered the catalog, whereas the VVDS can be considered an extension of the volume table of contents (VTOC).

Basic catalog structure (BCS)

It is a VSAM key-sequenced data set. It uses the data set name as a key to store and retrieve data set information. The BCS contains the information about where a data set resides. That can be a DASD volume or tape or any other storage medium.

The BCS portion of the ICF catalog contains the static information about the data set. 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). One 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.
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.
For non-VSAM data sets that are not SMS-managed, all catalog information is contained within the BCS. For other types of data sets, there is other information available in the VVDS.

VSAM volume data set (VVDS): It is a VSAM entry-sequenced data set. There are three types of entries in a VVDS.

One VSAM volume control record (VVCR): First logical record in a VVDS contains information for management of DASD space and the names of the BCSs that have data sets on the volume.

Multiple VSAM volume records (VVR) : Contain information about a VSAM data set residing on the volume and data set characteristics, SMS data, and extent information. Number of VVRs varies according to the type of data set (KSDS, ESDS, RRDS) and the options specified for the data set.

Multiple non-VSAM volume records (NVR) : Contain information about the non-VSAM data set on that Volume equivalent to a VVR for SMS-managed and non-VSAM data sets.
The VSAM volume data set (VVDS) 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. A VVDS resides on every volume which contains a VSAM or SMS-managed data set cataloged in an integrated catalog facility catalog. A VVDS may have data set information about data sets cataloged in distinct BCSs.

VVDS characteristics

The VVDS is a VSAM entry-sequenced data set (ESDS) that has a 4 KB control interval size.

A VVDS is recognized by the restricted data set name:
SYS1.VVDS.Vvolser Volser is the volume serial number of the volume on which the VVDS resides. You can explicitly define the VVDS using IDCAMS, or it is implicitly created after you define the first VSAM or SMS-managed data set on the volume. An explicitly defined VVDS is not related to any BCS until a data set or catalog object is defined on the volume. As data sets are allocated on the VVDS volume, each BCS with VSAM data sets or SMS-managed data sets residing on that volume is related to the VVDS.
The VVDS is composed of a minimum of two records: VVCR, VVDS self-describing volume record. The first record in the VVDS always is the VVCR, which contains control information.
The second logical record in the VVDS is the VVDS self-describing VVR. This self-describing VVR contains information that describes the VVDS itself as a VSAM ESDS.
The remaining logical records in the VVDS are the VVRs and NVRs for the data sets residing on the volume. The hexadecimal RBA of a record is used as its key or identifier.

Volume Table of Contents: (VTOC)

The VTOC and the VTOC index are system data sets which maintain extent and allocation information for a volume. The VTOC is used to find empty space for new allocations and to locate non-VSAM non-SMS managed data sets. For all VSAM data sets, and for SMS-managed non-VSAM data sets, the VTOC is used to obtain information not kept in the VVDS.

Catalogs by function

1. Master catalog
2. User catalog

A particular case of a user catalog is the volume catalog, which is a user catalog containing only tape library and tape volume entries. There is no structural difference between a master catalog and a user catalog. What make a master catalog different is how it is used, and what data sets are cataloged in it.

What is a MASTER Catalog

Each system has one active master catalog. One master catalog can be shared between different MVS images. It does not have to reside on the system residence volume. The master catalog for a system must contain entries for all user catalogs and their aliases that the system uses. Also, all SYS1 data sets must be cataloged in the master catalog for proper system initialization. During a system initialization, the master catalog is read so that system data sets and catalogs can be located.

Identifying the master catalog for IPL

At IPL, you must indicate the location (volser and data set name) of the master catalog.

This information can be specified in:

SYS1.NUCLEUS member SYSCATxx (default is SYSCATLG) or
SYS1.PARMLIB/SYSn.IPLPARM member LOADxx

To minimize update activity to the master catalog, and to reduce the exposure to breakage, we strongly recommend that only SYS1 data sets, user catalog connector records, and the aliases pointing to those connectors should be in the master catalog.

User catalogs

The difference between the master catalog and the user catalogs is in the function. User catalogs should be used to contain information about your installation cataloged data sets other than SYS1 data sets. There are no set rules as to how many you should have or how large they should be.It depends entirely on your environment. Cataloging data sets for two unrelated applications in the same catalog creates a single point of failure for them that otherwise may not exist.

Aliases

Aliases are used to tell catalog management which user catalog your data set is cataloged in. You define an appropriate alias name for a user catalog in the master catalog. Next, match the high-level qualifier (HLQ) of your data set with the alias. This identifies the appropriate user catalog to be used to satisfy the request. To define an alias, use the IDCAMS command **DEFINE ALIAS.**

Multilevel aliases

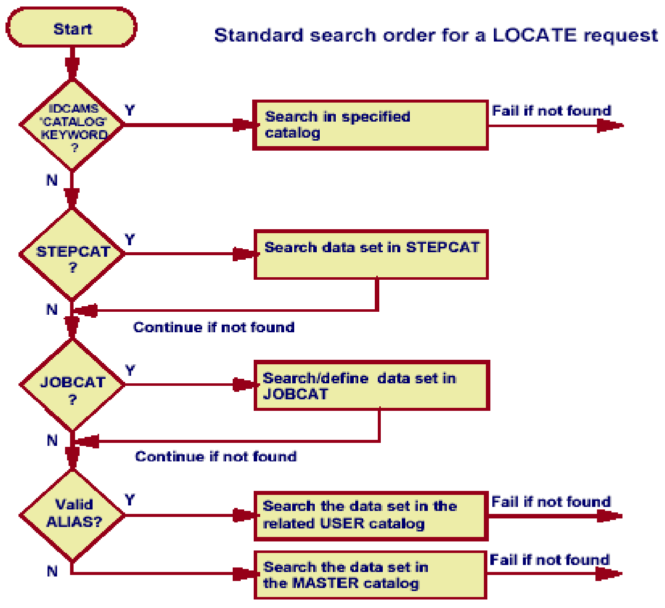
You can augment the standard catalog search order by defining multilevel catalog aliases. A multilevel catalog alias is an alias of two or more high-level qualifiers. You can define aliases of up to four high-level qualifiers. However, the multilevel alias facility should only be used when a better solution cannot be found. The need for the multilevel alias facility can indicate poor data set naming conventions. Before defining multilevel aliases, review your data set naming conventions. A multilevel catalog alias is an alias of two or more high-level qualifiers. You can define aliases of up to four high-level qualifiers. The multilevel catalog alias facility can only be used with integrated catalog facility catalogs. It should be used only if there is no better solution.

Ex: Alias PROJECT1.TEST for catalog SYS1.ICFCAT.PRO1TEST,
Alias PROJECT1.PROD for catalog SYS1.ICFCAT.PRO1PROD, and
Alias PROJECT1 for catalog SYS1.ICFCAT.PROJECT1.

If the alias search level is 2, then the above datasets are cataloged as follows.

Data Set	**Catalog**	**Reason**
PROJECT1.UTIL.CNTRL	SYS1.ICFCAT.PROJECT1	The second qualifier is neither TEST nor PROD
PROJECT1.PROD.DATA	SYS1.ICFCAT.PRO1PROD	There are two qualifiers, and the two qualifiers form the alias PROJECT1.PROD.
PROJECT1.PROD	SYS1.ICFCAT.PROJECT	There is only one qualifier in this data set name. PROJECT1 PROD is not a qualifier, so it is not used in the multilevel alias search.
PROJECT1.TEST.CNTRL	SYS1.ICFCAT.PRO1TEST	There are two qualifiers, and the two qualifiers from the alias PROJECT1.TEST
PROJECT1.TEST.AB	SYS1.ICFCAT.PRO1TEST	The first two qualifiers are used as the alias, since the search level is 2. The third qualifiers is not used in the search.

Catalog Search Order:


Start
Standard search order for a LOCATE request
IDCAMS 'CATALOG' KEYWORD ?
Y
N
Search in specified catalog
Fail if not found
STEPCAT ?
Search data set in STEPCAT
Continue if not found
JOBCAT ?
Search/define data set in JOBCAT
Continue if not found
Valid ALIAS?
Search the data set in the related USER catalog
Fail if not found
Search the data set in the MASTER catalog
Fail if not found

Defining a Data Set	Locating a Data Set
1. Use the catalog named in the IDCAMS CATALOG parameter, if coded.	1. Use catalog named in IDCAMS CATALOG parameter if coded. If the data set is not found, fail the job.
2. Otherwise, use the catalog named in the STEPCAT DD statement	2. Otherwise, search all catalogs specified in a STEPCAT DD statement in order.
3. If no STEPCAT, use the catalog named in the JOBCAT DD statement.	3. If not found, search all catalogs specified in a JOBCAT DD statement in order.
4. If no JOBCAT& the high-level qualifier is a catalog alias, use the catalog identified by the alias or the catalog whose name is the same as the high-level qualifier of the data set.	4. If not found, and the high-level qualifier is an alias for a catalog, search the catalog or if the high-level qualifier is the name of a catalog, search the catalog. If the data set is not found, fail the job.
5. If no catalog has been identified yet use the master catalog.	5. Otherwise, search the master catalog.

For SMS-managed data sets, JOBCAT and STEPCAT DD statements are not allowed, and cause job failure. You can use RACF to prevent the use of the CATALOG parameter and restrict the ability to define data sets in the master catalog.

SYSCATxx or LOADxx Definitions:

	Volser type	Catalog level	Alias low limit	CAS task name	Catalog high level qualifier	Tape volume catalog
Bytes	1	7	8	9	11	55

Volser is the volume serial number of the master catalog's volume. This must be in the first 6 bytes of the record.

catalog type is the catalog type and SYS% facility default, where:

blank or 0 Indicates a VSAM master catalog. In this case, leave bytes 8, 9, and 10 blank. Only the volume serial number and the catalog name can be specified for a VSAM master catalog.

1 indicates an integrated catalog facility catalog, with the SYS% facility turned off,

2 indicates an integrated catalog facility catalog with the SYS% facility turned on.

This value must be defined in the seventh column of the record. The setting for the SYS% conversion facility can be changed after IPL for a current session with the MODIFY CATALOG,SYS% operator command.

alias level specifies the multilevel alias search level. The default is 1, and the maximum is 4.

This value must be defined in the eighth column of the record. If you want the default value of 1, either specify 1 or leave the eighth column blank. This value can later be changed for a current session with the MODIFY CATALOG,ALIASLEVEL operator command.

CAS task low limit specifies the catalog address space service task lower limit, in hexadecimal. The value can range from hexadecimal 18 to hexadecimal B4, but the default is hexadecimal 3C. To specify hexadecimal B4, you can enter C'B4' or X'C2F4'. This value must be defined in the ninth and tenth columns. Specify the default value (either explicitly, or by leaving these two columns blank). If the catalog address space needs more services tasks, it creates them.

catalog name specifies the name of the master catalog. The name can be up to 44 characters. To change to a different master catalog, you have to IPL the system and specify a different SYSCATxx member.

tape volume catalog high level qualifier (optional, otherwise blank) specifies the first name qualifier of the volcat, 1 to 8 characters. See "Defining Names for a Tape

Volume Catalog" in topic 3.5. Leave this blank if you have no tape volume catalog member.

Ex:

```
//SYSCAT  EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY
//SYSUT2
DD DSN=SYS1.NUCLEUS(SYSCATLG),DISP=SHR,DCB=(RECFM=U)
//SYSUT1   DD *
  SYSRES11  SYS1.MASTERA.ICFCAT
/*
```

For LOADxx member the format is the same except that SYSCAT is appended in the beginning.

	Volser type	**Catalog level**	**Alias low limit**	**CAS task name**	**Catalog high level qualifier**	**Tape volume catalog**
Bytes	1	10	16	18	20	64

Defining the Catalog Data Space Cache (COFVLFxx):

The COFVLFxx member of SYS1.PARMLIB determines which catalogs are eligible for the catalog data space (VLF) cache. The syntax of the CLASS statement for defining the catalog data space cache is:

```
CLASS NAME(IGGCAS)  EMAJ(catname)  [EMAJ(catname)]  [...]
        MAXVIRT({4096|size})
```

NAME(IGGCAS) specifies the class name for the catalog data space cache.

EMAJ(catname) specifies the name of an integrated catalog facility catalog eligible for catalog data space caching.

MAXVIRT({4096|size}) specifies the maximum virtual storage that VLF can use to cache catalog records.

Recording SMF Records for Catalog Events (SMFPRMxx): You can use the system management facilities to record certain catalog events. These records can be used to keep track of catalog activity, and can be used during catalog recovery.

SMF Record Types used with the Integrated Catalog Facility

Type	Description
36	Records successful exports of integrated catalog facility catalogs. Contains the date and time of the export, and information needed for importing the backup copy. Written on successful completion of EXPORT.
60	Records any changes to VVDS records, both VVRs and NVRs written when a VVDS record is inserted, updated or deleted.
61	Records changes to the BCS during DEFINE processing. Written when a record is inserted or updated.
65	Records changes to the BCS during DELETE processing. Written when a record is deleted or updated.
66	Records changes to the BCS during ALTER processing. Written when a record is updated, inserted or deleted.

DEFINING A CATALOG:

You can define a catalog using the DEFINE USERCATALOG command. To specify integrated catalog facility catalogs, use the ICFCATALOG parameter. DEFINE USERCATALOG VOLCATALOG defines a tape volume catalog that only contains tape library and tape volume entries. A catalog should be defined with enough space so that it does not grow into secondary extents. Excessive secondary extents can decrease catalog performance. An integrated catalog facility catalog can have up to 123 extents but can only occupy space on a single volume.

If you are defining the BCS on a volume which does not contain a VVDS, a VVDS is also defined and allocated with the BCS. This "implicitly defined" VVDS is allocated with primary and secondary space of 10 tracks. If you want to define a VVDS with a space allocation different from the default, you must define a VVDS on the volume (using DEFINE CLUSTER) before you define the BCS.

Determining the Size of a BCS:

The following table will help you in arriving at a BCS size. Estimated Space Needed for each Type of Data Set or Object.

Data Set or Object Type	Number of Bytes
Generation Data Group	350
Generation Data Set	200
Alias Entry	150
Non-VSAM Data Set or OAM Object Collection	200
User Catalog Connector	200
VSAM Entry-Scheduled, Linear or Relative Record Data	400
VSAM Key-Sequenced Data Set	650
Alternate Index	400
Path	190

1. Estimate the number of each type of data set or object which will be cataloged in the BCS. Using these figures, determine the total space requirement, in bytes, for the BCS. This figure is the minimum amount of space that the BCS requires.

2. Increase this quantity by the amount of free space required, and add additional space to allow for growth and any inaccuracies in the calculation. For example, if you define free space as 20% of each control interval and area, multiply by 1.7.

3. Divide the total number of bytes by 1024 to determine the number of kilobytes, or by 1048576 to determine the number of megabytes. Round the result up to the nearest whole integer, and specify KILOBYTES or MEGABYTES as appropriate.

4. Choose an appropriate secondary allocation. It is best if the secondary allocation is larger than the equivalent of one cylinder, so that the control area is defined as one cylinder.

Estimating Space Requirements for the VVDS:

Data Set Type	SMS-managed Volume	Non-SMS-managed Volume
VSAM key-sequenced data set or alternate index	530	480
VSAM entry-sequenced or relative record data set	370	320
VSAM linear data set	340	290
non-VSAM data set	100	0

These numbers assume 3 qualifier data set and catalog names of 8 characters per qualifier. Key-sequenced data sets and alternate indexes are assumed to have 64 character keys. SMS class names are assumed to be eight characters.

To estimate the total amount of space needed:

1. Add the space required by each data set.
2. Add 8 kilobytes for use by the VVDS.
3. Multiply the result by 1.2 to leave room for errors in the calculation.
4. Divide the result by 1024 to determine the number of kilobytes, and round up to the nearest integer.

The following example defines a catalog with the attributes recommended in this publication. The catalog defined, SYS1.ICFCAT.TEST, is placed on volume SYS305, and is allocated with 15 megabytes primary and secondary space.

```
//DEFCAT   JOB ...
//DEFCAT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
     DEFINE USERCATALOG  ( NAME(SYS1.ICFCAT.TEST) -
             MEGABYTES(15 15)  VOLUME(SYS305)        -
             ICFCATALOG   FREESPACE(10 10)       -
             STRNO(3) IMBED  REPLICATE )           -
            DATA( CONTROLINTERVALSIZE(4096) -
            BUFND(4)  )  INDEX( BUFNI(4)  )
/*
```

STRNO: No. of Concurrent read requests to a BCS

BUFND Specifies the number of buffers for transmitting data between virtual and auxiliary storage. The default, STRNO+1, is usually adequate.

BUFNI Specifies the number of buffers for transmitting index entries between virtual and auxiliary storage. The default, STRNO+2, is adequate for 3 levels of index in the catalog. If the catalog index exceeds 3 levels of index, the minimum BUFNI equals the number of levels of index - 1 + STRNO.

IMBED Placing sequence-set records adjacent to control areas When you define a catalog, the default is that the sequence set index record for each control area is to be imbedded on the first track of the control area.

This reduces disk arm movement because it is not necessary to do separate seeks to locate both the sequence set index record and the data record.

One arm movement allows VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored.

REPLICATE Replicating index records to fill a track. You can specify that each index record be replicated, that is, written on a track as many times as it fits. Replication reduces the time lost waiting for the index record to reach the read/write head as the disk rotates (rotational delay).

STRNO Specifies the numbers of concurrent requests.
BUFFERSPACE Required buffer space for your catalog. It is determined by catalog.
FREESPACE an adequate value allows catalog updates without an excessive number of control interval and control area splits.

Defining an alias:

The following job defines one aliases for SYS1.ICFCAT.TEST, USER01

```
  //DEFALIAS JOB ...
  //ALIAS    EXEC PGM=IDCAMS
  //SYSPRINT DD SYSOUT=A
  //SYSIN    DD *
        DEFINE ALIAS(NAME(USER01)  RELATE(ARICH.UCAT.TEST))
 /*
```

The NAME parameter identifies the alias, and the RELATE parameter identifies the catalog for which the alias is being defined. Hence any Dataset having a HLQ of USER01 will be catalogued in ARICH.UCAT.TEST

Defining a VVDS

```
//DEFVVDS JOB ...
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
     DEFINE CLUSTER (NAME(SYS1.VVDS.VSER003) -
           TRACKS(10 10)  VOLUMES(SER003)  NONINDEXED)
 /*
```

MANAGING THE CATALOGS:

Listing a catalog

Use IDCAMS LISTCAT command to extract information from the BCS and VVDS for:

- Aliases
- User catalog connectors in the master catalog
- Catalogs self describing records
- VSAM data sets
- Non-VSAM data sets
- Library entries and volume entries of a volume catalog
- Generation data sets
- Alternate index and path for a VSAM cluster
- Page spaces

Listing the Contents of a Catalog:

1) You can list catalog records using the access method services LISTCAT command

```
//LSTSDENT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
    LISTCAT ALL ENTRIES(SYS1.ICFCAT.VSYS303) -
       CATALOG(SYS1.ICFCAT.VSYS303)
/*
```

2) You can use LISTCAT to determine which VVDSs are connected to a BCS. You can use this information to determine which VVDSs to compare to a BCS when you use the DIAGNOSE command. For example, the following step lists the VVDSs connected to SYS1.ICFCAT.VSYS303:

LISTCAT LEVEL(SYS1.VVDS) CATALOG(SYS1.ICFCAT.VSYS303)

3) You can use LISTCAT to list the aliases associated with a catalog. Specify ALL with the catalog name in the ENTRIES parameter. The aliases are listed in the Associations group for the user catalog. If you specify a catalog in the CATALOG parameter, specify the master catalog. The following example lists the aliases associated with SYS1.ICFCAT.VSYS303:

LISTCAT ALL ENTRIES(SYS1.ICFCAT.VSYS303)

4) List all ALIAS entries in the master catalog:

LISTCAT ALIAS CAT(master.catalog.name)

This command provides a list of all aliases that are currently defined in your master catalog. If you need information only about one specific alias, use the keyword ENTRY(aliasname) and specify ALL to get detailed information.

5) List the catalog's self describing record:

LISTCAT ENT(user.catalog.name) CAT(user.catalog.name) ALL

This gives you detailed information about a user catalog, like attributes, statistics, extent information, and more. Because the self describing record is in the user catalog, you need to specify the name of the user catalog in the CAT statement. If you don't use the CAT keyword, only the user catalog connector information from the master catalog is listed.

6) List a user catalog connector in the master catalog:

LISTCAT ENT(user.catalog.name) ALL

You can use this command to display the volser and the alias associations of a user catalog as it is defined in the master catalog.

7) Listing a VSAM or non-VSAM data set:

LISTCAT ENT(data.set.name) ALL

Deleting data sets

If there is no damage to the BCS, VVDS, or VTOC, you can easily delete a data set. For example, by running an IDCAMS **DELETE** job, by using a JCL DD statement, or by issuing the command **DELETE** in ISPF 3.4 in front of the data set name. This will delete the entries for the data set in the BCS, VTOC, and VVDS. By doing that, the reserved space for this data set on the volume is released. The data set itself is not overwritten until the freed space is reused by another data set. You can use the parameter ERASE for an IDCAMS **DELETE** if you want the data set to be overwritten with binary zeros for security reasons.

Delete aliases

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.

Delete the catalog entry

To only delete a catalog entry you can use the **DELETE NOSCRATCH** command, the VVDS and VTOC entries are not deleted. The entry deleted can be reinstated with the **DEFINE RECATALOG** command.

```
//ARICHS JOB ...
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DELETE TEST1.A NOSCRATCH            /* deletes TEST1.A from the BCS */
DEFINE NONVSAM -                    /* redefines TEST1.A into the BCS */
(NAME(TEST1.A) DEVICETYPE(3390) VOLUMES(VSF6S3) RECATALOG )
/*
```

Delete VVR or NVR records

When the catalog entry is missing, and the data set remains on the DASD, you can use the **DELETE VVR** for VSAM data sets and **DELETE NVR** for non-VSAM SMS-managed data sets to remove its entry from the VVDS. You can only use these commands if there is no entry in the BCS for the data set.

```
//ARICHS JOB ...
//S1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1 DD VOL=SER=VSF6S3,UNIT=3390,DISP=OLD
//SYSIN DD *
        DELETE TEST1.A FILE(DD1) NVR
/*
```

When deleting VSAM KSDS, you must issue a **DELETE VVR** for each of the components, the DATA, and the INDEX.

Delete generation data groups

In this example, a generation data group (GDG) base catalog entry is deleted from the catalog.

```
//ARICHS JOB ...
//S1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE TEST1.GDG GENERATIONDATAGROUP RECOVERY
/*
```

The **DELETE** command with keyword **RECOVERY** removes the GDG base catalog entry from the catalog.

Delete an ICF

When deleting an ICF, you must take care to specify whether 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.

Delete with recovery is deleted in preparation for replacing it 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 the JCL.

```
//DELET13 JOB ...
//STEP1 EXEC PGM=IDCAMS
//DD1 DD VOL=SER=VSER01,UNIT=3390,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DELETE USER.CATALOG FILE(DD1) RECOVERY USERCATALOG
/*
```

RECOVERY specifies that only the catalog data set is deleted, without deleting the objects defined in the catalog.

DELETE AN EMPTY USER CATALOG

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.

```
//DELET6 JOB ...
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DELETE USER.CATALOG PURGE  USERCATALOG
/*
```

The **FORCE** parameter deletes all data sets in the catalog. The **DELETE** command deletes both the catalog and the catalog's user catalog connector entry in the master catalog.

Printing a Catalog or 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.

```
   //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)
  /*
```

Listing a Volume Table of Contents (VTOC):

The VTOCLIST line operator can also be used to generate a VTOC listing for a data set in the IEHLIST format. If you want to use batch processing to get a VTOC listing, you can use the IEHLIST utility. The listing can be formatted or dumped in hexadecimal.

Changing the Size or Contents of a Catalog:

The main issues concerning the size or content of a catalog are availability and recoverability, not performance. Catalog performance is normally unaffected by the size of the catalog, or by the aliases defined for it.

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. Use the following steps 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.

   ALTER SYS1.USERCAT.SOURCE LOCK

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 you are moving to the new catalog. Use the OUTFILE parameter to define a data set to contain the output
LISTCAT ALL ENTRIES(SYS1.USERCAT.SOURCE)

4. Use **EXAMINE** and **DIAGNOSE** to ensure that the catalogs are error-free.

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. 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. In the following example all entries that match the generic name VSAMDATA.* are moved from catalog USERCAT4 to USERCAT5.

```
//MERGE76 JOB ...
//STEP1 EXEC PGM=IDCAMS
//DD1 DD VOL=SER=VSER01,UNIT=DISK,DISP=OLD
// DD VOL=SER=VSER02,UNIT=DISK,DISP=OLD
// DD VOL=SER=VSER03,UNIT=DISK,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
REPROINDATASET(USERCAT4) OUTDATASET(USERCAT5)
ENTRIES(VSAMDATA.*) MERGECAT FILE(DD1)
/*
```

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 and uses a different master catalog.

7. Unlock both catalogs using **ALTER UNLOCK**.

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 a different catalog, so that you can delete the obsolete catalog. Use the following steps 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:

```
//JOB ...
//S1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1 DD DSN=listcat.output,DISP=(NEW,CATLG),
// SPACE=(TRK,(10,10)),
// DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSIN DD *
LISTC ENT(catalog.name) ALL OUTFILE(DD1)
/*
```

3. Use **EXAMINE** and **DIAGNOSE** to ensure that the catalogs are error-free.

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. Here is an example:

```
//MERGE6 JOB ...
//STEP1 EXEC PGM=IDCAMS
//DD1 DD VOL=SER=VSER01,UNIT=DISK,DISP=OLD
// DD VOL=SER=VSER02,UNIT=DISK,DISP=OLD
// DD VOL=SER=VSER03,UNIT=DISK,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
REPRO INDATASET(USERCAT4) OUTDATASET(USERCAT5) MERGECAT
FILE(DD1)
/*
```

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.
   The **DELETE ALIAS/DEFINE ALIAS** sequence must be run on each system that shares the changed catalogs and uses a different master catalog.

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. If your catalog is shared, run the **EXPORT DISCONNECT** command on each shared system to remove unwanted user catalogconnector 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.

Catalog attributes that can be altered:

- All password protection and ownership attributes
- Buffer sizes (BUFFERSPACE, BUFND, BUFNI)
- FREESPACE

- MANAGEMENTCLASS
- SHAREOPTIONS
- STORAGECLASS
- STRNO
- WRITECHECK.

Use Alter command to change the desired attribute. Close the catalog with modify catalog & close.

Catalog attributes that are unalterable

- Control interval size
- Data class
- Record size
- Replicate

To Change the above attribute or change the size of BCS follow the steps.

You can change the size of a catalog if the catalog is much larger than necessary, or if it has grown into excessive secondary extents. Before changing the size of a catalog, consider merging small catalogs or splitting large catalogs.

1. Lock the catalog.

```
//LOCKCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  ALTER ICFCAT.USER.VSYS303 LOCK
/*
```

2. Export the BCS with the EXPORT command. The aliases of the catalog are saved with the exported copy, and can be used in later steps to redefine the aliases.

```
//EXPORT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//EXPDD    DD DSN=CATBACK.ICFCAT.USER.VSYS303,DISP=OLD
//SYSIN    DD *
  EXPORT ICFCAT.USER.VSYS303  OUTFILE(EXPDD)  TEMPORARY
/*
```

TEMPORARY option is used to prevent the source from getting deleted.

3. Delete the BCS with the RECOVERY option.

```
//DELCAT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
 DELETE ICFCAT.USER.VSYS303 RECOVERY USERCATALOG
```

```
/*
```

4. Redefine the BCS with the desired space and performance attributes.

```
//DFNEWCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
   DEFINE USERCATALOG  (NAME(ICFCAT.USER.VSYS303) -
        VOLUME(SYS303)  MEGABYTES(15 5)  ICFCATALOG -
        FREESPACE(20 20)  STRNO(3) REPLICATE )  -
        DATA( CONTROLINTERVALSIZE(4096) -
        BUFND(4)  )  INDEX( BUFNI(4)  )
/*
```

5. Import the BCS using the IMPORT command. Specify INTOEMPTY on the IMPORT command.Also, specify ALIAS, so that the aliases exported with the catalog are redefined.
You can also specify UNLOCK to unlock the catalog (UNLOCK is the default).

```
//IMPORT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
   IMPORT INDATASET(CATBACK.ICFCAT.USER.VSYS303) -
        OUTDATASET(ICFCAT.USER.VSYS303)        -
        ALIAS  UNLOCK   INTOEMPTY
/*
```

6. If you did not unlock the catalog when you imported it, unlock it with the ALTER UNLOCK command.

Changing the Size of a VVDS:

If you need to change the size of a VVDS, you must backup all data sets that have an entry in the VVDS, remove these data sets from the volume, delete the VVDS, redefine the VVDS, and finally restore the data sets from their backups to the volume.

1. Obtain exclusive use of the volume by: stopping work, varying the volume offline to sharing systems, or quiescing sharing systems.

2. Remove all data sets from the volume that have an entry in the VVDS. All VSAM, SMS-managed data sets, and their catalogs should be removed using DFSMSdss (logical dump) or access method services (EXPORT, or REPRO followed by DELETE). With volumes that are not SMS-managed, you can use DFSMSdss to dump all VSAM data sets from the volume with one command. With SMS-managed volumes, use DFSMSdss to dump the entire volume.

3. Delete the VVDS specifying RECOVERY.

4. Define a new VVDS specifying the desired size. The default recommended size is trk(10,10).

5. Restore the data originally on the volume, using DFSMSdss or access method services (use the same utility used to dump the data sets).

Renaming a Catalog:

You cannot directly rename a catalog with the ALTER command. To rename a catalog, you must define a new catalog with the desired name and copy the old catalog into it. You can do this with the REPRO NOMERGECAT command.

1. Before you rename a catalog, make sure it does not have structural and logical errors; to do this, run IDCAMS DIAGNOSE and EXAMINE jobs.

2. When you use NOMERGECAT, recovery is required in the event of a failure. Make sure you have a current back up of the catalog.

3. Lock the old catalog using ALTER LOCK.

4. List the aliases of the old catalog using LISTCAT. Specify the name of the old catalog in the ENTRIES parameter, and specify the ALL parameter. This lists all the aliases of the catalog in the ASSOCIATIONS group of the LISTCAT listing. Specify a data set using the OUTFILE parameter to contain the LISTCAT listing. This data set can be used in step 8.

5. Define the new catalog with the desired name, size, and other attributes. You can use the old catalog as a model.

6. Copy the original catalog into the new catalog using the REPRO NOMERGECAT command. Specify the old catalog in the INDATASET parameter, and the new catalog in the OUTDATASET parameter. Do not specify any other parameters.

7. Delete the original catalog with the RECOVERY option. This also deletes the old aliases.

8. Use the listing of aliases created with LISTCAT to create a sequence of DEFINE ALIAS commands for each alias. Specify the name of the new catalog in the RELATE parameter.

Moving the Catalog:

Import the catalog to the new volume using the IMPORT command with the OBJECTS parameter. Note that IMPORT deletes the old catalog on the previous device and moves the catalog to the specified different volume.

```
//NEWVOL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
   IMPORT INDATASET(CATBACK.ICFCAT.USER.VSYS303) -
          OUTDATASET(ICFCAT.USER.VSYS303) -
          OBJECTS((ICFCAT.USER.VSYS303 -
                 VOLUMES(SYSNEW)))  ALIAS LOCK
/*
```

In this example, the new volume is SYSNEW, and is specified in the OBJECTS parameter.

Moving a Catalog to a Different System:

The following example shows how to connect a user catalog on SYSTEMA to SYSTEMB using IMPORT CONNECT, and then how to remove the catalog from SYSTEMA. Before removing the catalog from SYSTEMA, the aliases to the catalog are listed, so they can later be defined for the catalog on SYSTEMB. The LISTCAT output is directed to a data set which can later be edited to produce a series of DEFINE ALIAS commands to define the aliases on the receiving system. The alias associations are listed in the Associations group in the LISTCAT output.

```
//SYSMVCT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//LISTDATA DD DSN=LISTCAT.ALIAS.LISTING,DISP=OLD
//SYSIN    DD *
   LISTCAT ALL  ENTRIES(ICFCAT.USER.VSYS303) -
           OUTFILE(LISTDATA)

   IMPORT CONNECT OBJECTS((ICFCAT.USER.VSYS303 -
            DEVICETYPE(3390) VOLUMES(SYS303))) -
          CATALOG(SYSTEMB.MASTER.CATALOG)
```

```
     EXPORT ICFCAT.USER.VSYS303 -
          DISCONNECT  CATALOG(SYSTEMA.MASTER.CATALOG)
  /*
```

Establishing and breaking connections between BCS and VVDS:

The system connects VVDSs with BCSs as the need arises. However, if you want to explicitly establish a connection between a VVDS and a BCS, you can use DEFINE RECATALOG, specifying the VVDS as the entry name, and specifying the BCS in the CATALOG parameter.

To break the connection between a BCS and a VVDS, use DELETE NOSCRATCH, specifying the VVDS as the entry name, and the BCS in the CATALOG parameter. If the VVDS is available, it is checked to determine if the BCS in the CATALOG parameter has any data sets on the volume. If data sets are found on the volume, the DELETE command will fail. If the VVDS is not available, no check is performed.

Establishing and breaking connections between Master Catalog and User Catalog:

A BCS can be connected to the master catalog using the IMPORT CONNECT command. A catalog does not have to exist in order to connect it to a master catalog.

If you specify ALIAS on the IMPORT CONNECT command, any aliases which are already defined in the target master catalog for the catalog are maintained. Aliases which are not already defined in the target master catalog are not added, however. ALIAS can be specified even if no aliases are defined for the catalog.

Alternatively, you might need to disconnect a catalog from a system. In this case, use EXPORT DISCONNECT. This command removes the catalog's entry from the master catalog, and deletes all associated aliases. You do not have to lock a catalog before disconnecting it. Any jobs oriented to the catalog should end normally.

The catalog does not have to exist in order to disconnect it. For instance, if another system has deleted the catalog, any sharing system can disconnect that catalog.

Updating Catalog Connector Records

A connector record is a record in the master catalog for a user catalog. For shared catalogs, this record requires updating if another system has moved the catalog. If a connector record must be updated due to a change in device type or volume serial number for the catalog, use the IMPORT CONNECT ALIAS command. Specifying ALIAS preserves any aliases already defined for the catalog. Use the OBJECTS parameter to indicate the changed device type and volume serial number.

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.

To protect your catalogs and cataloged data, use the authorized program facility (APF) and the Resource Access Control Facility (RACF).

Authorized Program Facility Protection for Access Method Services:

The authorized program facility (APF) limits the use of sensitive system services and resources to authorized system and user programs. You must enter the names of those access method services commands requiring APF authorization to execute under TSO in the authorized command list (AUTHCMD) in the SYS1.PARMLIB member IKJTSOxx.

RACF authorization checking

RACF provides a software access control measure you can use in addition to or instead of 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.
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

RACF 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).

Examples of some profiles are:

STGADMIN.IDC.DIAGNOSE.CATALOG - protects the ability to use the access method services DIAGNOSE command against catalogs.

STGADMIN.IDC.DIAGNOSE.VVDS - protects the ability to use the access method services DIAGNOSE command against a VVDS when a comparison is made to a BCS.

STGADMIN.IDC.EXAMINE.DATASET - protects the ability to use the access method services EXAMINE command on integrated catalog facility catalogs.

STGADMIN.IGG.ALTBCS - protects the ability to alter BCS attributes.

Controlling Catalog Functions with RACF Profiles in the FACILITY Class:

RDEFINE FACILITY IGG.CATLOCK UACC(NONE) OWNER(CATADMIN)

PERMIT CLASS(FACILITY) IGG.CATLOCK ID(USER01) ACCESS(READ)

SETROPTS CLASSACT(FACILITY)

If you have READ access to the IGG.CATLOCK profile and ALTER authority to the catalog, you can lock or unlock a catalog. If you have READ access to the IGG.CATLOCK profile, you can access and repair a locked catalog.

To define entries in a catalog, users only need UPDATE authority to the data set profile protecting the catalog. Therefore, you should consider specifying UACC(UPDATE) for the data set profiles protecting user catalogs. To delete entries in a catalog, users need either ALTER authority to the data set or ALTER authority to the catalog. We recommend that you only give users ALTER authority to their own data sets.

Backing Up and Recovering Catalogs:

Because catalogs are essential system data sets, it is important that you maintain backup copies. Besides the backup and recovery of BCSs, you should also develop a strategy for backing up VVDSs, VTOCs, and VTOC indexes.

Backup of BCS:

The BCS can be backed up as a data set using the EXPORT command, DFSMSdss, or DFSMShsm. The aliases defined for the catalog are saved with the backup copy when EXPORT or DFSMShsm is used, or when logical dump is used with DFSMSdss. When you recover the catalog by importing it, you can have the saved aliases redefined and merged with existing aliases. DFSMSdss and DFSMShsm redefine the aliases automatically. However, when you use the IMPORT command, you must specify ALIAS to have aliases redefined or retained.

Backup of VVDS and VTOC:

The VVDS and VTOC should not be backed up as data sets, but are backed up as part of a full volume dump using DFSMSdss or DFSMShsm. The entries in the VVDS and the VTOC are backed up with the data sets they describe when the data sets are backed up with the IDCAMS EXPORT command, DFSMShsm, or DFSMSdss logical dump.

There are two ways that a VVDS or VTOC can be recovered:

1. Restore the volume containing the VVDS or VTOC, or
2. Rebuild the VVDS and VTOC by recovering the data sets on the volume.

Recovering a BCS:

Before recovering the BCS lock the BCS with the ALTER LOCK command. 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.

Additional considerations for Catalogs shared across systems:

For example, the user catalog SYS1.ICFCAT.SHARED, which has many aliases, resides on volume 339001 (a 3390), and is shared by SYSTEMA and SYSTEMB. If SYS1.ICFCAT.SHARED is successfully recovered by SYSTEMA to volume 339002 (another 3390), SYS1.ICFCAT.SHARED is inaccessible to SYSTEMB because its connector record in SYSTEMB's master catalog has an incorrect volume serial number. Executing the following step on SYSTEMB updates the volume serial number and preserves the aliases already defined for the catalog:

```
//CONNECT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
   IMPORT CONNECT ALIAS  OBJECTS((SYS1.ICFCAT.SHARED  -
              DEVICETYPE(3390)   VOLUMES(339002)))
/*
```

RLS Considerations When Recovering Shared Catalogs:

If you use VSAM record-level sharing (RLS), the AMS SHCDS CFREPAIR command should be used. When a catalog is recovered, but before making the catalog available, use the AMS SHCDS CFREPAIR command to reconstruct critical RLS information in the catalog. The catalog must be import connected on all systems to the master catalog before the SHCDS CFREPAIR command can be used.

Recovering a VVDS:

Before recovering a VVDS, decide if the VVDS is systematically damaged, or if only certain entries in the VVDS are damaged. If you cannot open the VVDS, for example, when you try to print it or access data sets which have entries in it, then the VVDS is probably systematically damaged, and should be recovered in its entirety.

If you can open the VVDS, run DIAGNOSE to determine which entries are damaged. You can then use the access method services DELETE command followed by data set recovery to recreate the VVDS entries for the affected data sets, and avoid a total VVDS recovery.

Before recovering a volume, it is necessary to get the volume offline, so that users cannot allocate resources on the volume as you try to restore it. Use the following procedure to get the volume offline if it is not managed by the Storage Management Subsystem:

1. Use the VARY command to get the volume offline.

2. Use the DISPLAY command to determine if the volume has been successfully varied offline, or if resources are still allocated on the volume.

3. Use MODIFY CATALOG to unallocate the VVDS or any catalogs on the volume which are allocated. Use the VUNALLOCATE parameter to unallocate the VVDS. Use the UNALLOCATE command to unallocate the catalog.

If the volume is SMS-managed, set the SMS VOLUME STATUS to DISALL before using the VARY command. Then, check for allocations with the DISPLAY command, and use MODIFY CATALOG if necessary. After you have gotten control of the volume, use DFSMShsm or DFSMSdss to recover it.

Recataloging Data Sets and VSAM Objects:

After a BCS recovery, some entries in the recovered BCS might not accurately reflect the current characteristics of a data set or VSAM object. The VVDS, VTOC, and tape labels should contain the accurate information for existing data sets. Data sets can only be recataloged into the catalog specified in the VVR/NVR unless they are pagespace, swapspace or SYS1 data sets.

For SMS-managed data sets and all VSAM data sets and associations, BCS entries can be recataloged using the access method services DEFINE command with the RECATALOG option.
If there is a BCS entry for the data set, first remove it using DELETE NOSCRATCH.
For non-VSAM, non-SMS-managed data sets, you can delete the BCS entry using the DELETE NOSCRATCH command. Then, you can catalog the data set with DEFINE NONVSAM.

Recataloging a VVDS:

If you want to recreate the BCS entry for a VVDS, use the access method services DEFINE CLUSTER command with the RECATALOG option. Specify the name, volume of the VVDS, and NONINDEXED. The BCS entry is rebuilt using information in the VVDS and the command. A VTOC entry for the VVDS must also exist.

Analyzing Catalogs for Errors and Synchronization:

Proper catalog functioning requires that the structure of the BCS remain sound. It also requires that data set information contained in the BCS, VVDS, and VTOC be synchronized. That is, the characteristics of a data set (extent information, SMS class names, etc.) must be the same in the data set's entries in the BCS, VVDS, and VTOC.

Analyzing a BCS for Structural Errors:

If a catalog is causing jobs to fail and you cannot trace the failure to unsynchronized entries in the catalog, the catalog might be structurally unsound. Structural problems are those which affect a BCS's characteristics as a VSAM key-sequenced data set, not as a catalog.

When using EXAMINE against a BCS, first use the access method services ALTER command to lock the catalog. This prevents updates to the catalog while you are inspecting its structure.
To test the structure of a BCS, use the access method services EXAMINE command.
With this command, you can test both the index and data components of a BCS.

Analyzing a Catalog for Synchronization Errors:

Catalog entries might become unsynchronized, so that information about the attributes and characteristics of a data set are different in the BCS, VVDS, and VTOC. These differences may make a data set inaccessible or otherwise unusable. To analyze a catalog for synchronization errors, you can use the access method services DIAGNOSE command. With this command, you can analyze the content of catalog records in the BCS and VVDS, and compare VVDS information with DSCB information in the VTOC.

To check the dependent content of catalog records, use the COMPAREDD or COMPAREDS parameters. If you are analyzing a BCS, specify related VVDSs. If you are analyzing a VVDS, specify related BCSs. You can limit dependency checking to selected BCSs or VVDSs, rather than specifying all related BCSs or VVDSs. For example, the VVDS is checked for an entry that is consistent with the BCS entry. This is not a complete check of the external data set or its entry. It is only a consistency check between the two.

If you are comparing:

- A BCS, the VVDS record is checked for dependency.
- A VVDS, the BCS record, and the VTOC DSCB are checked for dependency.

EX:

To diagnose a BCS use the following IDCAMS command:

```
DIAGNOSE ICFCATALOG   INFILE(CATDD) -
         COMPAREDD(VVDSDD)   ERRORLIMIT(1)
```

CATDD points to the name of the catalog and VVDSDD points to a VVDS.

To diagnose a VVDS use the following IDCAMS command:

```
DIAGNOSE VVDS  INFILE(VVDSDD)  COMPAREDD(CATDD)   ERRORLIMIT(1)
```

Catalog performance

Factors that have influence on the performance:

- Main factor: amount of I/Os
  Cache catalogs to decrease number of I/Os
  Use Enhanced Catalog Sharing
- No user data sets in master catalog
- Options for DEFINE USERCATALOG
- Convert SYSIGGV2 resource to avoid enqueue contention and deadlocks between systems.
- Eliminate the use of JOBCAT/STEPCAT

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 types 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.
The two types of cache are used to keep catalog records in the storage. This avoids I/Os that would be necessary to read the records from DASD.
There are several things you need to take into considerations to decide what kind of cache to use for which catalog.

Convert SYSIGGV2 resource

Catalog management uses the SYSIGGV2 reserve while serializing access to catalogs. The SYSIGGV2 reserve is used to serialize the entire catalog BCS component across all I/O as well as to serialize access to specific catalog entries.
If the catalog is shared only within one GRSplex, you should convert the SYSIGGV2 resource to a global enqueue to avoid reserves on the volume on which the catalog resides. If you are not converting SYSIGGV2, you can have ENQ contentions on those volumes and you can even run into deadlock situations.

Eliminate use of JOBCAT/STEPCAT

The catalog address space

Catalog functions are performed in the *catalog address space* (CAS). The jobname of the catalog address space is CATALOG.

As soon as a user requests a catalog function the CAS gets control to handle the request. When it has finished, it returns the requested data to the user. A catalog task which handles a single user request is called a *service task*. To each user request a service task is assigned. The minimum number of available service tasks is specified in the SYSCATxx member of SYS1.NUCLEUS (or the LOADxx member of SYS1.PARMLIB). A table called the CRT keeps track of these service tasks. The CAS contains all information necessary to handle a catalog request, like control block information about all open catalogs, alias tables, and cached BCS records. During the initialization of an MVS system, all user catalog names identified in the master catalog, their aliases, and their associated volume serial numbers are placed in tables in CAS.You can use the **MODIFY CATALOG** operator command to work with the catalog address space.

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.
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. There is no way to suppress these indicative dumps.

Working with the catalog address space

Use the MODIFY CATALOG command to:

List information like Settings

Catalogs currently open in the CAS
Performance statistics
Cache statistics
Service tasks
Module information

Interact with the CAS by

Changing settings
Ending or abending service tasks
Restarting the CAS
Closing or unallocating catalogs

Working with the catalog address space

You can use the command **MODIFY CATALOG** to extract information from the CAS and to interact with the CAS. This command can be used in many variations.

Examples of the **MODIFY CATALOG** command:

MODIFY CATALOG,REPORT

The catalog report lists various information about the CAS, like the service level, the catalog address space ID, service tasks limits, and more.

MODIFY CATALOG,OPEN

This command lists all catalogs that are currently open in the CAS. It shows whether the catalog is locked or shared, and the type of cache used for the catalog. A catalog is opened after an IPL or catalog restart when it is referenced for the first time. It remains open until it is manually closed by the **MODIFY CATALOG** command or it is closed because the maximum number of open catalogs has been reached.

MODIFY CATALOG,LIST

The **LIST** command shows you all service task that are currently active handling a user request. Normally the active phase of a service task is only of a short duration, so no tasks are listed. This command helps you to identify tasks which could be the reason for catalog problems if the performance slows down.

MODIFY CATALOG,DUMPON(rc,rsn,mm,cnt)

Use this command to get a dump of the catalog address space when a specific catalog error occurs. This catalog error is identified by the catalog return code (rc), the reason code (rsn) and the name of the module which sets the error (mm). You can also specify, in the cnt-parameter, for how many of the rc/rsn-combinations a dump is to be taken. The default is one. The module identifier corresponds to the last two characters in the catalog module name. For example, the module identifier is A3 for IGG0CLA3. You can substitute the module name by two asterisks (**) if you don't know the module name or if you don't care about it.

MODIFY CATALOG,ENTRY(modulename)

This command lists the storage address, fmid, and the level of a catalog module (csect). If you do not specify a module name, all catalog csects are listed.

MODIFY CATALOG,REPORT,PERFORMANCE

The output of this command shows you significant catalog performance numbers about number and duration of ENQs and DEQs for the BCS and VVDS and more. Use the command to identify performance problems.

MODIFY CATALOG,REPORT,CACHE

The cache report lists cache type and statistics for the single catalogs which are open in the CAS.

MODIFY CATALOG,RESTART

Use this command to restart the catalog address space. This should be done *only* in error situations when the other option would be an IPL.

Fixing temporary catalog problems

Fixing temporary catalog problems involves

- Rebuild information of BCS or VVDS control blocks

Close or unallocate a BCS
Close or unallocate a VVDS

- Determine tasks that causes performance problems

Display GRS information about catalog resources and devices
Display information about catalog service tasks
End service tasks

Rebuild information about a BCS or VVDS

Occasionally, the control blocks for a catalog kept in the catalog address space might be damaged. You might think the catalog is damaged and in need of recovery, when only the control blocks need to be rebuilt. If the catalog appears damaged, try rebuilding the control blocks first. If the problem persists, recover the catalog. You can use the following commands to close or unallocate a BCS or VVDS in the catalog address space. The next access to the BCS or VVDS reopens it and rebuilds the control blocks.

MODIFY CATALOG,CLOSE(catalogname) - Closes the specified catalog but leaves it allocated.

MODIFY CATALOG,UNALLOCATE(catalogname) - Unallocates a catalog; if you do not specify a catalog name, *all* catalogs are unallocated.

MODIFY CATALOG,VCLOSE(volser) - Closes the VVDS for the specified volser.

MODIFY CATALOG,VUNALLOCATE - Unallocates *all* VVDSs; you cannot specify a volser, so try to use VCLOSE first.

Recover from performance slow downs

Sometimes the performance of the catalog address space slows down. Catalog requests may take a long time or even hang. There can be various reasons for such situations, for example a volume on which a catalog resides is reserved by another system, thus all requests from your system to this catalog wait until the volume is released. Generally, the catalog component uses two resources for serialization:

SYSIGGV2 to serialize on the BCS
SYSZVVDS to serialize on the VVDS

Delays or hangs can occur if the catalog needs one of these resources and it is held already by someone else, for example by a CAS of another system. You can use the following commands to display global resource serialization (GRS) data:

D GRS,C

Display GRS contention data for all resources, who is holding a resource, and who is waiting.

D GRS,RES=(resourcename)

Displays information for a specific resource.

D GRS,DEV=devicenumber

Displays information about a specific device, such as if it is reserved by the system. You should route these commands to all systems in the sysplex to get an overview about hang situations. When you have identified a catalog address space holding a resource for a long time, or the GRS outputs do not show you anything but you have still catalog problems, you can use the following command to get detailed information about the catalog services task:

MODIFY CATALOG,LIST

Lists the currently active service tasks, their task IDs, duration, and the job name for which the task is handling the request. You should watch for tasks with long duration time. You can get detailed information about a specific task by running the following command for a specific task ID:

MODIFY CATALOG,LISTJ(taskid),DETAIL

Shows you detailed information about a service task, for example if it's waiting for the completion of an ENQ. When you have identified a long running task which could be in a deadlock situation with another task (on another system), you can end and redrive the task to resolve the lockout.

Commands to end a catalog service task

MODIFY CATALOG,END(taskid),REDRIVE

End a service task and redrive it.

MODIFY CATALOG,END(taskid),NOREDRIVE

Permanently end the task without redriving.

MODIFY CATALOG,ABEND(taskid)

Abnormally end a task which could not be stopped by using the **END** parameter. You can use the **FORCE** parameter for these commands if the address space that the service task is operating on behalf of has ended abnormally. Use this parameter *only* in this case. You could also try to end the job for which the catalog task is processing a request.

Sharing catalogs across systems

If you need to share data sets across systems, you need to share the catalogs which contain these data sets. A BCS catalog is considered shared when *both* of the following are true:
 It is defined with share options (3 4). It resides on a shared device. You can check if a catalog is shared by running the operator command:

MODIFY CATALOG,ALLOCATED If the ***R***-flag is on for the catalog, it is shared.