

## Agenda

---

### Day 1

- Mainframe Hardware Overview
- z/OS High Level Overview
- TSO & ISPF for z/OS
- UNIX System Services
- Mainframe Access

### Day 2

- Hardware Management & Definition
- z/OS Operations
- Data Sets
- Using ISPF

### Day 3

- Virtual Storage and Address Spaces
- JES2 Introduction
- Introduction to JCL

### Day 4

- More on JCL
- The IPL Process

---

# The Mainframe

## Unit objectives

---

After completing this unit, you should be able to understand:

- Current zSeries hardware range
- History of System 360 -> z/OS
- Concepts of IO
- Concepts of LPAR
- Basic understanding of zVM
- How zVM hosts Linux

# IBM System z Hardware Overview – 1



## CPC

Central Processor Complex

### **Current Range (2008/9)**

IBM System z10 EC (Enterprise Class) (*Machine Type 2097*)

- 1 to 64 way processor units (PUs) – ‘engines’
- Multiple types of Speciality Engines (zAAPs, IFLs, etc)
- Up to 1.5 TeraBytes Memory
- Up to 336 x 4Gbps I/O channels (FICON Express4)
- Up to 48 OSA ports (Network Interface Cards)
- Crypto Assist Functions on each PU
- Up to 16 Crypto Express 2 (PCI-X) Adapters
- plus many other options

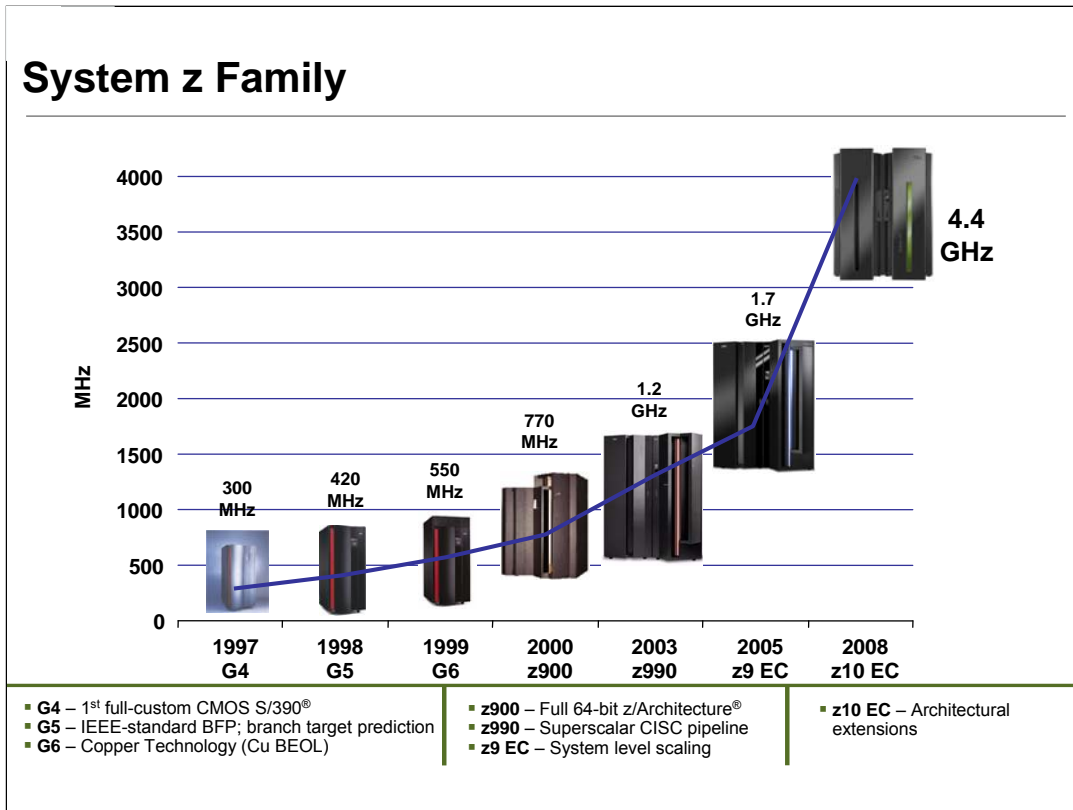


IBM System z10 BC (Business Class (*Machine Type 2098*))

- Single model – E10
- Multiple types of Speciality Engines (zAAPs, IFLs, etc)
- Up to 128 GBytes Memory
- Lower numbers for similar features on the z10 EC, as above

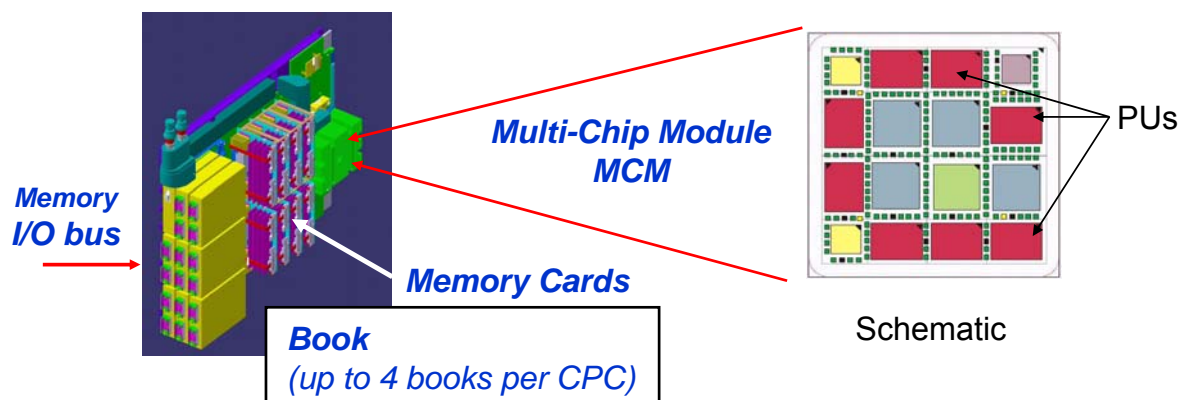
The System z10™ EC provides a strong combination of past mainframe characteristics, plus new functions designed around scalability—including flexible granularity solutions, virtualization, availability, and security. And the z10 EC continues to extend and support the use of dedicated processors for specialized workloads, including the System z10 Application Assist Processor (zAAP), Integrated Facility for Linux® (IFL), Internal Coupling Facility (ICF) and System z10 Integrated Information Processor (zIIP).

The z10™ BC is designed specifically as a midrange mainframe and delivers extensive growth options and excellent price/performance for those customers requiring a lower-capacity entry point and more granular growth options than offered with the System z10 Enterprise Class



The design of the IBM System z10 processor chip is the most extensive redesign in over 10 years, resulting in an increase in frequency from 1.7 GHz (z9 EC) to 4.4 GHz on the z10 EC. It is designed for secure data serving, yet also was enhanced to provide improvement enhances for CPU intensive workloads. The result is a platform that continues to improve upon all the mainframe strengths customers expect, yet opens a wider aperture of new applications that can all take advantage of System z10s extreme virtualization capabilities, and lowest TCO Vs distributed platforms.

## IBM System z Hardware Overview - 2



A PU is the generic term for the z/Architecture processor on the Multichip Module (MCM) that can be characterized as:

- A Central Processor (CP) – **CPs are required to run z/OS**
- An Internal Coupling Facility (ICF) to be used by the Control Facility Control code (CFCC).
- An Integrated Facility for Linux (IFL) – This can only be used by Linux and z/VM
- An additional System Assist Processor (SAP®) to be used by the Channel Subsystem.
- A System z Application Assist Processor (zAAP) – for Java workloads.
- A System z Integrated Information Processor (zIIP) – for complex DB2 and other workloads.

The fundamental building block of the System z9™ is the 'book'. A book contains processing power, memory and I/O connectivity.

The System z9™ EC is available in 5 basic hardware models; the S08 (1 book), S18 (2 books), S28 (3 books), S38 (4 books) and the top end S54 model (4 books). The top end model S54 has 16 PUs per book, whereas all the others have 12 PUs per book.

The PU configuration is made up of two System Assist Processors (SAPs) per book and two spare PUs per server. The remaining PUs can be characterized as Central Processors (CPs); Integrated Facility for Linux (IFL) processors; System z Application Assist Processors (zAAPs); System z9 Integrated Information Processors (zIIPs); Internal Coupling Facility (ICFs) processors; or additional SAPs.

SAPs provide the processing resource for the Channel Subsystem (CSS) which handles all I/O functions. Hence, you might like to think of them as 'Shopping Assist Processors'.

Each book supports up to 128 GB of memory, for a server maximum of 512 GB, and 16 high-performance Self-Timed Interconnects for data communications between memory and I/Os.

Note that the software license cost of z/OS, and associated products, increases with the power of the CPC being used. A value, called the MSU (machine service units) is generally used to compute these software licenses. Only PUs configured as CPs count in this calculation. The other engine types have to be purchased but they do not affect the z/OS software license charges.

### Example

Someone requiring a 3 engine box (e.g. 3 CPs) would order a hardware model S08 together with a software order which relates to just 3 CPs. Such a configuration would comprise of 3 CPs, plus 2 SAPs, plus 7 spare PUs. Additional features can be ordered which allow a customer to use the additional 'spare' PUs 'on demand', and only pay for the additional capacity as and when required.

# Hardware Models & Operating Systems

## System z

EC / BC

Enterprise Class  
Business Class

## zSeries

Z990 / z890

Z900 / z800



## System/390 (s/390)

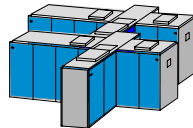
*Smaller Footprint Machines  
CMOS (Cool) technology*

967x

309x

308x

*Large Water-cooled Machines  
Bipolar (Hot) technology*



## System 370 (s/370)

## System 360 (s/360)



z/OS

OS/390

MVS/ESA

MVS/XA

MVS

MVT

OS

Other  
Operating  
Systems  
Supported

-  
z/VM

Linux

z/VSE

z/TPF

Visit the archives at [http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe\\_intro.html](http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro.html)

The IBM mainframe, as we know it, started in the 1960s and was called the System 360 range. These were built from individual transistors, resistors, diodes, etc. With the development of computer chips these were used in the S/370 range. The S/370 top end models were physically very large. Much more complex and dense electronics were used in the subsequent family, the S/390. The technology used at that time was called Bipolar. It consumed large amounts of electricity and produced a lot of waste heat, requiring the use of water cooling in such systems. Such machines had high environmental costs (power, space, cooling).

A later development in the S/390 family was the introduction of CMOS technology, which brought dramatic reductions in environmental costs. The machines were physically much smaller, required far less electrical power and cooling.

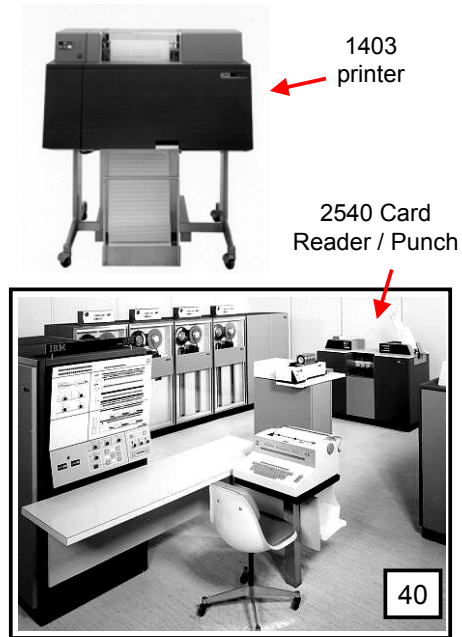
The z900 and z800 were the first models in the zSeries range (Equivalent to the System z9™ EC & BC models).

The z990 and z890 introduced the modular 'book' structure which was carried forward with the System z9™ and z10™ ranges.

z/OS evolved from MVS (and previous OSs), as shown in the diagram.

Note that operating systems, other than z/OS, can also run on System z.

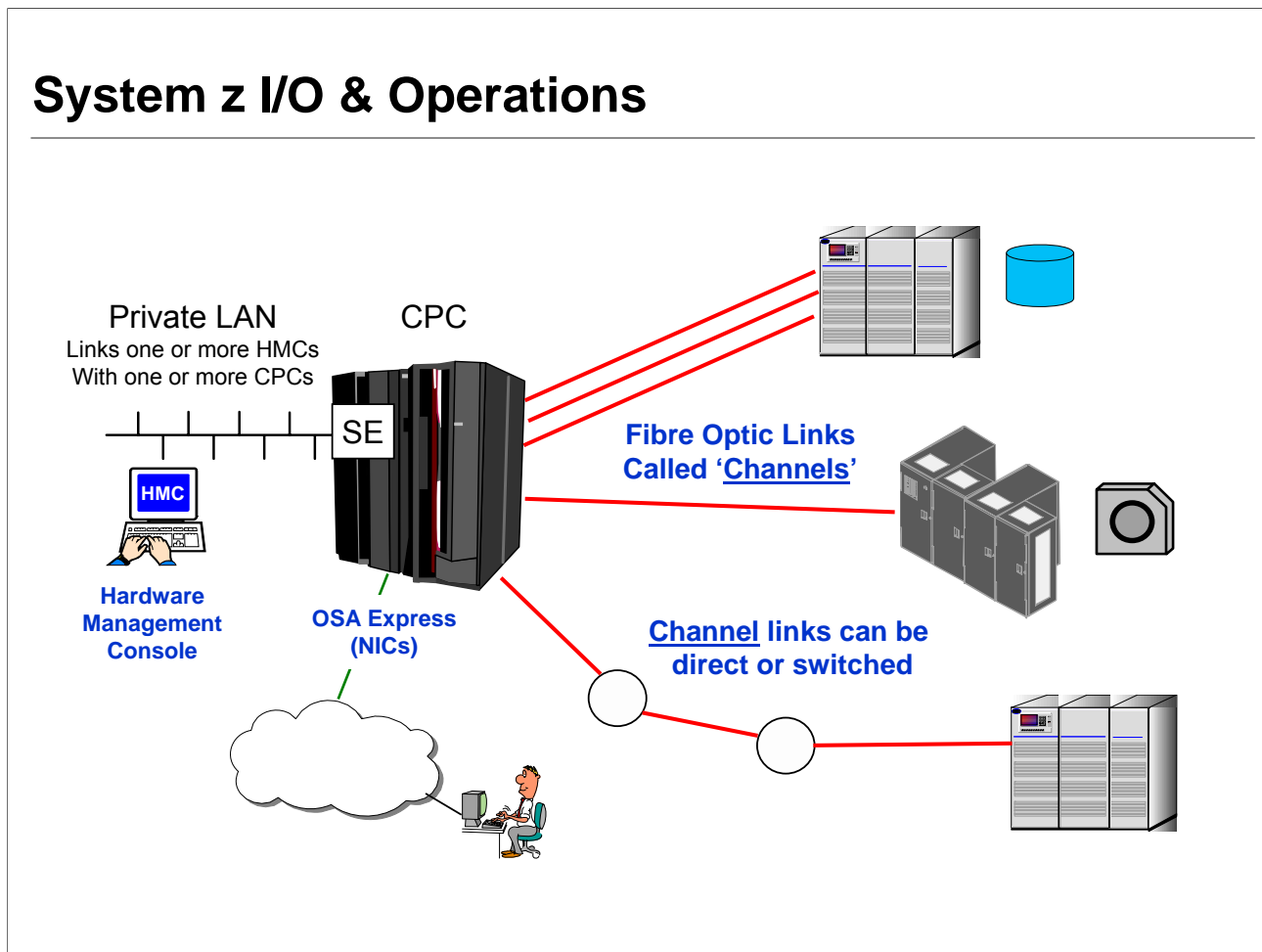
## System/360 Models 40 & 50



The foil above shows S/360 model 50 and S/360 model 40 configurations, as well as one of the original 'fan fold' printers, the IBM 1403 printer.



# System z I/O & Operations



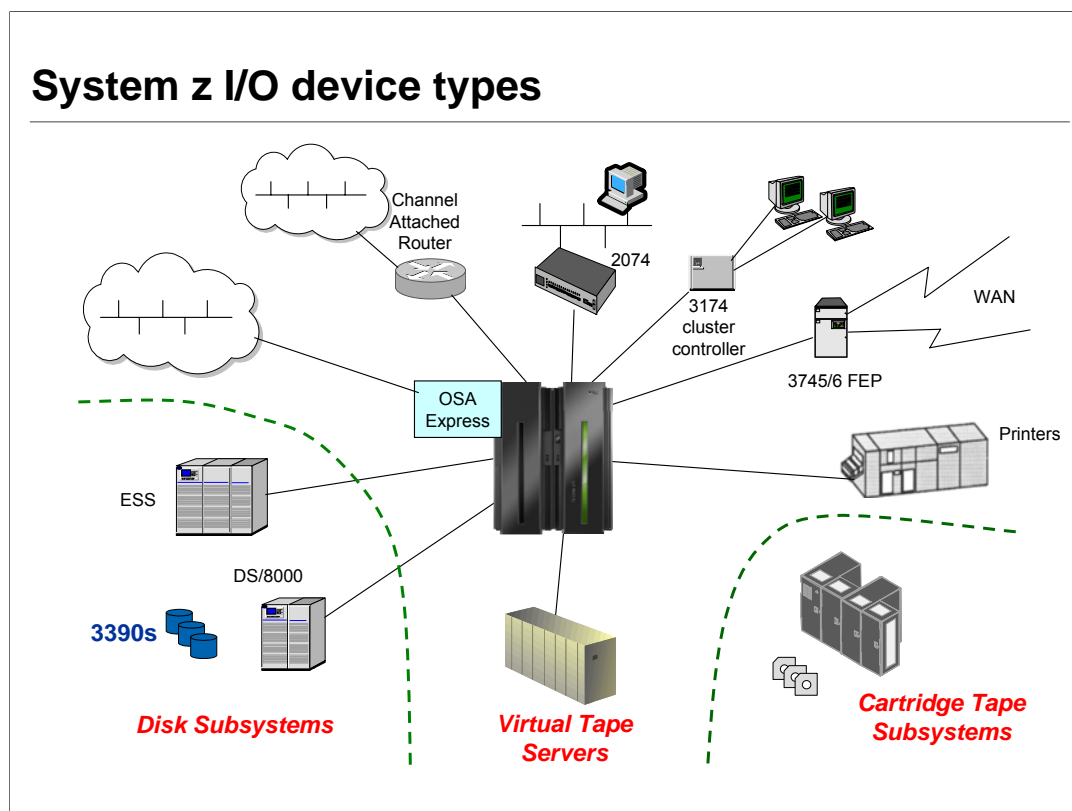
All I/O is connected to the CPC using 'channels'. A single channel is comprised of an adapter (PCHID/CHPID) on the CPC, the fibre optic cable, and the connector (channel adapter) on the external I/O equipment. The fibre optic link can connect *directly* to the I/O, or go via fibre optic switches, called 'directors'.

Architecturally the System z9™ can have up to 1024 channel links but these cannot all be high speed links. Multiple channel types are available. One of the latest types is called FICON Express4 which supports 4Gbps (400MByte/sec). The System z9™ supports up to 336 x 4Gbps FICON Express4 channels. (The first fibre optic channel type was called ESCON with a bandwidth of 17MBytes/sec and many customers still have these installed. However most will have switched, or are switching to some form of FICON channel – 1, 2 or 4 Gbps).

Network connectivity is accomplished using a special type of channel (PCHID) called an OSA Express (generic name). These are offered in various bandwidths (10/100/1000/10000 Mbps) and LAN types (TokenRing, ATM, Ethernet, Fast Ethernet, xGbit Ethernet). The current high bandwidth version offered is 10 Gbit Ethernet.

The System z9™ can support a maximum of 24 x 10 Gbps OSA Express2 ports.

The CPC is powered on, activated, and IPL'd (Initial Program Load) using an HMC. The HMC connects to one or more CPCs over a private LAN. The HMC is a standard desktop PC, supplied by IBM, with a specific application suite. The HMC communicates with a Thinkpad, called the Service Element (SE), located within the CPC. The SE is the front end to 'hardware' operation and management of the CPC. You can have multiple HMCs connected to multiple CPCs. So from any HMC you can manage any of the CPCs. You may have an HMC in the machine room, one in the operations bridge, another in the system programming area, and so on. Most customers locate their HMCs in physically secure areas.



The above diagram shows a variety of the I/O device types attached to System z systems. Some device types shown, for example the 3174 cluster controller with dumb screens, are very old and unlikely to be found in most installations.

As with most large servers, the predominant device type will be a disk subsystem such as the IBM DS/8000. However, it should be noted that from the z/OS perspective these will be seen as 3390 device types. The 3390 was a real family of disk devices with a specific geometry. The 3390, and previous IBM mainframe disk devices, do not have a 'Fixed Block' architecture, as used by UNIX and Windows systems. The 3390 uses a 'Count, Key, Data (CKD) format.

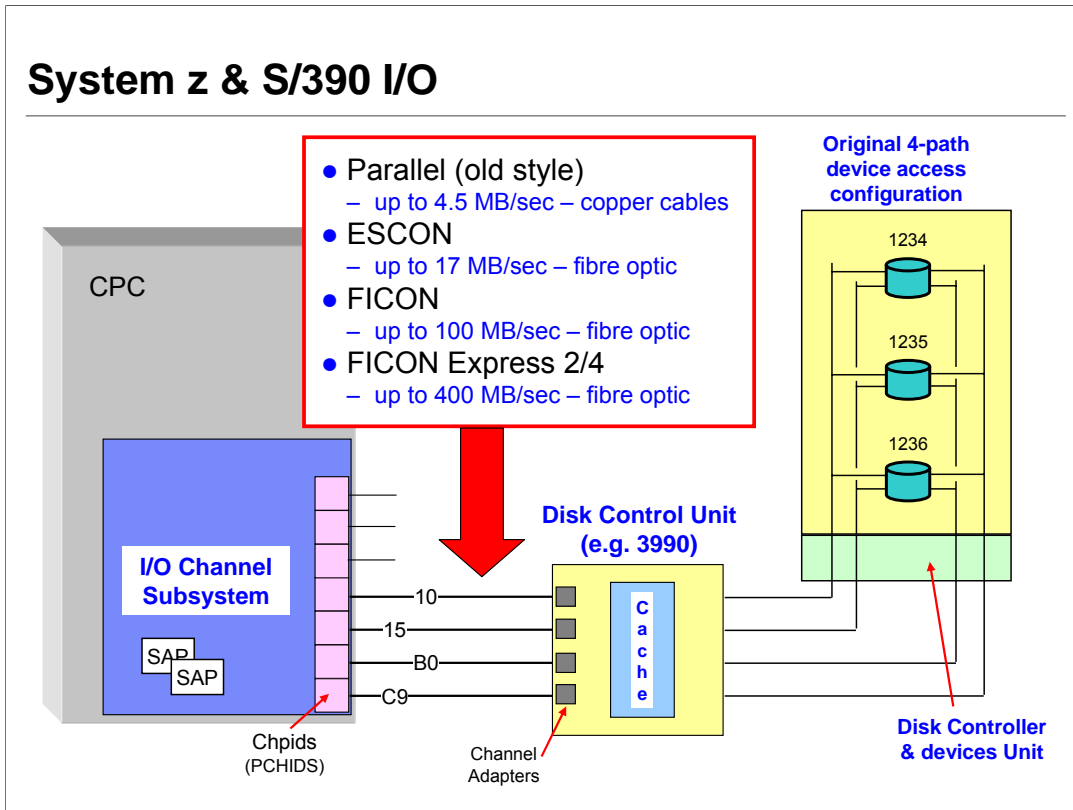
There were four 3390 models which are 'virtualised' by current day disk subsystems such as the IBM DS/8000. These models are:

- 3390-1 .. 1/3 the capacity of the model 3
- 3390-2 .. 2/3 the capacity of the model 3
- 3390-3 .. The most popular with a capacity of around 2.7GigaBytes
- 3390-9 .. Three times the capacity of the model 3

To provide increased capacity additional 'virtualised' 3390 models can be configured, although these never actually existed. These are:

- 3390-27 .. Three times the capacity of the model 9
- 3390-54 .. Six times the capacity of the model 9

Apart from the OSA Express family (Network Interface Devices), all devices are connected via fibre optic links called 'channels'. The original channel types consisted of large copper cables but you are very unlikely to come across these in current day installations.



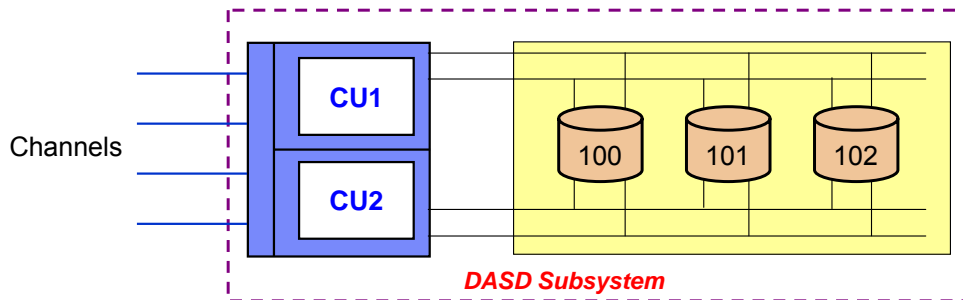
At the CPC end of a channel, the physical adapter was originally called a CHPID (Channel Path ID), however this has now been 'virtualised' to become a 'logical CHPID'. Since the time of the IBM z990, the real adapter is called a PCHID (Physical Channel Identifier).

Within any I/O subsystem, there is a Control Unit (CU) function, which then connects to the devices, either real or virtualised.

In the Hardware Configuration Definition (HCD), all paths from the Logical CHPID to the PCHID, from the PCHID to the Control Unit and then to the Device are defined.

## Early Disk Subsystems

3990, 3390 and RAMAC RVA style Subsystems



Multiple channels to Subsystem connections may be cross-coupled within the box for availability

Each volume supported by an individual actuator (3390) or mapped to a set of RAID disks

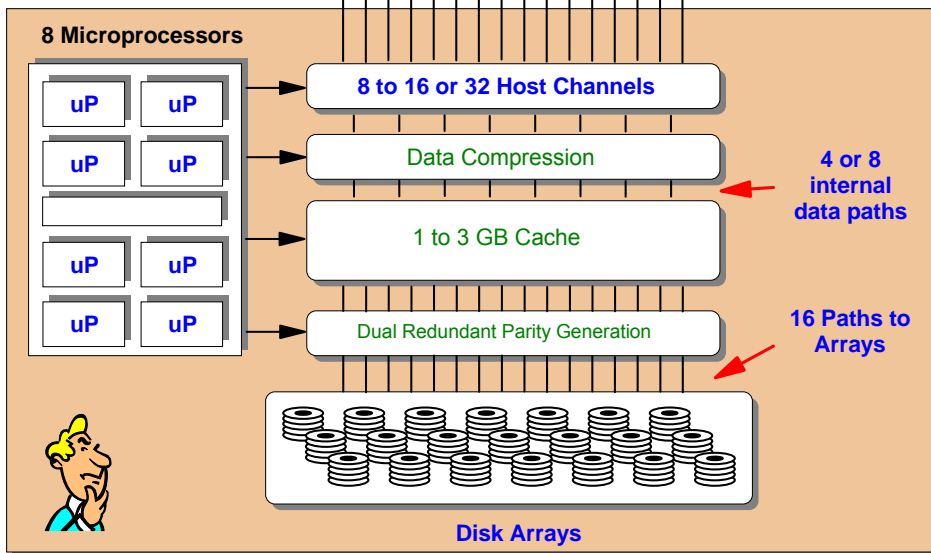
3390 devices were front ended by a 3990 control unit. In fact, the 3990 had two physical Control Unit functions within it, which were cross connected to all channels. Typically 3990s were paired to avoid a single point of failure. (This is not shown in the above diagram).

For example, assume we have two banks of 3390 disks (3390X and 3390Y) and two 3990s (3990A and 3990B).

- 3990A-CU1 and 3990B-CU1 would connect to 3390X
- 3990A-CU2 and 3990B-CU2 would connect to 3390Y.

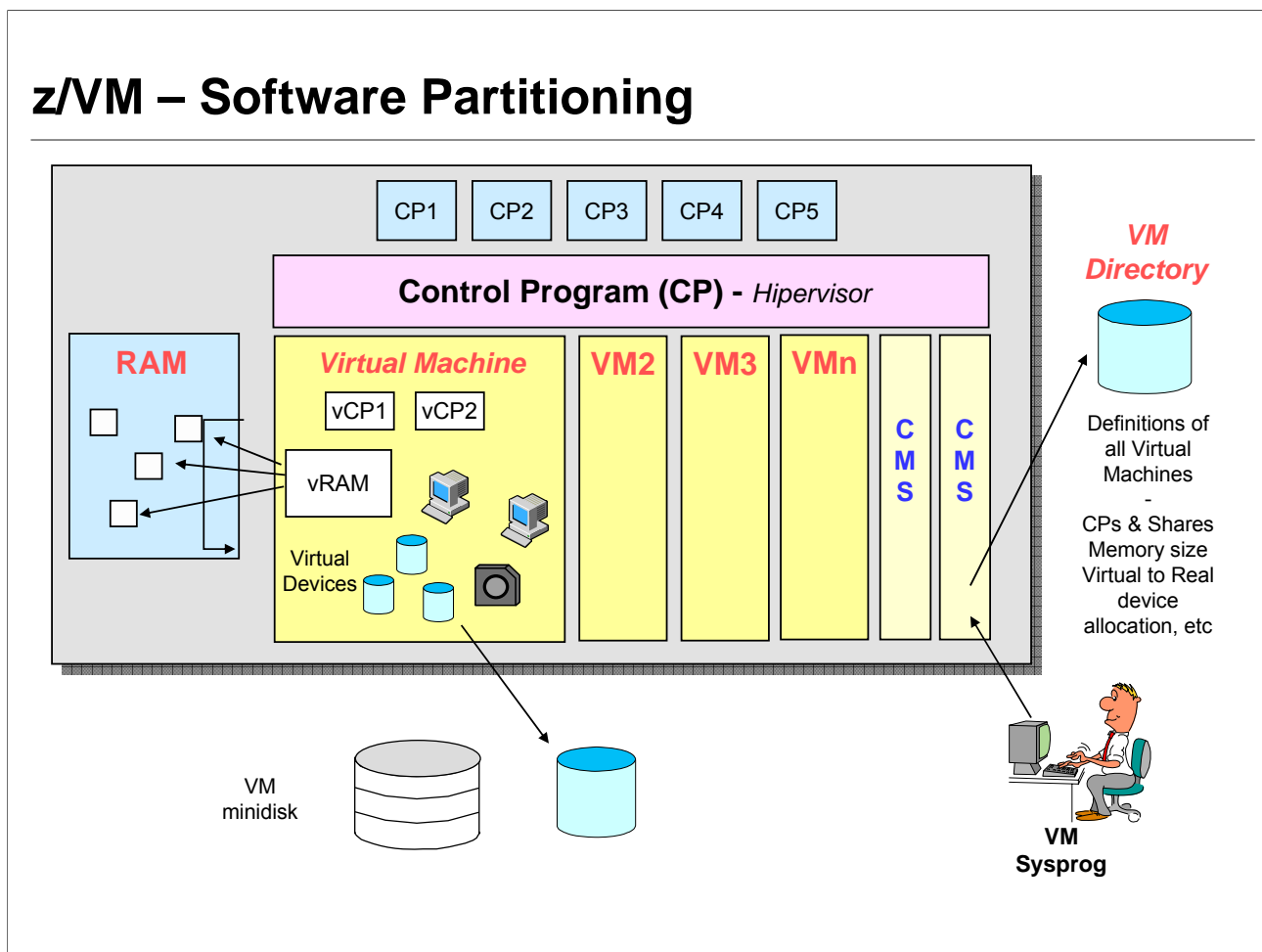
## Original Mainframe 'Virtualised' Disk Subsystems

RAMAK Virtual Array (RVA)



The original IBM 'virtualised' disk subsystem was the RAMAK Virtual Array (RVA). This used multiple FBA (Fixed Block Architecture) disks, configured in RAID5 Arrays. The RVA appeared to the S/390 as a collection of 3990 control units with 3390 disks.

## z/VM – Software Partitioning



Software partitioning was developed many years before LPAR. In many respects LPAR is a cut down and much simplified version of z/VM.

The main part of z/VM is the Control Program (CP) which provides the 'hypervisor' functionality. The other key part of z/VM is a specialised operating system called the Conversational Management System (CMS). CMS is used to configure and manage the z/VM system and can also be used to run applications. These can be customer written applications or IBM supplied (for example TCP/IP).

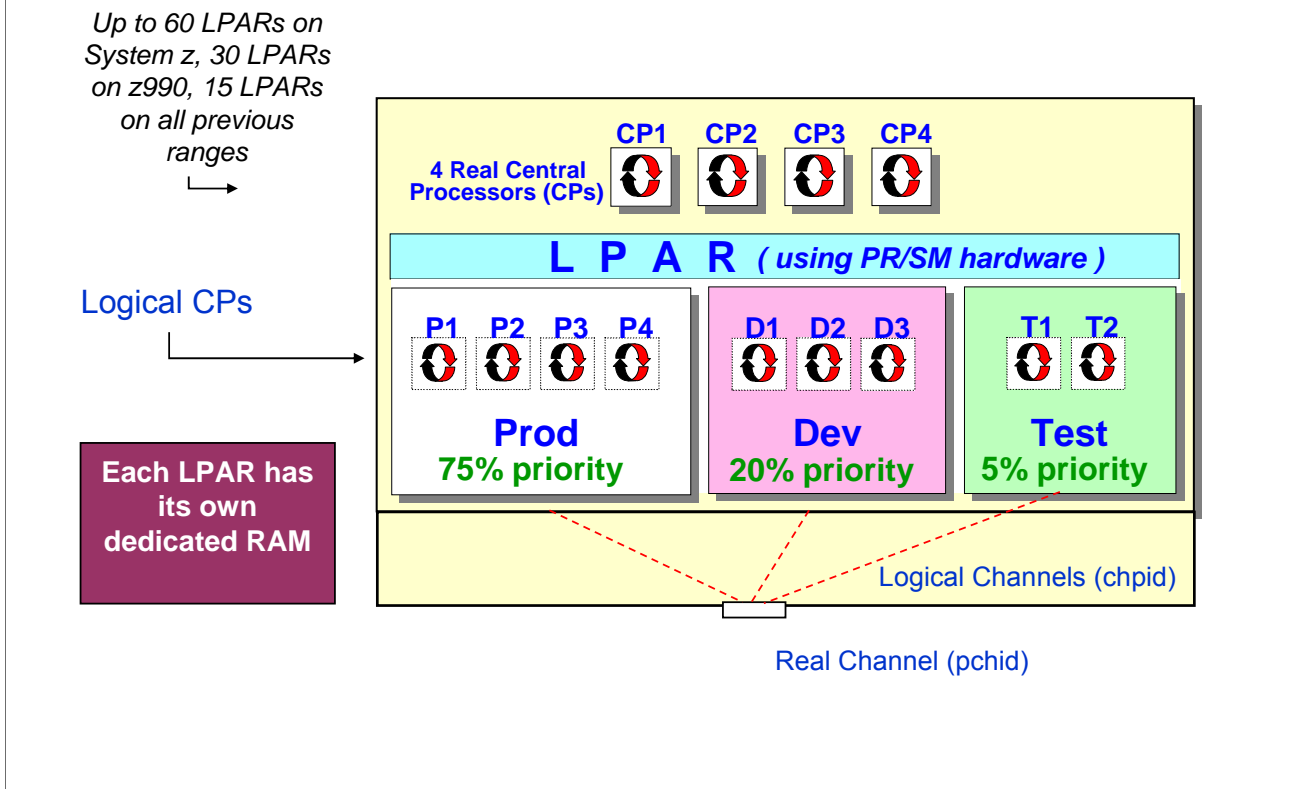
Each VM Userid is configured as a complete Virtual Machine, comprising CPs, Memory and I/O devices. The information about each Virtual Machine (VM) is maintained in the VM Directory. The directory entry for each VM defines the number and relative shares of Virtual CPs, the virtual memory allocation, virtual devices and their mapping to real devices. A virtual disk may be mapped to a complete real disk, or may be mapped to a portion of a real disk. This latter type is called a VM mini-disk.

RAM is not dedicated to VMs but is shared by all z/VM manages.

z/VM provides a more flexible environment to LPAR. However z/VM is a very different operating system to z/OS so requires a different skill set. This is the main reason why most z/OS customers prefer to use LPAR, as this requires no additional operating system skills.

Unlike LPAR, z/VM is not limited to 60 LPARs but can support many hundreds of VMs. This is one of the reasons for its popularity with organisations wishing to run Linux on System z.

# Logical Partitioning - LPAR



LPAR stands for Logical Partitioning. LPAR is microcode which uses a special hardware feature called PR/SM. LPAR and PR/SM has been standard on all IBM CPCs since around 1986.

In the foil we show a CPC which has four real CPs (engines).

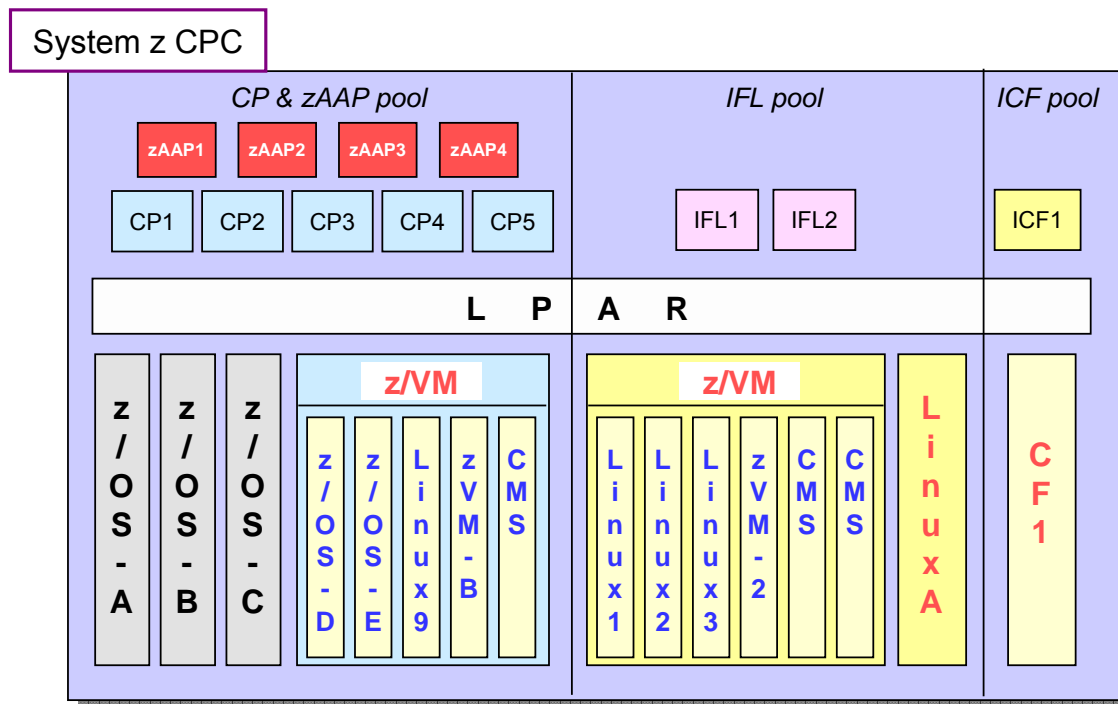
There are three logical partitions; PROD for production, DEV for development and TEST for system programmer use. PROD is set up with four **logical** CPs, DEV with three, and TEST with two. The LPAR microcode maps the work dispatched on each logical processor to any one of the real processors using the priorities defined in the LPAR configuration. For example the relative priorities might be 70 percent for PROD, 25 percent for DEV and 5 percent for TEST. With this configuration, if all partitions have sufficient work to drive the CPC to full capacity then the processor resource would be shared in the above ratios. However, should the PROD partition only need 50 percent, then other partitions can use the spare capacity and extend beyond their nominal share. When the higher priority partition needs more resource, LPAR once again reallocates processing power as appropriate. This allocation is performed on a sub-second basis such that good response times are maintained. As a result, peaks in one workload are accommodated by troughs in another.

RAM is dedicated to each LPAR. If LPARs have been defined so that RAM is overcommitted then not all LPARs can be activated. Some RAM is used by the Channel Subsystem (CSS) and some is used by LPAR itself. The remainder can be divided up amongst all LPARs.

Fibre optic channels (chpids/pchids) can be shared across LPARs. The system programmer specifies which LPARs are allowed to share which channels.

Any operating system IPL'd in a specific LPAR only 'sees' the logical CPs, logical channels, and RAM that have been defined to that LPAR.

## Some LPAR & z/VM configurations



LPAR can support many different configurations.

The one above may not be typical but represents some of the combinations that might be used.

The main points to remember are

1. z/OS can only run on PUs configured as CPs.
2. A CP can be used to run z/OS, z/VM and Coupling Facility Control Code (CFCC). However, the power and number of CPs configured affects the license costs for z/OS and associated products
3. Due to point 2 above, it is more cost efficient to run z/VM & Linux workloads on IFLs.
4. Similar to the above, it is better to use ICFs for Coupling Facility LPARs.
5. You cannot have more zAAPs than CPs on a CPC, however you can configure individual LPARs to have zero or more zAAPs. In the above configuration, a single z/OS LPAR could be configured with just one CP but with 4 zAAPs.
6. zAAPs and zIIPs (not shown above) can only be used in z/OS LPARs.





Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.1

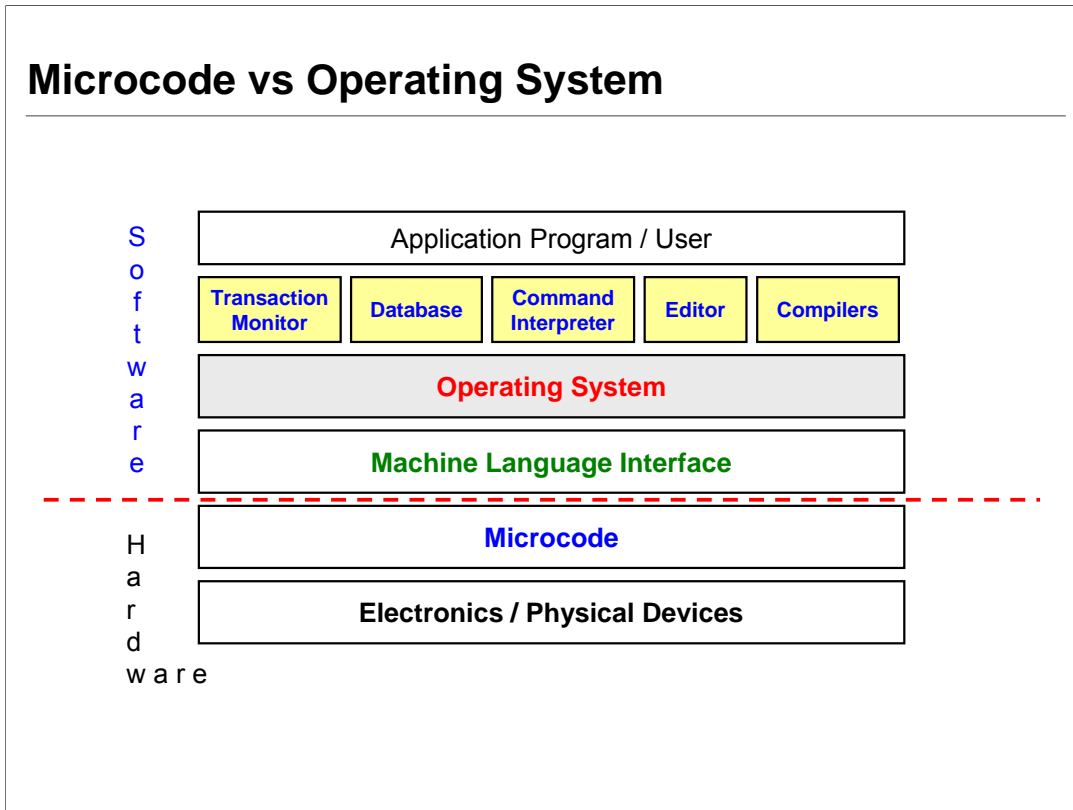
## **z/OS High Level Overview**

---

**Discuss the following topics:**

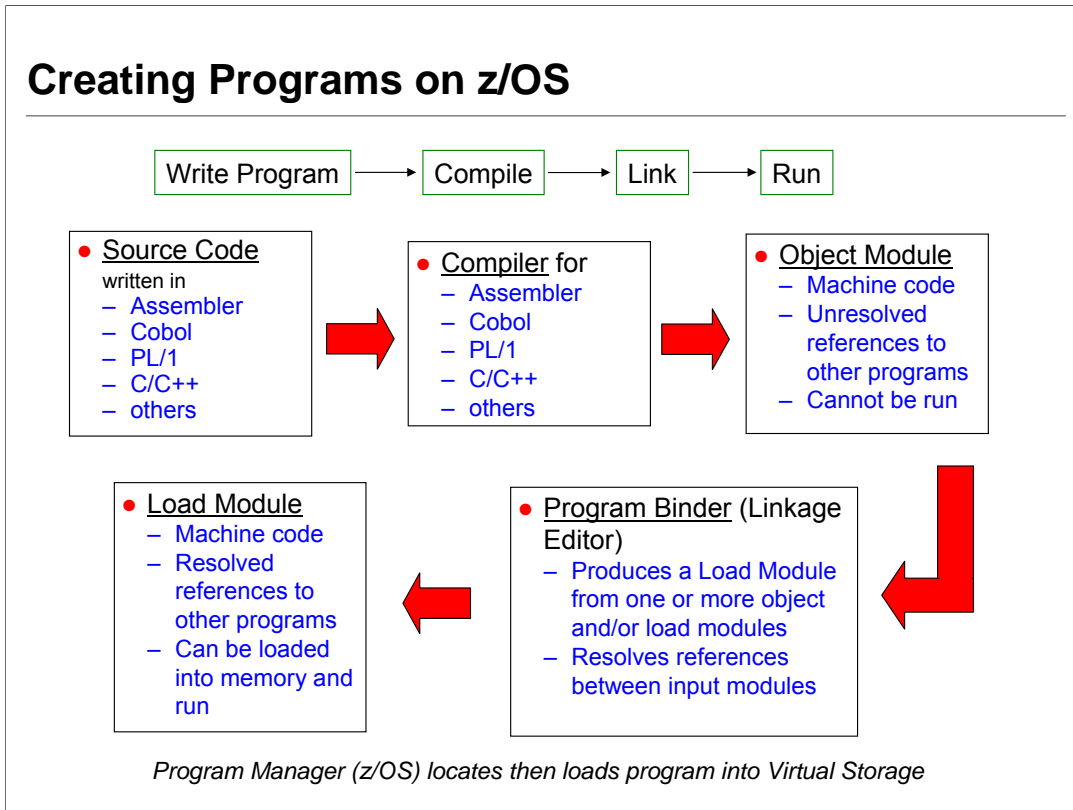


- **MVS Structure & Components**
- **System z Operations Setup**
- **System z Disk Jargon, data types**
- **Unix Services**
- **RACF security system**
- **Running programs / task**



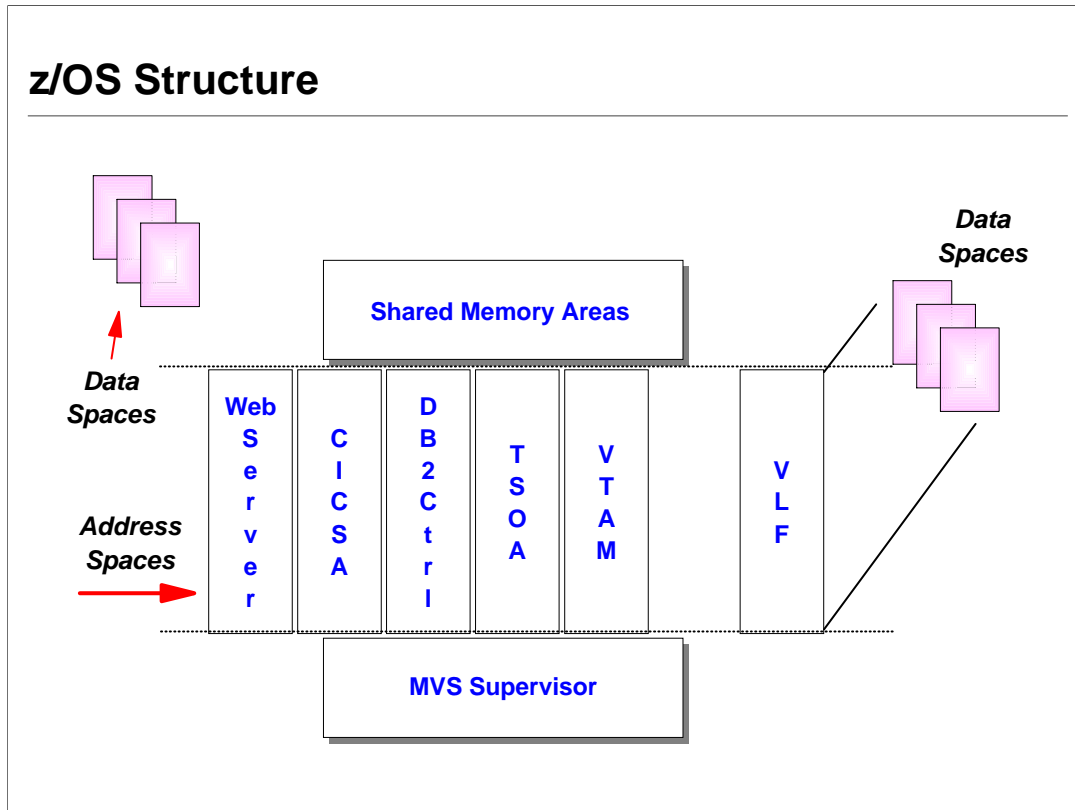
Microcode is an extra layer of software that enhances the functionality of the base electronics. Microcode is not a customer responsibility as it is treated as hardware by the manufacturer. The interface to the microcode is by way of machine language instructions. The foil shows the layers of software and their function in the overall system.

(Other hardware platforms use BIOS which can be thought of as the equivalent of microcode).



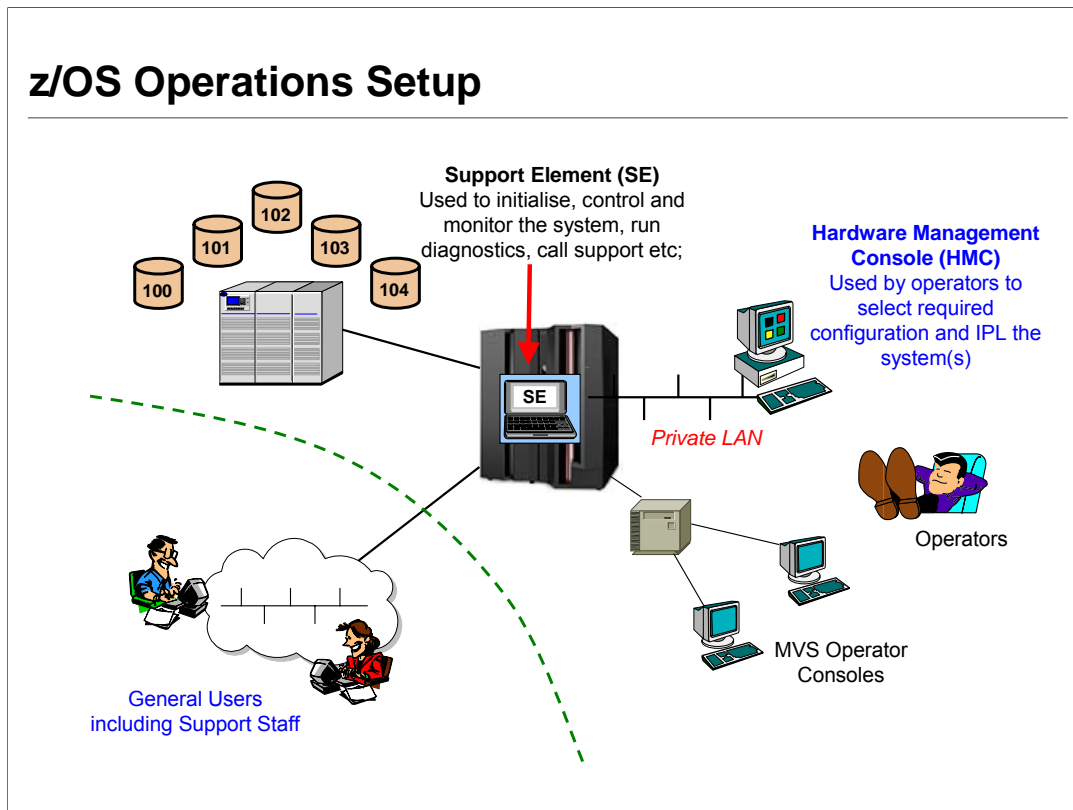
Whether a program is part of z/OS or one created by customers or other third parties, the developer has to go via a series of steps, as shown above.

Note that the compiler is for a specific platform, in our case for an IBM System z architecture machine. The resulting executable code will only run on this type of machine.



At the heart of z/OS is the MVS supervisor, or nucleus. Some other system functions and jobs run in Address spaces. Address spaces contain instructions and data. Data spaces, on the other hand, only contain data. Both address spaces and data spaces are 'virtual' constructs but their data is backed up by real storage either in main storage or on disk (paging disks).

Address spaces will be covered in more detail in a later topic.



The System z server is powered up and IPLed from a Hardware Management Console (HMC), which connects to a Service Element (SE) within the System z CPC (Central Processing Complex)

The base configuration and customisation of the CPC is held on the hard disk within the SE. The z/OS system is IPLed from one or more disks attached via one or more control units and channels.

z/OS communicates with operations personnel via dedicated terminals connected via a channel attached cluster controller. General users connect into the system over a variety of links.

## Some z/OS Components - 1

---

- **MVS Supervisor + DFP (data management)**
  - **Major Operating System components (run as address spaces)**
    - System extensions ..e.g. VLF, LLA, GRS, APPC
    - JES2 (&JES3) - Job Management
    - DFSMS - Systems Managed Storage
    - RACF - Security Management
    - TSO - Time Sharing Option - main system access tool
    - ISPF - Panel display Facility used with TSO
    - WLM - Workload Manager
  - **Transaction Systems**
    - CICS
    - IMS/DC
  - **Database Systems**
    - DB2
    - IMS/DB
- Continued.....*



z/OS is built from multiple components. Many components come as standard and are part of the basic system. Others are optional and need to be selected by the customer. In some cases the customer may use a third party product rather than the IBM one (For example they may use ACF2 instead of RACF).

## Some z/OS Components - 2

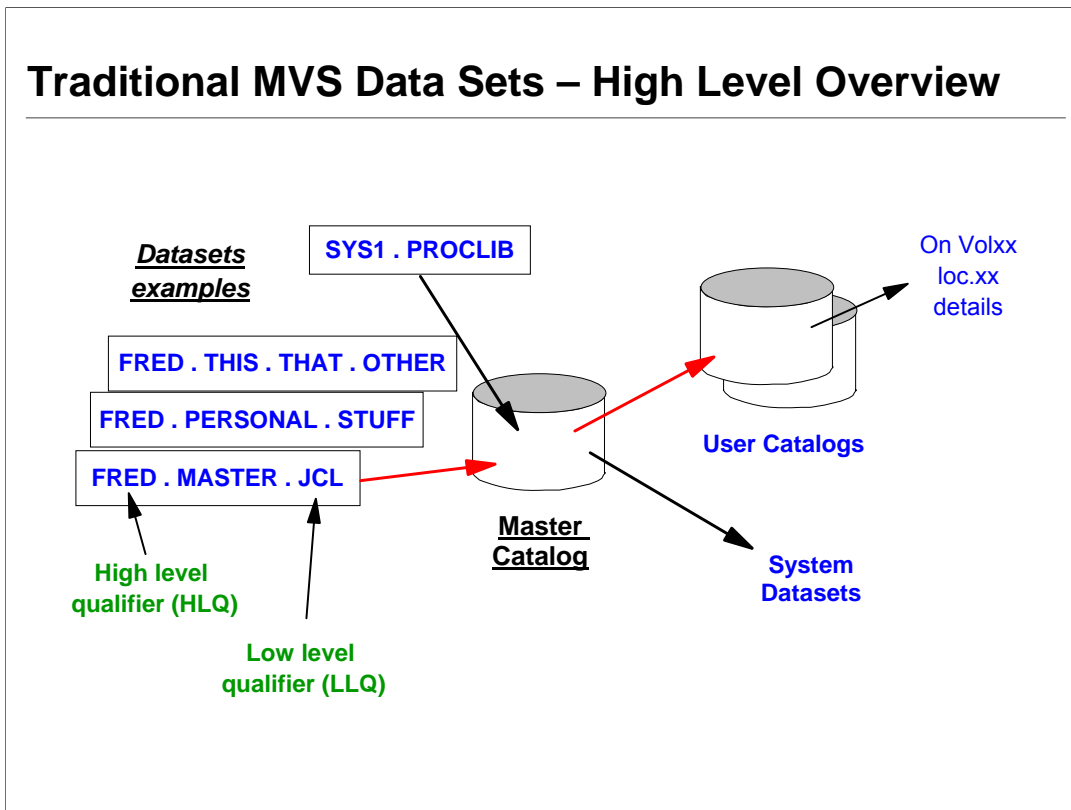
---

- **Storage Management**
  - DFSMS/dss - Data movement / backup / restore
  - DFSMS/hsm - Space and availability management
  - DFSMS/rmm - removable media manager (mainly tape)
  - DFSORT - sort facility
- **Network software and network management**
  - VTAM (SNA) & NCP (Network Control Program for FEPs)
  - TCP/IP & associated functions (e.g. ftp, telnet, nfs)
  - Netview (network management)
- **Capacity/Performance measurement & statistics**
  - RMF - Resource Measurement Facility
  - SLR - Service Level Reporter
  - SMF - Systems Management Facility (Stats recording)
- **Software Management**
  - SMP/E - Software management program / extended



*plus many more both IBM and non-IBM*





The concept of a data set can be very confusing to people coming from a PC or UNIX background as it is not hierarchical in structure. That is, it does not consist of files located in directories.

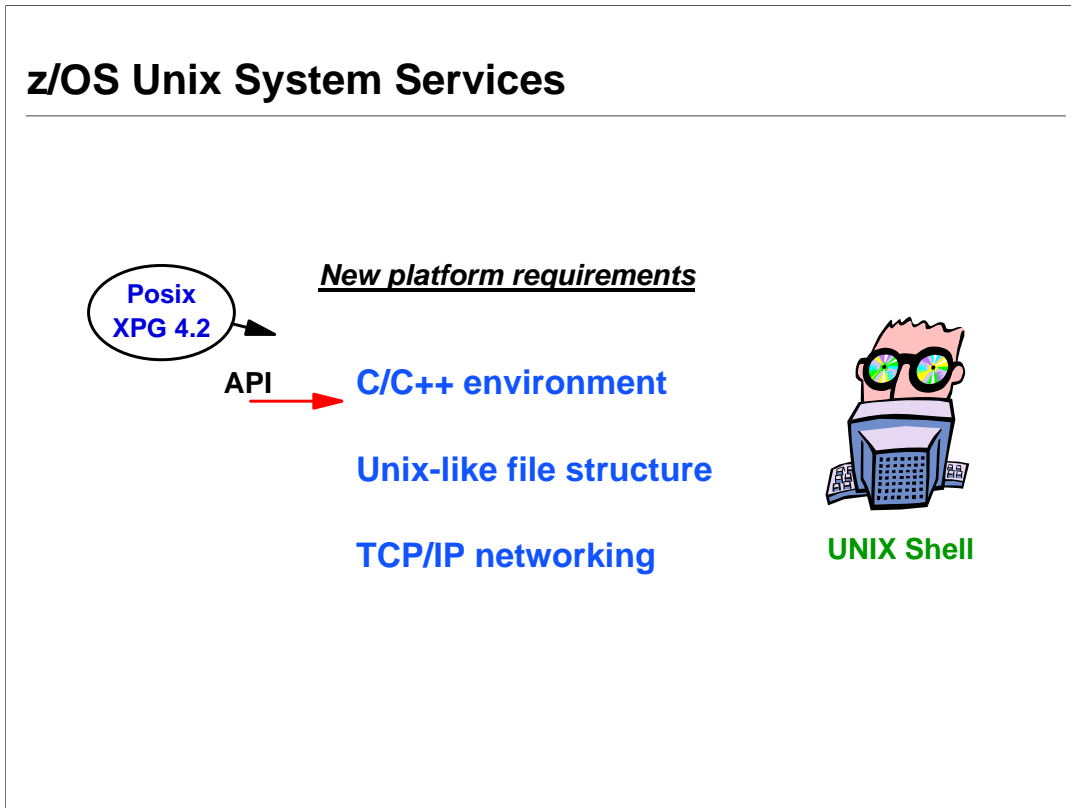
Dataset names are split up into two or more 'qualifiers'. The topmost is called the high level qualifier (HLQ), and the bottom, the low level qualifier (LLQ). Typically the HLQ of a dataset is used to point to a catalog which will have a pointer to the disk volume on which the dataset resides.

## Data Set Types

---

- **Old style**
  - Sequential (QSAM, BSAM) - (flat file almost)
  - PDS (partitioned dataset) - (more like a folder)
  - Random Access (BDAM) [e.g.used by Adabas, IDMS]
- **VSAM**
  - KSDS (Keyed sequential data set) [e.g.used by IMS/DB]
  - ESDS (Entry sequenced dataset) [e.g. used by IMS/DB]
  - RRDS (Relative record dataset)
  - LDS (Linear dataset) [e.g. used by DB2]
- **PDSE**
  - ★ Partitioned dataset extended - (more like a folder)
- **HFS (Hierarchical File System)**
  - ★ much like AIX journal file system (JFS)
  - ★ used by Domino and other UNIX-style applications

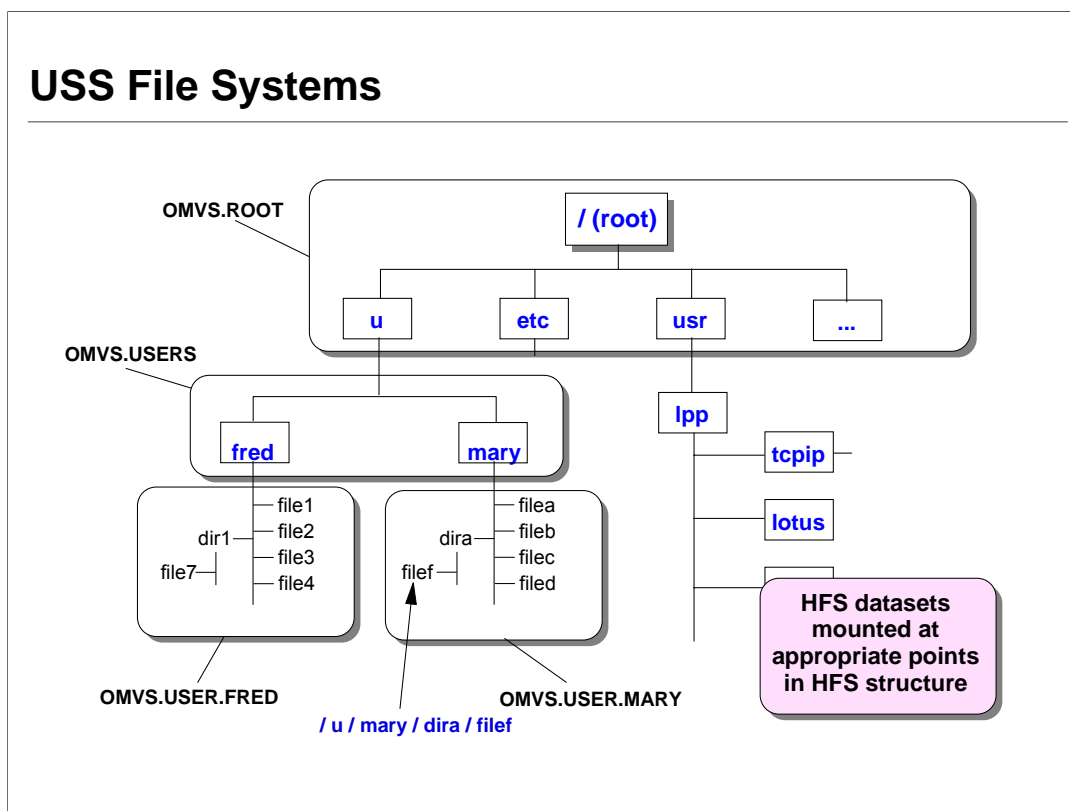




Beginning with MVS/ESA Version 4.3, UNIX System Services were added to the MVS platform. The definitions are known by the name POSIX, which stands for Portable Operating System Interfaces for UNIX. They include both a C programming API and an interactive environment that is referred to as the shell.

The Open Group then became the brand owners of the term UNIX. The Open Groups standards are published in X/Open Portability Guides (XPG). All these definitions describe what to implement, not how to implement. The MVS designers took these standards and modelled UNIX Systems Services after them. OpenEdition was previously the name given to the MVS implementation and is made in such a way that UNIX functions coexist with traditional MVS functions. What was called OpenEdition is now referred to by the more meaningful title of UNIX System Services for z/OS.

These functions and many others are packaged together in the base z/OS system which also includes TCP/IP, the prime network communications protocol.



UNIX applications rely on the existence of a Hierarchical File System (HFS). An example of a UNIX services application is the z/OS Web server, the WebSphere Application Server. The UNIX file system structure is unlike MVS where most data is held in record format and where applications access specific records rather than offsets in a byte stream. Conventional z/OS file structures are not hierarchical.

Access to a data set in MVS is done via a catalog entry and there is no concept of a directory in the PC or UNIX sense. HFS file systems are implemented in z/OS like any other data structure using the DFP component of DFSMS/MVS. The HFS is made up from multiple MVS data sets, of type=HFS and are allocated in the standard manner.

During z/OS UNIX initialization, these datasets are linked together in a logical hierarchy. Each HFS is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be mounted on any directory within the root file system or on a directory within any mounted file system.

Note that zFS file systems are the strategic replacement for HFS ones. zFS file systems are 'contained' within VSAM Linear Data Sets.

# RACF

---

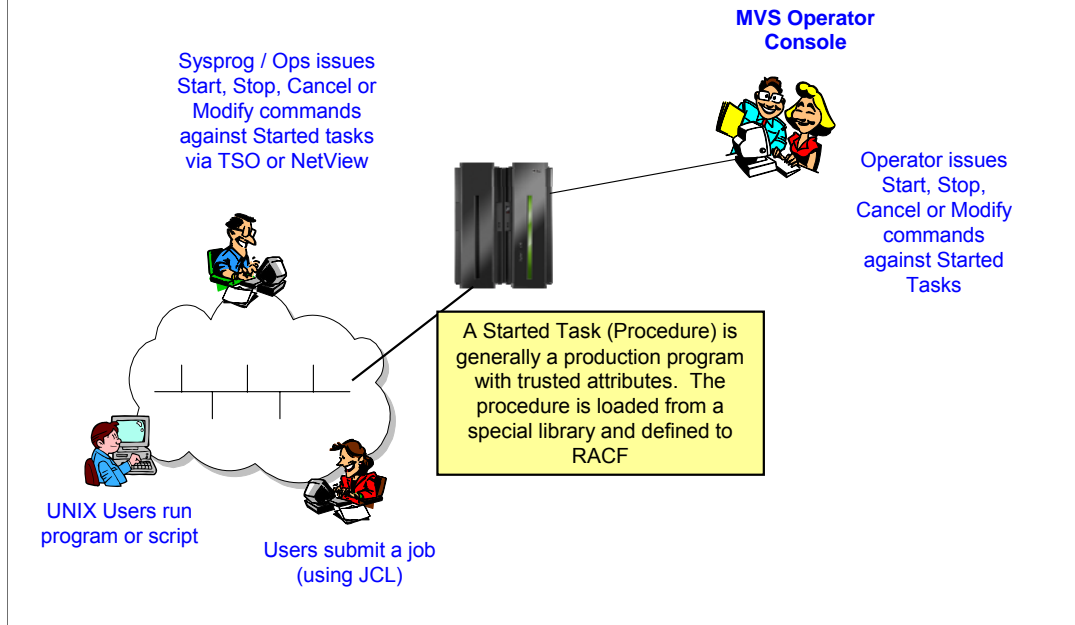
## *Resource Access Control Facility*

- **Security information is held in RACF database**
- **Offers protection of all resources**
  - Data / Datasets / Files (Disk, Tape, etc;)
  - Operations / programs / functions
  - Roles
  - Invoked via the MVS SAF interface
- **APF (Authorised Program Facility)**

MVS mechanism used to insure only 'Authorised programs' can enter supervisor state (kernel mode). In order to be APF authorised, a program must have been loaded from an APF library.



## Ways of Starting Programs



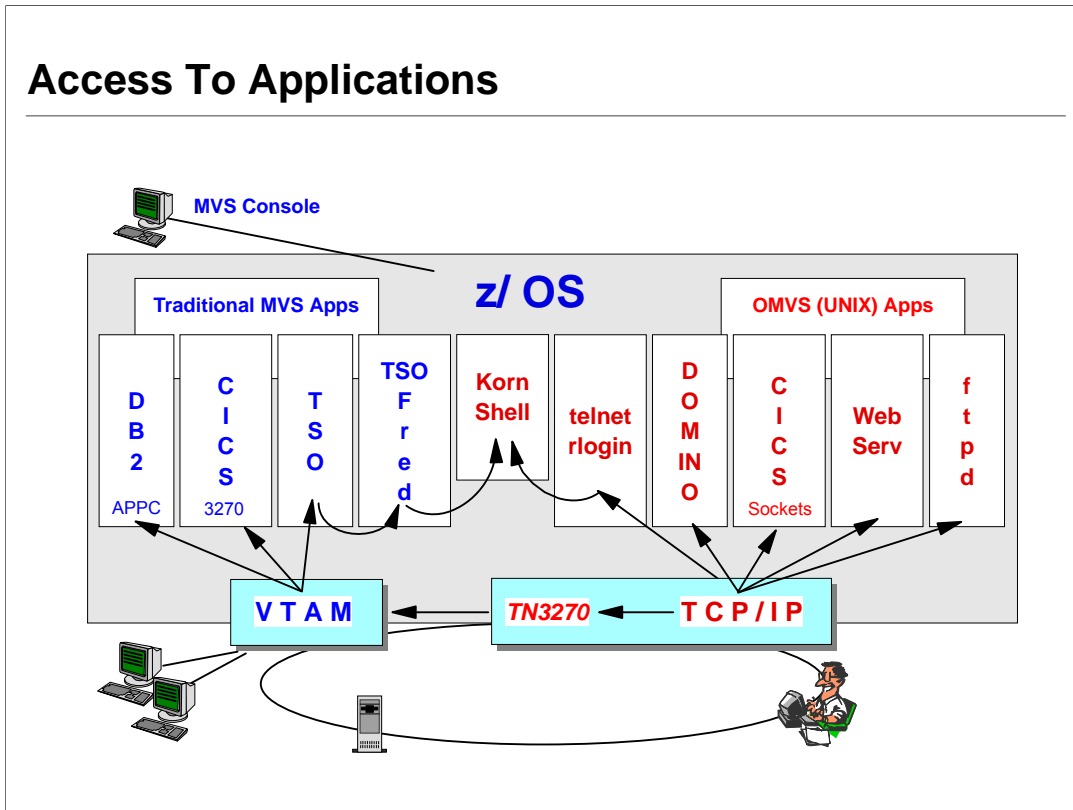
## Job Control Language - JCL

```
-----  
//EWXDISK JOB (ACCOUNTING-INFORMATION),G7.....  
//* LANRES/MVS DISK SERVER *//  
//SERVER EXEC PGM=EWXLDDSK,  
// PARM='FRED .EWXCONFIG .DISKS (NICKNAME FREDDISK....  
//STEPLIB DD DSN=LANRES .EWX131 .SEWXLMOD,DISP=SHR  
//SYSEXEC DD DSN=LANRES .EWX131 .SEWXEXEC,DISP=SHR  
//EWXPRINT DD SYSOUT=*  
//SYSTSPRT DD SYSOUT=*  
-----
```

Command ==> **S u b m i t**

The Job Control Language goes back to the 1960s and punched cards. This is an example of the JCL statements used to start up a server program.

The primary purpose of JCL is to tell the system which program is to run and what resources are required. The resources in this case are typically the names of data sets.



This foil shows a summary of how traditional and new style UNIX applications might be accessed





Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

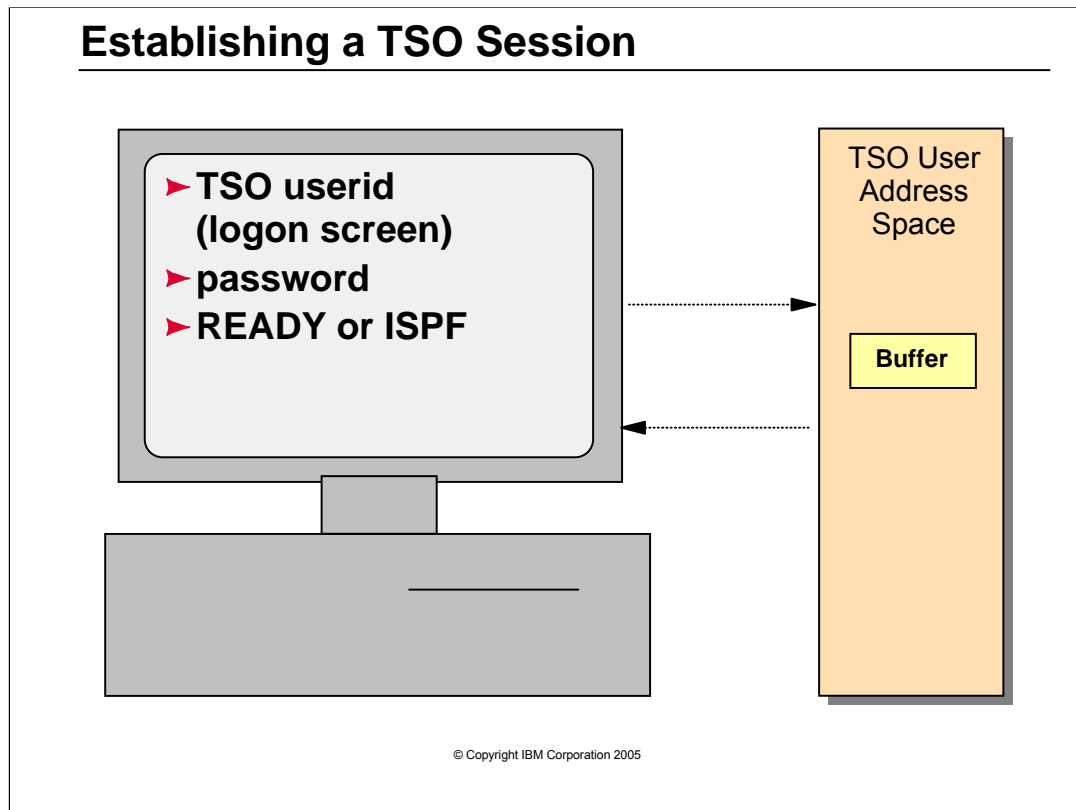
4.1

## Unit objectives

---

After completing this unit, you should be able to understand:

- TSO logon
- ISPF user interface
- Working with ISPF dialog



Performing a **Logon** to a TSO system means starting a TSO/E session and identifying yourself to TSO/E.

To do so you have to provide:

- A unique, valid user identification called a **userid** and
- A valid **password** that is associated with the userid.

(In addition, you might also have to specify an account number, a procedure name, and possibly a RACF GROUP name.)

To actually log on to a TSO system, you have to enter the LOGON TSO command.

The logon request is passed on to Terminal Control Access Space (TCAS). If TCAS accepts the logon request, it creates the user's address space and initializes its buffers.

Once the user address space is set up, TCAS declares the user address space to VTAM, which now handles all further communications.

## Time Sharing Option / Extensions

```

----- TSO/E INFORMATION CENTER FACILITY -----
OPTION ==>

READY

----- TSO/E LOGON-----

Enter LOGON parameters below:                RACF LOGON parameters:
Userid   ==> AUES100
Password ==>
Procedure ==> LOGON
Acct Nbr ==> ACCNT#
Size     ==> 4096
Perform  ==>
Command  ==> isppdf

Enter an 'S' before each option desired below:
-Nomail      -Nonotice      -Reconnect      -OIDcard

PF1/PF13 ==> Help   PF3/PF15 ==> Logoff  PA1 ==> Attention  PA2 ==> Reshow
You may request specific help information by entering a '?' in any entry

© Copyright IBM Corporation 2005

```

**TSO/E** is a base element of z/OS's System Services function.

You can use TSO/E in many environments, such as with the Interactive System Productivity Facility/Program Development Facility (ISPF/PDF), session manager, and line mode TSO/E. How you interact with TSO/E depends upon the environment.

### ISPF/PDF

Interactive System Productivity Facility (ISPF) and its **Program Development Facility (ISPF/PDF)** provide panels through which users can interact with the system. PDF provides an environment for the development, testing and execution of programs and applications.

### Session Manager

The TSO/E session manager is an interface to line mode TSO/E. It saves the commands that you enter and the responses that you receive and allows you to redisplay or print them.

### Line Mode

A command-oriented interaction with the z/OS system, one command (-line) at a time. On most display terminals, when you are in line mode TSO/E, three asterisks, **\*\*\***, on the screen means that when you read the screen and press the **Enter** key. TSO/E will present you with a new screen and allow you to continue.

## Interactive System Productivity Facility

```

Menu  Utilities  Compilers  Options  Status  Help
-----
                    ISPF Primary Option Menu

0  Settings      Terminal and user parameters      User ID . . : AUES100
1  View          Display source data or listings   Time. . . . : 14:41
2  Edit          Create or change source data      Terminal. . : 3278
3  Utilities     Perform utility functions        Screen. . . . : 1
4  Foreground   Interactive language processing   Language. . : ENGLISH
5  Batch        Submit job for language processing Appl ID . . : ISR
6  Command      Enter TSO or Workstation commands TSO logon . : LOGON
7  Dialog Test  Perform dialog testing           TSO prefix: AUES100
8  LM Facility  Library administrator functions   System ID . : SYS1
9  IBM Products IBM program development products MVS acct. . : ACCNT#
10 SCLM         SW Configuration Library Manager Release . . : ISPF 4.8
11 Workplace   ISPF Object/Action Workplace

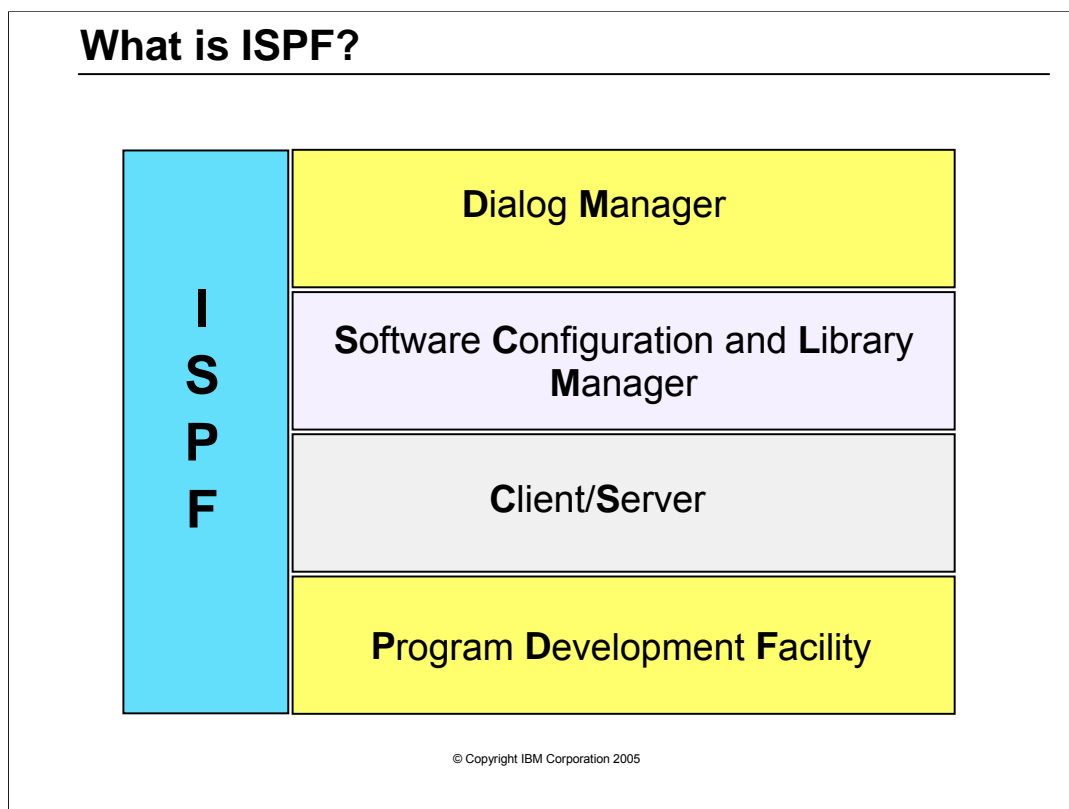
Enter X to Terminate using log/list defaults

Option ==>
F1=Help      F3=Exit      F10=Actions  F12=Cancel

```

© Copyright IBM Corporation 2005

This is the standard ISPF Primary Option Menu (POM). This is customizable so you might not have this same panel at your installation.



The **Interactive System Productivity Facility (ISPF)** provides a panel-driven interface and can be seen as an extension of the z/OS Time Sharing Option (TSO) host system on which it runs. The services provided through ISPF complement those of the host system.

ISPF is similar to a control program or access method in that it provides services to dialogs (applications) during their execution. The types of services provided by ISPF are:

- Display services
- Variable services
- Table services
- Dialog test facilities
- And so forth

A dialog receives requests and data from a user at a terminal. The dialog responds by using ISPF services to obtain information from, or enter information into, the z/OS system.

ISPF consists of four major components:

### **The Dialog Manager (DM)**

The Dialog Manager provides services to dialogs and end users. PDF assists dialog or application developers by providing development services.

### **The Software Configuration Library Manager (SCLM)**

The SCLM component offers services to application developers to manage their application development libraries. SCLM is a library facility that supports projects in developing complex software applications. SCLM supports the software development cycle of an application from the program design phase to release of the final product.

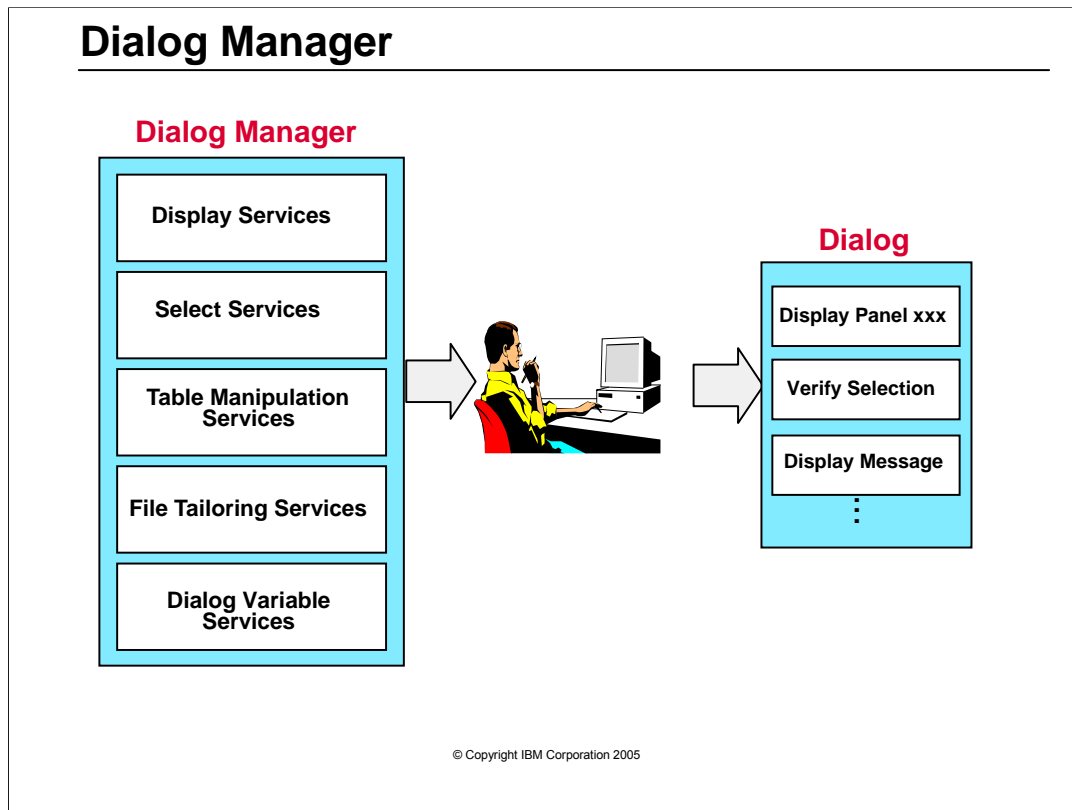
### **The Client/Server component (C/S)**

The Client/Server part enables you to run ISPF on a programmable workstation, to display the panels using the display function of your workstation operating system, and to integrate workstation tools and data with host tools and data.

### **The Program Development Facility (PDF)**

The PDF component of ISPF is an integrated work environment used to develop programs, dialogs, and documents. It provides numerous productivity-improving functions.

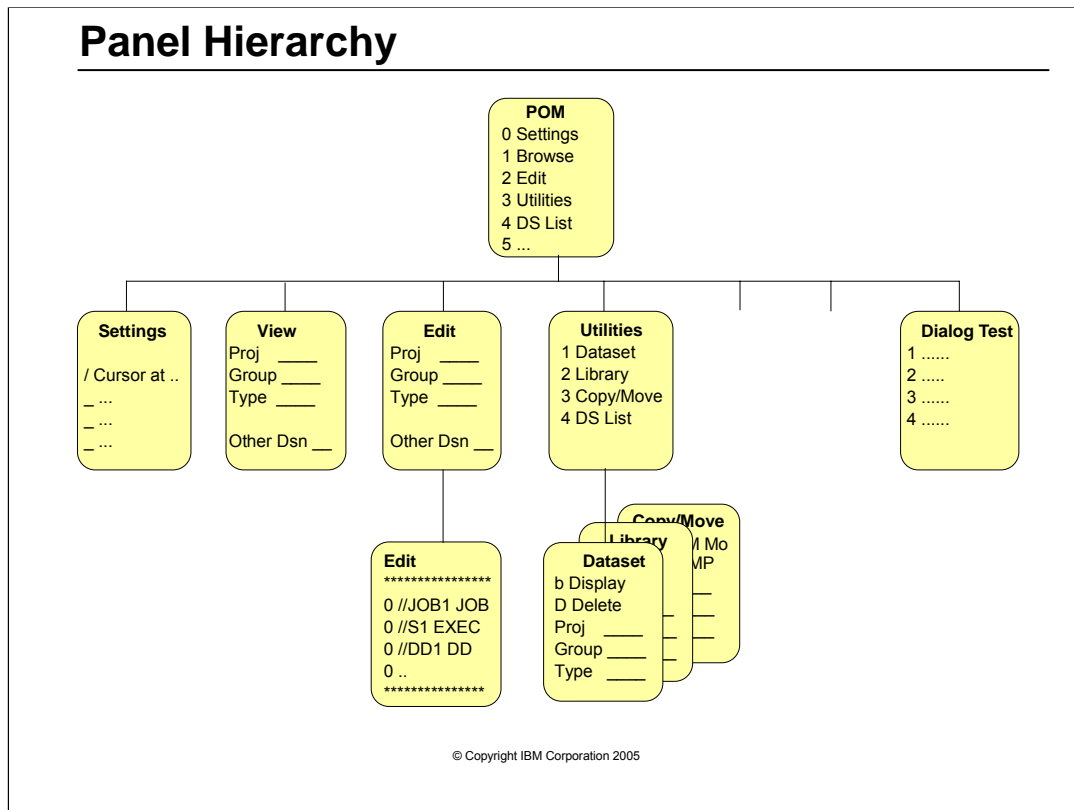
We do not deal with the Software Configuration and Library Manager, or the ISPF Client/Server component in the rest of this class.



The **Dialog Manager** provides various kinds of services to dialogs during their execution, and controls the interaction of the dialog's elements. For example, ISPF can issue requests for panels to be displayed, and screens to be formatted. It can verify, process, store input, create output, and so forth. ISPF can also function as a simplified data management system for small amounts of data stored in tables.

ISPF's dialog management services include:

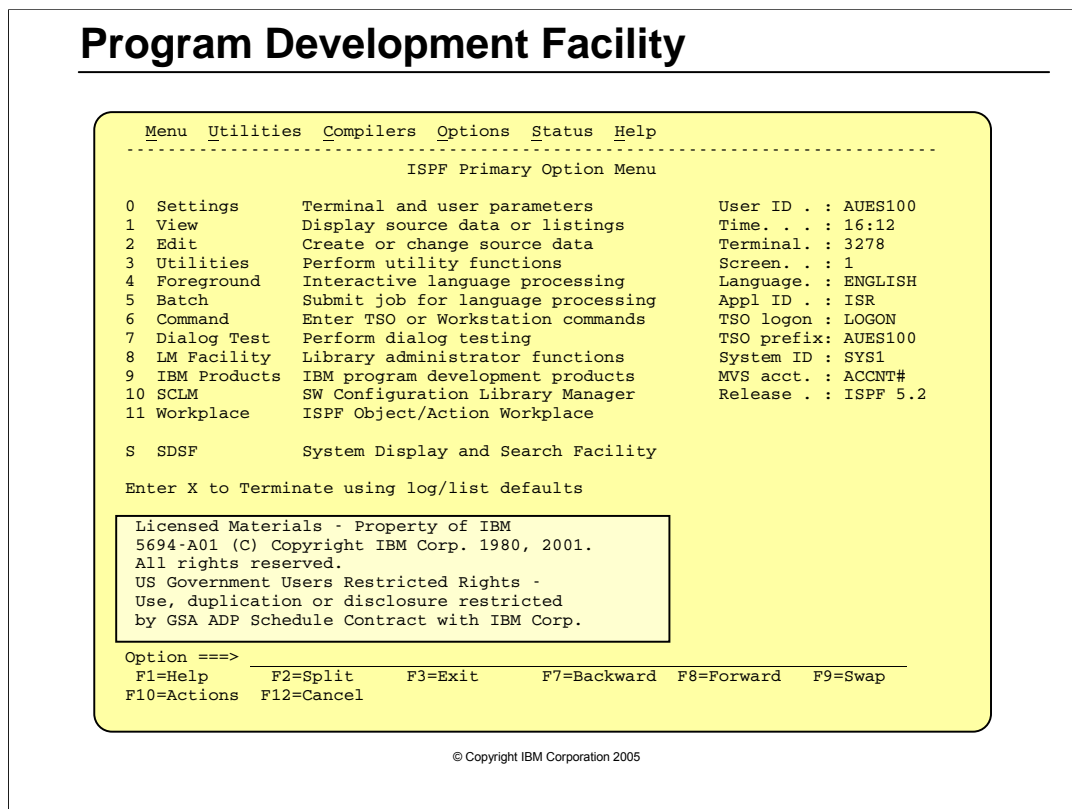
- Display services for panels, pop-up windows, pull-down menus, messages, tables, etc;
- Select services for panels and functions
- Table manipulation services
- File Tailoring Services
- Services to handle and display dialog variables



This foil shows three common panel types, arranged in a hierarchy. The starting panel at the top is a **Selection** panel, known as a **Primary Option Menu**.

From the Primary Option Menu, **Data Entry** and other selection panels can be selected. Panels used for features like View and Edit use scrollable panels.





The **Program Development Facility** component of ISPF provides the dialog or application developer with a variety of services to create and test applications. PDF lets you manage data sets, create and test panels and messages, generate tables, keep track of variable values, set checkpoints, trace applications, and so forth.

This visual shows the PDF's main menu. It contains the options that you can use to create your own applications online. It is possible to customize the ISPF Primary Option Menu, therefore the PDF main menu at your home system might offer additional or alternative choices to the following options:

**0 - Settings** Lets you display and change ISPF parameters such as function key definitions, panel display, display colors, and so forth.

**1 - View** View mode allows you to view or browse data sets.

**2 - Edit** The edit mode lets you create or update data sets and members. The ISPF/PDF Editor provides macros and models which help you change or create data sets.

**3 - Utilities** The option 3 of the ISPF/PDF Main Menu offers you a collection of system utility and data set management functions, including printing, renaming, deleting, and so forth, a member or data set.

**4 - Foreground** This option is intended to interactively run language processing programs, including assembler, COBOL, VS/FORTRAN, PL/I, VS Pascal, and SCRIPT/VS.

**5 - Batch** Generates and submits job control statements and command streams to execute language processing programs in background.

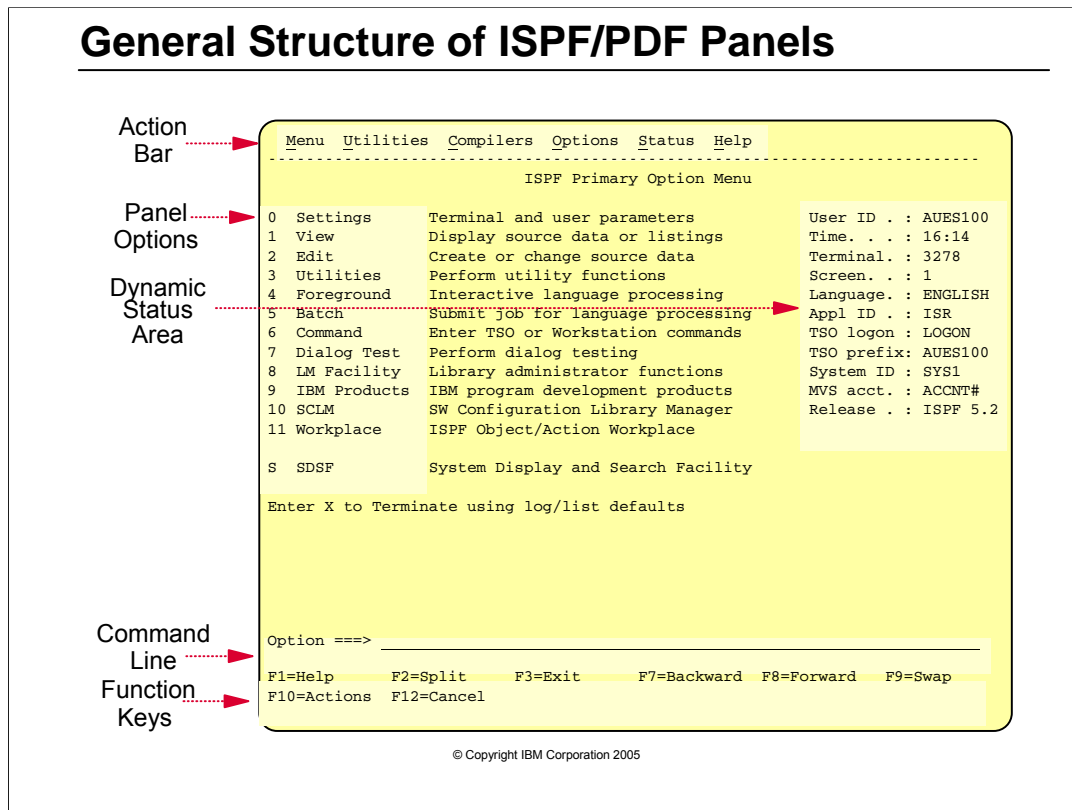
**6 - Command** This ISPF/PDF Main Menu option enables you to enter TSO commands, or invoke CLISTS or REXX EXECs, while ISPF/PDF remains active.

**7 - Dialog Test** Allows for the testing of dialog elements (panels, messages, and so forth.) of an application before assembling them in an application.

**8 - LM Utilities** Library Management Facility ensures you are working with the latest level of your development libraries, or controlling who is updating what. LMF has been superseded by SCLM.

**9 - IBM Products** Provides an interface to other IBM products.

**10 - SCLM** This is the main menu option to work with the Software Configuration and Library Manager component of ISPF.



An ISPF/PDF panel consists of the following sections:

### Action Bar

Most ISPF/PDF panels have an action bar at the top. The action bar offers a set of functions grouped in categories.

### Panel Options

This area contains a list of all options that can be selected from this panel.

### Dynamic Status Area

The dynamic status area is ISPF/PDF main menu-specific and displays a set of important ISPF/PDF settings.

### Command Line

The command line enables you to execute TSO commands, invoke CLISTs or REXX EXECs, branch to another panel, and so forth, or to enter your selection.

### Function Keys

This area is used to display the

## Navigating in ISPF/PDF

### Action Bar

Menu Utilities Compilers Options Status Help

### Point-and-Shoot

```

0  Settings      Terminal and user parameters
1  View         Display source data or listings
2  Edit         Create or change source data
3  Utilities     Perform utility functions
.

```

### Option Number

```

0  Settings      Terminal and user parameters
1  View         Display source data or listings
2  Edit         Create or change source data
3  Utilities     Perform utility functions
.

```

Options ==> 3

### Function Keys

F1=Help      F3=Exit      F7=Bkwd      F8=Fwd  
 F10=Actions   F11=Retrieve   F12=Cancel

© Copyright IBM Corporation 2005

ISPF offers a user a number of ways to navigate through its dialogs:

#### Action Bars

Action bar choices vary from panel to panel, as do the choices from their pull-downs.

An action bar groups the user options into categories. Only the categories are displayed on a panel. To see the options within a category, place the cursor on an action bar choice and press the *Enter* key. ISPF/PDF now displays the entire list of options for this one category. To select an option, enter the corresponding selection code or move the cursor to the proper selection and press *Enter*. *Options marked with \* are unavailable.*

#### Point-and-Shoot

Point-and-shoot fields are cursor-sensitive. Moving the cursor on a point-and-shoot field

**Note:** A command entered on a command line is always processed prior to any point-and-shoot selection.

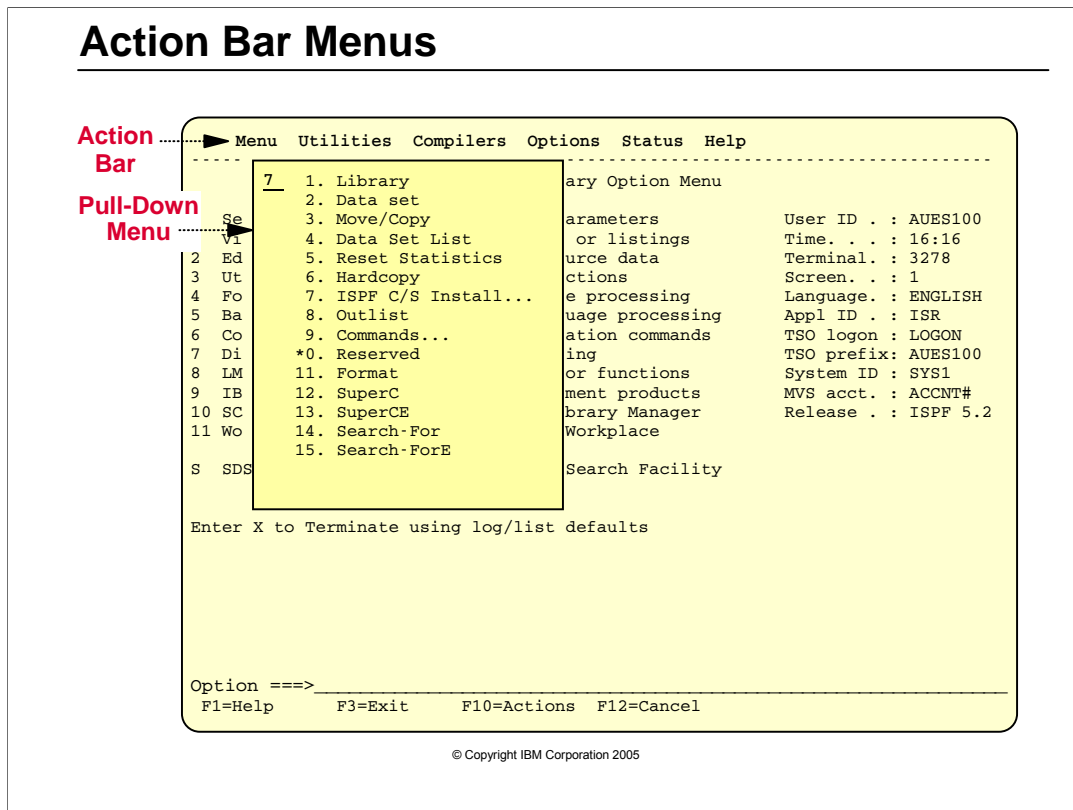
#### Selection by Option Code

The traditional way of selecting a panel option is by entering the selection code, that is the number or characters that corresponds to a panel option. This form of selection is available on most ISPF/PDF panels.

#### Function Keys

Function keys allow a user to perform a predefined action by simply pressing a specific key. Function keys and their settings are usually displayed at the bottom of a screen. The function key display can be turned off to give you two extra lines on the screen.





Most of the ISPF/PDF panels have action bars at the top.

To move the cursor from any screen location to the action bar:

- Use the cursor keys to place it on the action bar choice
- Enter *ACTIONS* on any command line and press the *Enter* key
- Press F10 or the *Home* key

Use the tab key to move the cursor among action bar choices. To display the pull-down menu of one of the action bar choices, place the cursor on it and press the *Enter* key. To select a choice from the pull-down menu, type its number in the entry field (underlined) or place the cursor on your choice and press *Enter*. To leave a pull-down menu without further selection press F12 (Cancel).

## Pop-Up Window

---

Menu   Utilities   Compilers   Options   Status   Help

---

	ISPF Primary Option Menu	Invalid Option
0	Settings      Terminal and user parameters	User ID . : AUES100
1	View          Display source data or listings	Time. . . : 16:16
2	Edit          Create or change source data	Terminal. : 3278
3	Utilities      Perform utility functions	Screen. . : 1
4	Foreground    Interactive language processing	Language. : ENGLISH
5	Batch         Submit job for language processing	Appl ID . : ISR
6	Command      Enter TSO or Workstation commands	TSO logon : LOGON
7	Dialog Test   Perform dialog testing	TSO prefix: AUES100
8	LM Facility   Library administrator functions	System ID : SYS1
9	IBM Products  IBM program development products	MVS acct. : 5820
10	SCLM         SW Configuration Library Manager	Release . : ISPF 5.2
11	Workplace    ISPF Object/Action Workplace	
S	SDSF         System Display and Search Facility	

Enter X to Terminate using log/list defaults

**Pop-Up Window**

→

The option that was entered was not valid.

Option ==> Z

F1=Help      F3=Exit      F10=Actions   F12=Cancel

© Copyright IBM Corporation 2005

**Pop-up windows** are windows appearing on top of other panels.

In the example shown above, the long message is displayed in a pop-up window. (The long message is displayed when F1 is pressed after a short message is issued.)

**Modal pop-up windows** are a special kind of pop-up window. They require the user's interaction, that is, some kind of reply, before the underlying dialog continues.

**Modeless pop-up windows**, in contrast, allow you to interact with the dialog, before you interact with the window.

## Settings - Option 0

```

Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
-----
                                ISPF Settings

Options
Enter "/" to select option
/  Command line at bottom
/  Panel display CUA mode
/  Long message in pop-up
/  Tab to action bar choices
/  Tab to point-and-shoot fields
/  Restore TEST/TRACE options
/  Session Manager mode
/  Jump from leader dots
/  Edit PRINTDS Command
/  Always show split line
-  Enable EURO sign

Print Graphics
Family printer type 2
Device name . . . . _____
Aspect ratio . . . . 0

General
Input field pad . . N
Command delimiter . ;

Terminal Characteristics
Screen format  1  1. Data    2. Std    3. Max    4. Part

Terminal Type  6  1. 3277  4. 3278A  7. 3278CF  a. 3278DE
                2. 3277A  5. 3290A  8. 3277KN  b. 3278FI
                3. 3278  6. 3278T  9. 3278KN

Command ==> _____

F1=Help      F2=Split    F3=Exit     F7=Backward F8=Forward  F9=Swap
F10=Actions  F12=Cancel

```

© Copyright IBM Corporation 2005

The *ISPF Settings* option allows you to display and modify selected ISPF parameters. The settings panel can be invoked from the primary option menu as option 0 or from any panel by entering the `SETTINGS` command on a command line. All settings are persistent across ISPF sessions except some of those available from the *Identifier* action bar.

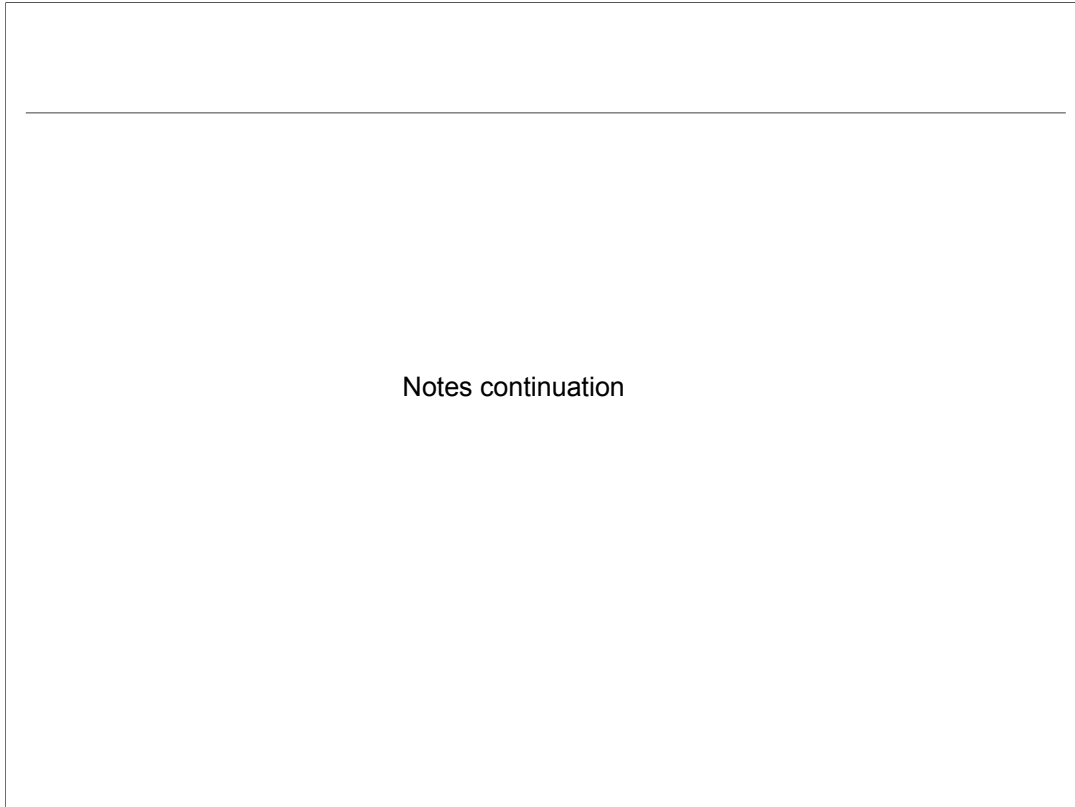
The main ISPF Settings panel has setting option displayed in four areas on the panel:

- Options
- Terminal characteristics
- Print Graphics
- General

Additional settings are available from the action bars.

Some individual settings can be set from the command line using the following commands:

< see next page >



**FKA** Controls the display of the function key area at the bottom of the screen.

**KEYLIST** Invokes the Keylist Utility when entered without parameters.

**KEYS** Invokes the appropriate utility to modify function keys for the panel from which the command was invoked.

**PFSHOW** Almost synonymous with the FKA command, PFSHOW controls the display of the function key area at the bottom of the screen.

**ZKEYS** Invokes the PF Key Definitions and Labels panel.

**COLOR** Invokes the Global Colour Change Utility to modify the display of all colours.

**CUAATTR** Invokes the CUA Attribute Change Utility to modify the colour, intensity, and highlight of CUA panel elements.

**PSCOLOR** Invokes the CUA Attribute Change Utility to modify the colour, intensity, and highlight of CUA panel elements. This command indexes you directly to the point-and-shoot entry.

**ENVIRON** Invokes the ISPF ENVIRON Command Settings panel allowing you to enable traces, dumps, and obtain information about your terminal or terminal emulator.

**PANELID** Turns the display of the panel ID on or off. This setting does not persist across ISPF sessions and only affects the screen from which it was entered.

**MSGID** Turns the display of the message ID on or off. This setting does not persist across ISPF sessions and only affects the screen from which it was entered.

**SCRNAME** Turns the display of the screen name on or off and assigns a screen name to a logical screen. This setting does not persist across ISPF sessions and only effects the screen from which it was entered.

**ISPFVAR** Controls some of the settings on the main ISPF Settings panel.



## Utilities - Option 3

Menu	Help
-----	
Utility Selection Panel	
1 Library	Compress or print data set. Print index listing. Print rename, delete, browse, edit or view members
2 Data Set	Allocate, rename, delete, catalog, uncatalog, or display information of an entire data set
3 Move/Copy	Move, copy, or promote members or data sets
4 Dslist	Print or display (to process) list of data set names. Print or display VTOC information
5 Reset	Reset statistics for members of ISPF library
6 Hardcopy	Initiate hardcopy output
7 ISPF C/S	Install ISPF C/S workstation code from MVS to your workstation.
8 Outlist	Display, delete, or print held job output
9 Commands	Create/change an application command table
* Reserved	This option reserved for future expansion.
11 Format	Format definition for formatted data Edit/Browse
12 SuperC	Compare data sets (Standard Dialog)
13 SuperCE	Compare data sets Extended (Extended Dialog)
14 Search-For	Search data sets for strings of data (Standard Dialog)
15 Search-ForE	Search data sets for strings of data Extended (Extended Dialog)
Option ==>	
F1=Help	F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F10=Actions	F12=Cancel

© Copyright IBM Corporation 2005

The *Utilities Selection Panel*, option 3 of the ISPF/PDF main menu, offers a collection of tools and utilities to work with data sets and ISPF.

Here is a brief overview of the choices of the *Utilities Selection Panel*:

**Library** Print an index, or data set, work with data set members, compress a data set, or display data set information.

**Data Set** Utility to allocate, rename, delete, catalog, or uncatalog a data set and display data set information. It also offers some VSAM functions.

**Move/Copy** Tool for moving or copying data sets.

**Dslist** Display and/or print a list of data set names according to their catalog or VTOC entry.

**Reset** Delete or reset ISPF library statistics.

**Hardcopy** Print a data set.

**ISPF C/S** Install the ISPF C/S workstation code from z/OS or OS/390 on your workstation.

**Outlist** Handling of held jobs, that is, functions to list, delete, print, or queue held job output.

**Commands** Create/change application command table.

**Reserved** Reserved for future releases.

**Format** Provided for support of the IBM 5550 terminal using the Double-byte Character Set (DBCS).

**SuperC** Tool to quickly compare the contents of two data sets.

**SuperCE** An extended version of the SuperC dialog.

**Search-For** Lets you search data sets for strings of data.

**Search-ForE** An extended version of the Search-For dialog.

## ISPF Command Shell - Option 6

```

Menu List Mode Functions Utilities Help
-----
                ISPF Command Shell

Enter TSO or Workstation commands below:

===> listds tsoe.panels

-----

Place cursor on choice and press enter to Retrieve command

=> lista st h
=> printds dataset(jcl.cntl) members
=> profile
=> submit jcl(test)
=>
=>
=>
=>
=>
=>

ISPF Command ===>
F1=Help      F2=Split    F3=Exit     F7=Backward F8=Forward  F9=Swap
F10=Actions  F12=Cancel

```

© Copyright IBM Corporation 2005

The *ISPF command shell*, ISPF/PDF option 6 allows TSO commands, CLISTs, and REXX execs to be executed under ISPF. Commands entered on the ISPF command shell are appended to a list of most recent issued commands from where they can be retrieved.

**Note:** ISPF allows TSO commands to be entered in the command input field of any panel.

The commands have to be prefixed with **TSO**.

Example:

Command ===> TSO LISTC

When issuing a command from a panel's command line, you are limited to the length of the command line. The ISPF command shell allows you to enter commands wrapping to the next lines.

## SDSF Primary Option Menu

```

Display  Filter  View  Print  Options  Help
-----
HQX7740 ----- SDSF PRIMARY OPTION MENU -----
COMMAND INPUT ==>                                SCROLL ==> CSR

DA   Active users          INIT  Initiators
I    Input queue          PR    Printers
O    Output queue         PUN   Punches
H    Held output queue    RDR   Readers
ST   Status of jobs      LINE  Lines
                                     NODE  Nodes
LOG  System log          SO    Spool offload
SR   System requests     SP    Spool volumes
JC   Job classes
SE   Scheduling environm RM   Resource monitor
RES  WLM resources       CK   Health checker
ENC  Enclaves
PS   Processes          ULOG  User session log

END   Exit SDSF

F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=IFIND   F6=BOOK
F7=UP     F8=DOWN    F9=SWAP  F10=LEFT   F11=RIGHT  F12=RETRIEVE

```

The ISPF/PDF main menu selection *S SDSF* displays the **System Display and Search Facility (SDSF)**, a product that lets you display active users and job input queues, browse job output, show logs, check printers, and so forth.

Filtering can be used on many of the SDSF panels. To see what filtering is in effect type the following on the command line `==> SET DISPLAY`

Many SDSF panels have a column where action characters (commands) may be entered. The name of this column is 'NP'. To get additional prompts for action characters do the following:-

- Select the 'OPTIONS' dropdown
- Select '1' – (Set action character display) press Enter
- Select '1' - (Display action characters with descriptions) press Enter

## SDSF – Display Address Spaces (DA)

```

Display Filter View Print Options Help
-----
SDSF DA SYS1  SYS1      PAG  0 SIO   16 CPU  2      LINE 1-35 (94)
COMMAND INPUT ==>
PREFIX=**  DEST=(ALL)  OWNER=**  SORT=JOBNAME/A  SYSNAME=
          SCROLL ==>  CSR
NP  JOBNAME  StepName  ProcStep  JobID  Owner  C  SysName  Pos  DP  Real  Paging
*MASTER*
ALLOCAS  ALLOCAS
ANTAS000  ANTAS000  IEFPROC
ANTMAIN  ANTMAIN  IEFPROC
APPC     APPC     APPC
ASCH    ASCH    ASCH
BAESTC  BAESTC  BAESTC  STC01161  BAESTC
BPXOINIT  BPXOINIT  BPXOINIT
CATALOG  CATALOG  IEFPROC
CBDQDISP  CBDQDISP  TDIS    STC01173  STCHCM
CICSP1  CICSP1  CICS    STC01130  STCCICP
CNMS    CNMS    AOFAPPL  STC01186  STCCNMS
CNMSAM  CNMSAM  HSAMPROC
CNMSSSI  CNMSSSI  AOFASSI
CONSOLE  CONSOLE
CSQACHIN  CSQACHIN  PROCSTEP  STC01146  STCMQS
CSQAMSTR  CSQAMSTR  PROCSTEP  STC01137  STCMQS
DAS13    STEP1    STC01115  DASUSER
DBRCTM  DBRCTM  IEFPROC  STC01185  STCIMS
DB2DDBM1  DB2DDBM1  IEFPROC  STC01149  STCDB2
DB2DDIST  DB2DDIST  IEFPROC  STC01153  STCDB2
DB2DIRLM  DB2DIRLM  STC01143  STCDB2
DB2DMSTR  DB2DMSTR  IEFPROC  STC01134  STCDB2

```

Selecting SDSF option DA (Display Active Users) really means Display all Address Spaces, which are the major work elements in z/OS

## Time for Lab Exercise

---

**Exercise 1**  
Logon to TSO  
Use SDSF (basic investigation)



© Copyright IBM Corporation 2005

### **Exercise 1**

Logon to TSO  
Use SDSF to do some basic investigation

## **Unit Summary**

---

- **Interactive System Productivity Facility (ISPF)**
  - Dialog Manager
  - Program Development Facility (PDF)
- **General Panel Structure**
- **Action Bar Menus**
- **Pop-Up Menus**
- **ISPF Primary Option Menu**
  - 0 Settings
  - 1 View a Data Set
  - 2 Edit a Data Set
  - 3 Utilities
  - 6 ISPF Command Shell
  - S System Display and Search Facility

© Copyright IBM Corporation 2005

---

# **Unix System Services**

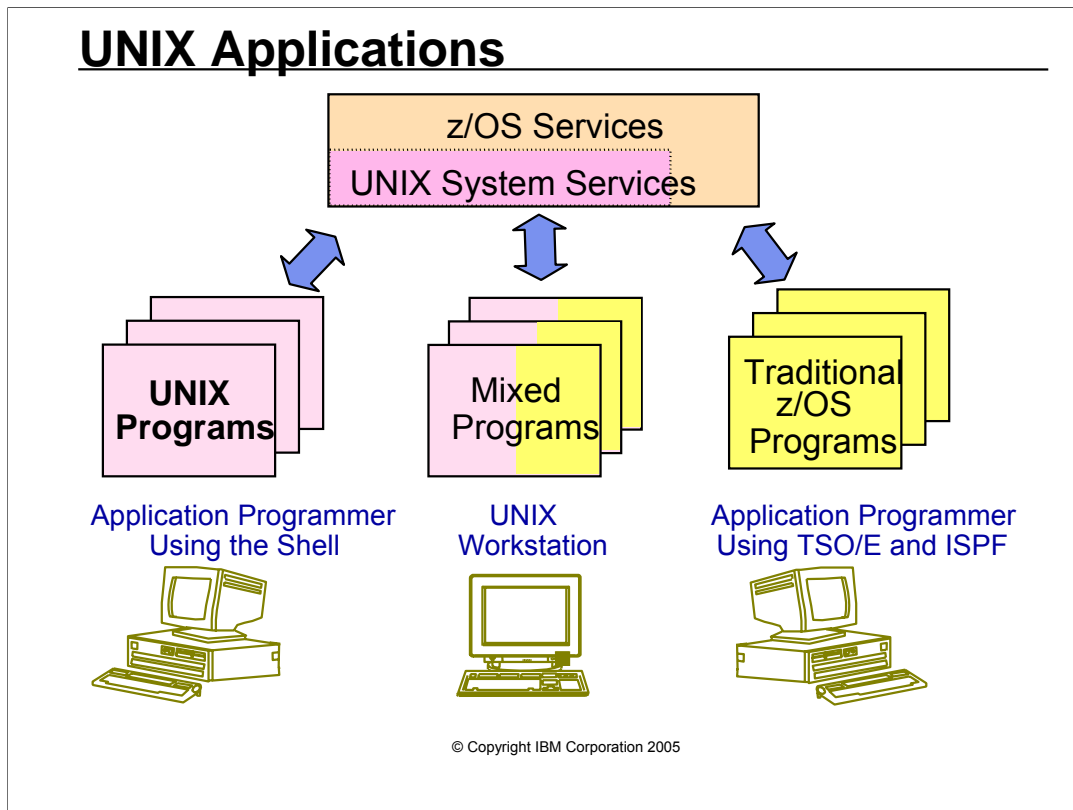
## Unit objectives

---

After completing this unit, you should be able to understand:

- Why we have Unix System Services (USS)
- Different ways to connect to USS
- ISHELL interface
- OMVS interface
- Direct logon to USS



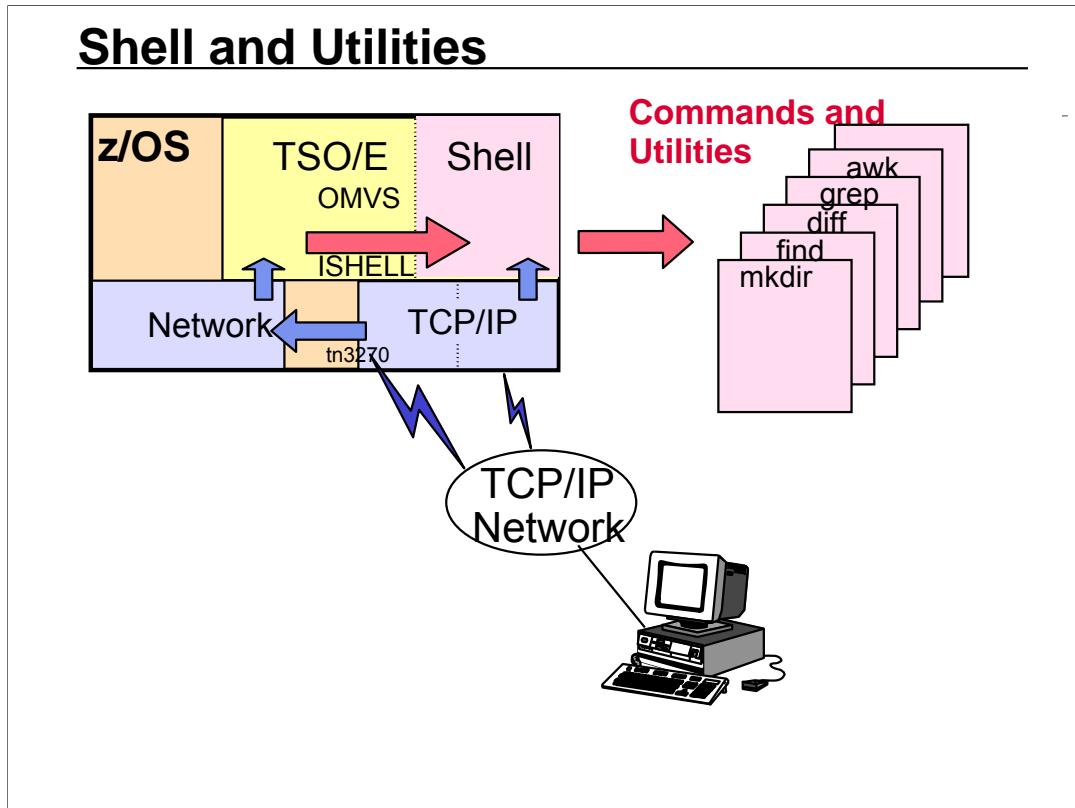


z/OS provides traditional MVS programming services plus the UNIX System Services, the UNIX shell and utilities feature, the IBM Language Environment for z/OS and VM, and the C/C++ for z/OS Compiler.

The services that support UNIX application programming make it possible to develop and run C application programs that provide XPG4.2 defined function. The C program source can be developed on an z/OS system using UNIX System Services, or on a programmable workstation (PWS). Creation of an application executable file must be done using UNIX System Services through the shell.

UNIX applications that are XPG4.2 conforming can be ported to z/OS, and XPG4.2 applications on z/OS can be ported to any XPG4.2 supported platform.

Many of the XPG4.2 function calls are available as callable services which can be used in z/OS applications to exploit XPG4.2 functions. This provides the capability to access hierarchical files from an z/OS program. UNIX System Services programs can also invoke z/OS system services and access z/OS data sets. These types of applications are called mixed programs. None of these services affect existing z/OS applications.



Shows the different interfaces to get into USS.

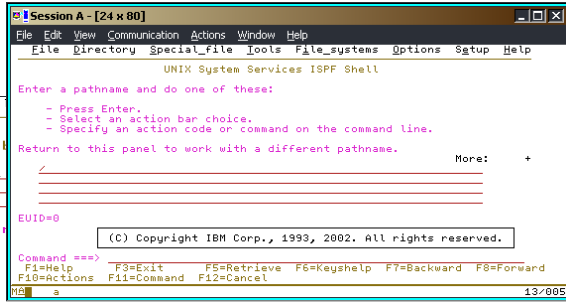
OMVS  
ISHELL  
tn3270

Examples are given over the next couple of pages.

# ISHELL and OMVS

Menu List Mode Functions Utili  
 ISPF  
 Enter TSO or Workstation commands  
 ==> ishell

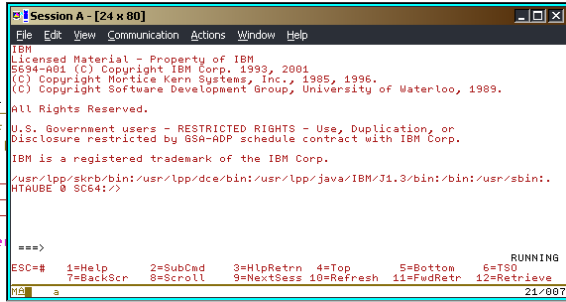
Place cursor on choice and press e



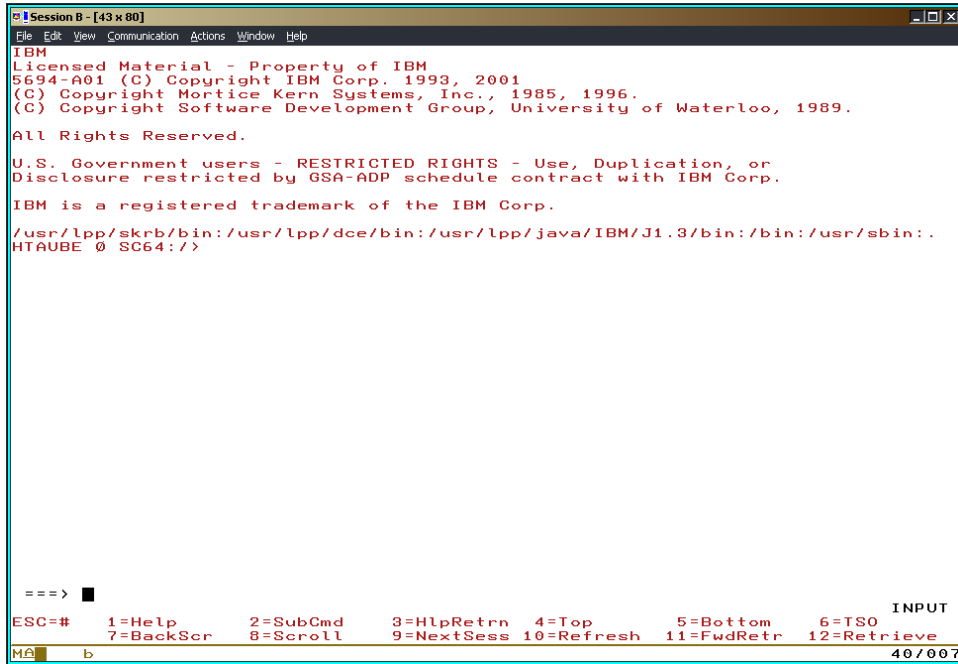
READY  
 OMVS \_

Menu List Mode Functions Utili  
 ISPF  
 Enter TSO or Workstation commands  
 ==> omvs

Place cursor on choice and press e



## A Shell Screen



```
Session B - [43 x 80]
File Edit View Communication Actions Window Help
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2001
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.
All Rights Reserved.
U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.
IBM is a registered trademark of the IBM Corp.
/usr/lpp/skrb/bin:/usr/lpp/dce/bin:/usr/lpp/java/IBM/J1.3/bin:/bin:/usr/sbin:.
HTAUBE @ SC64:/?

===> █
ESC=# 1=Help 2=SubCmd 3=HlpRetrn 4=Top 5=Bottom 6=TSO INPUT
7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MA b 40/007
```


## Asynchronous Terminal Access to Shell

```

$login OS390
$login -l jane OS390
enter password:

$telnet -t vt220 OS390
$telnet -t vt220 -p 623 9.24.104.126
Enter userid:
Enter password:

```



1. TCP/IP  
login or telnet port
2. INETD  
fork an address space
3. rlogind or otelnetd  
validate user and  
spawn new process
4. Shell

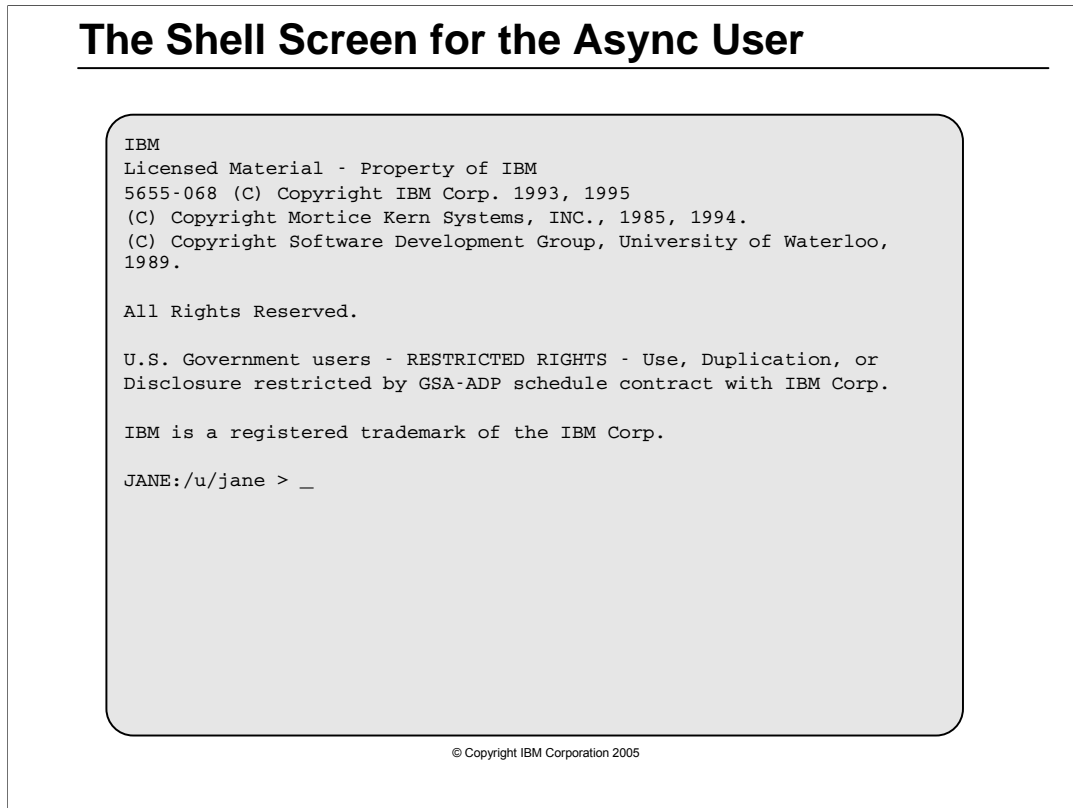
The rlogin or telnet commands used by the end user are in the syntax of the remote system. The picture shows only some examples.

The **rlogin** command names the z/OS system that you want to connect to and a user name if you are logging onto z/OS with a different user identity.

The **telnet** command may name the terminal type, the host name or IP address and the port number if different from the well-known ports used by TCP/IP daemons. In either case, z/OS asks the user for a password.

On the UNIX System Services system the INETD system daemon must be running to receive these requests. It creates an address space for each user which contains the appropriate server daemon (rlogind or otelnetd). The shell is started by this daemon after it validates the user identity. The daemon is also responsible for ASCII to EBCDIC conversion as data flows from the terminal to the host. Once the shell is started the user's profile or individual commands are used to set up the user's shell environment. A user can start multiple shells by issuing multiple rlogin or telnet commands.

It is also possible to use OpenSSH which is a non-chargeable optional product available from IBM.



This is the screen that is displayed when a user invokes the shell after logging on with rlogin or telnet.

The \$ prompt is an indication from the shell that it is ready to accept input from the command line. For a superuser the default prompt is #. A user can decide to use a different prompt than \$ as this example shows.

The shell works in line mode and everything typed on the command line is processed in line mode (canonical mode). This means the input is not processed until the user presses the Enter key.

When an application that requires character mode (or raw mode) support is invoked, the shell runs the application as defined. For example, the vi editor executes in character mode, and when you exit this editor the shell returns to line mode.

In this mode the ASCII control key on the terminal works as it should. For instance, to interrupt an application use <Ctrl-C> and to end the shell session use <Ctrl-D>.

## Time for Lab Exercise

---

### **Exercise 2** Access USS in various ways



© Copyright IBM Corporation 2005

### **Exercise 2**

From TSO access the Unix Services Shell (USS)

ISHELL

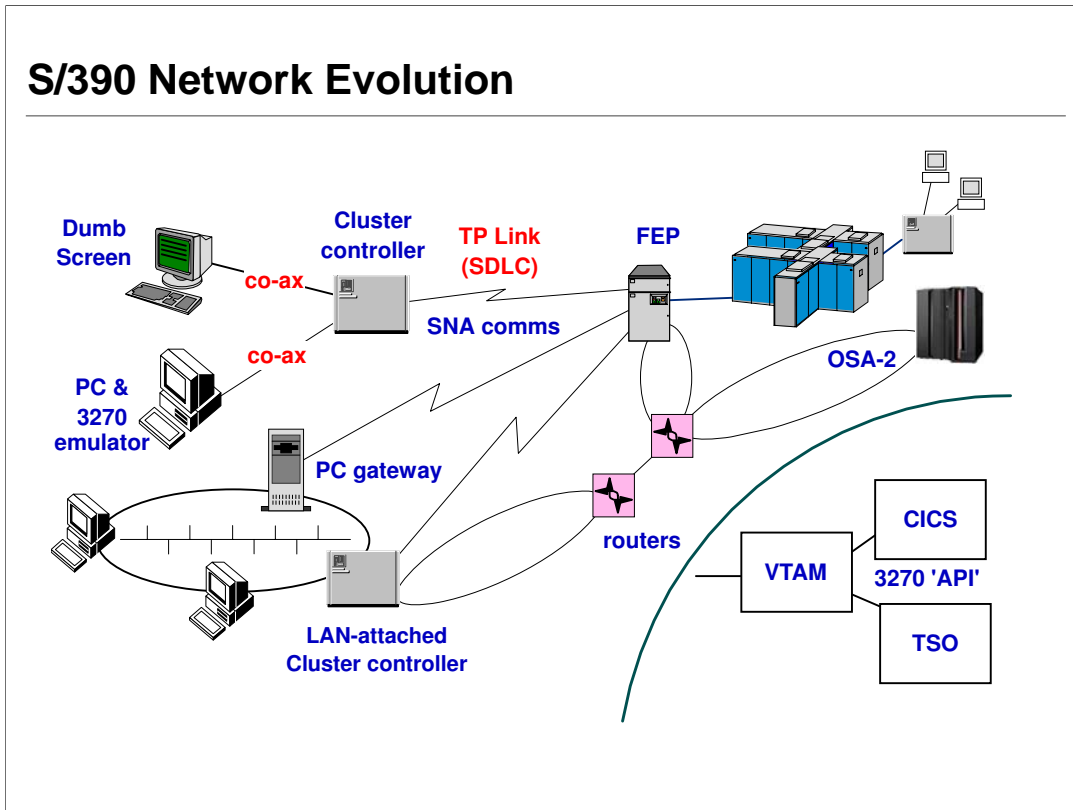
OMVS

Using PuTTY, telnet directly into USS



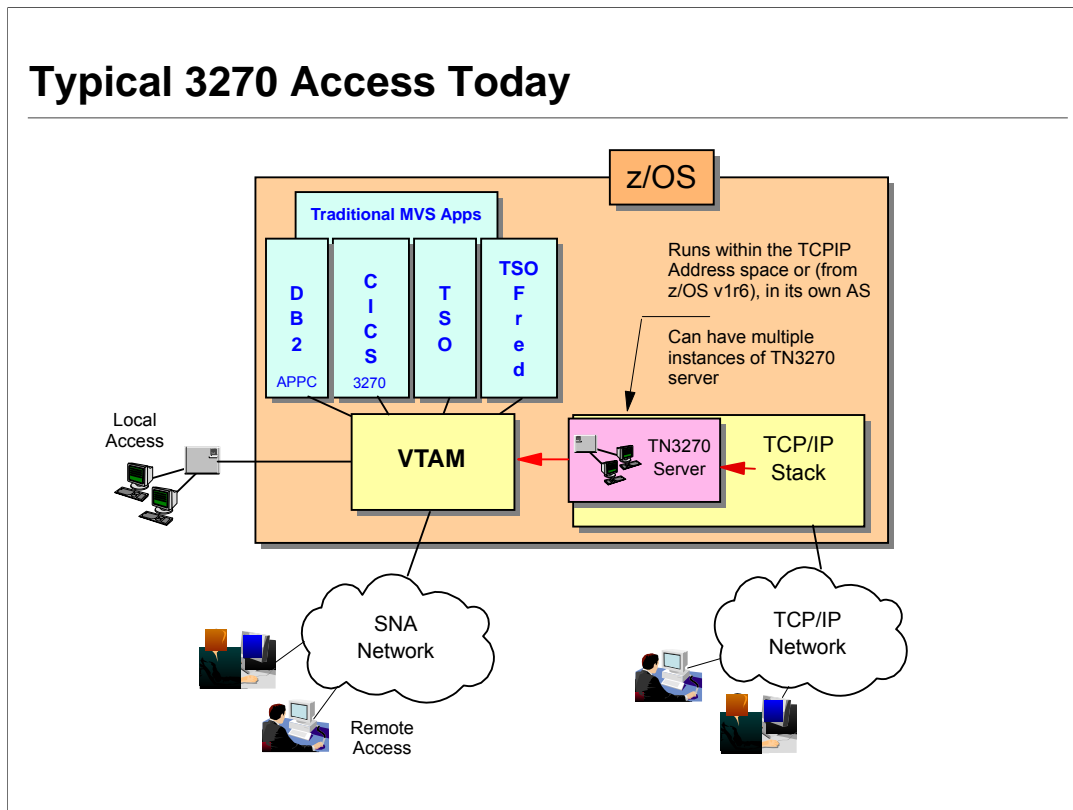






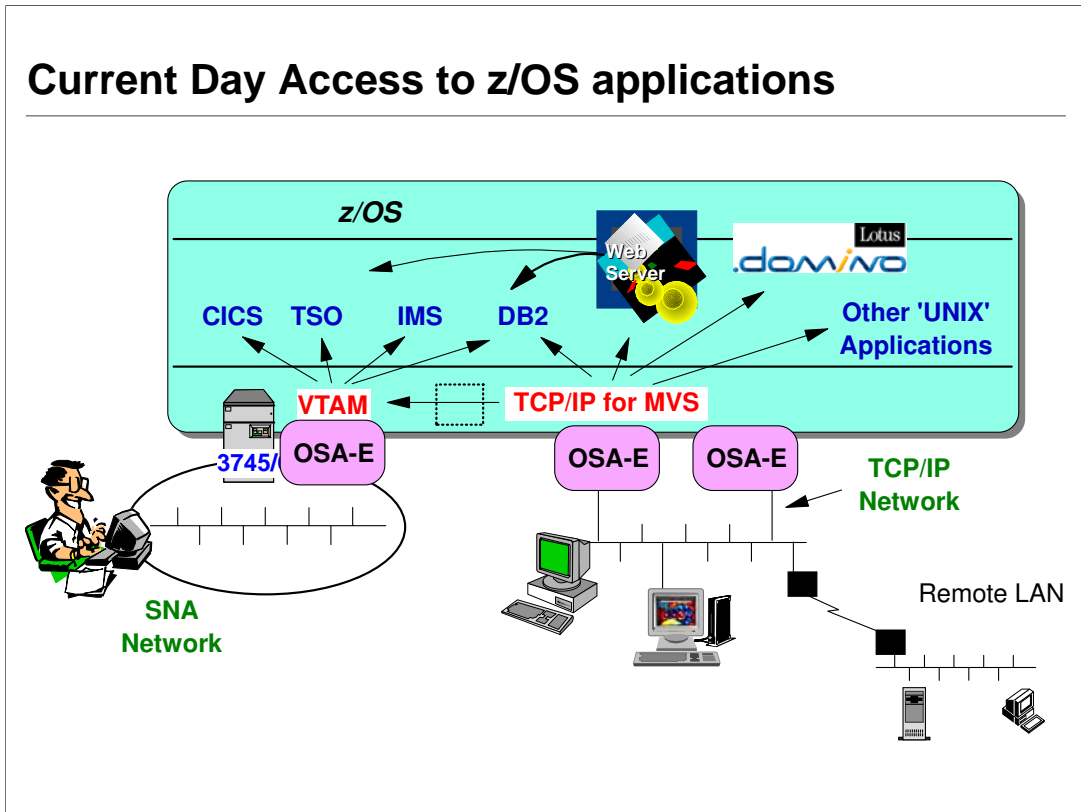
This foil shows the different S/390 network links from a historical perspective.

First there were co-ax attached dumb terminals, then co-ax attached PCs and PC LAN gateways. Later on there were LAN attached gateways and LAN links over router networks.



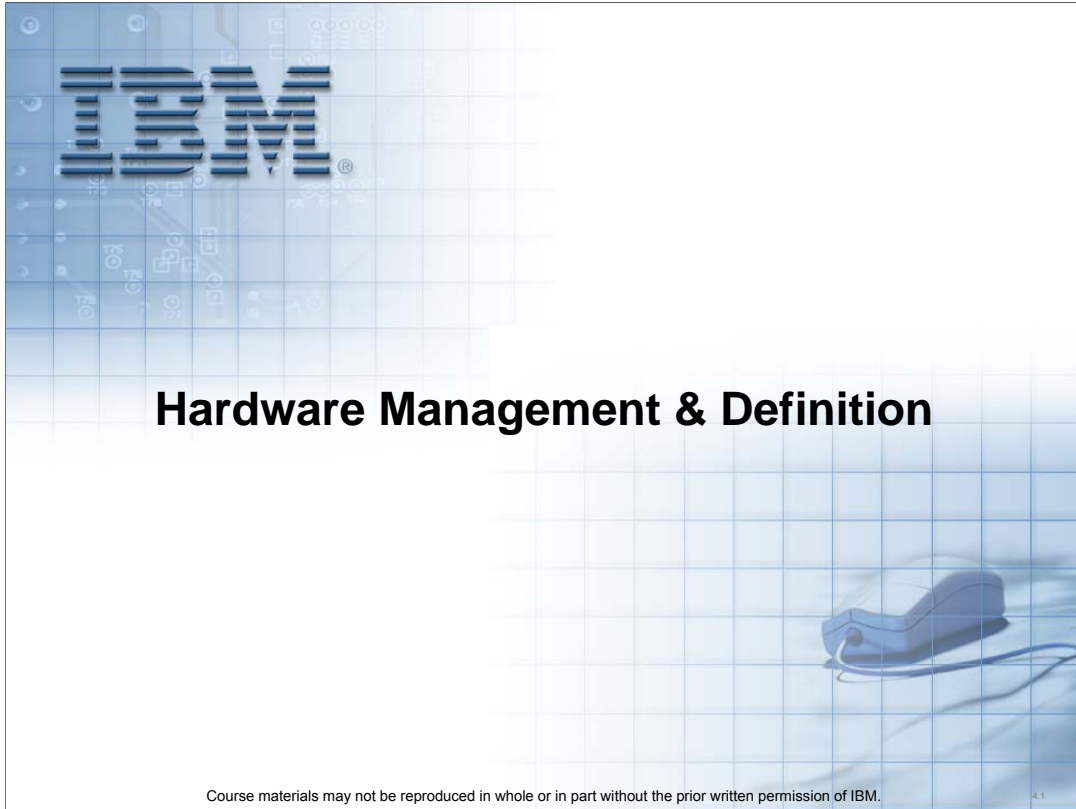
The TN3270 server is one of the most prominent TCP/IP applications to be found on a z/OS system. It consists of two basic parts. The first is a standard Telnet server, the second is a VTAM application. To VTAM, it appears as a collection of local dumb screens. The TN3270 server maps a TCP/IP telnet request to a VTAM LU, thereby allowing access to traditional SNA applications, such as TSO, NetView, CICS, etc;

Prior to z/OS v1r6, the server ran in the same address space as the TCP/IP stack. From z/OS v1r6 the server can now run in its own address space providing greater independence. In this way there can be multiple instances of the TN3270 server.



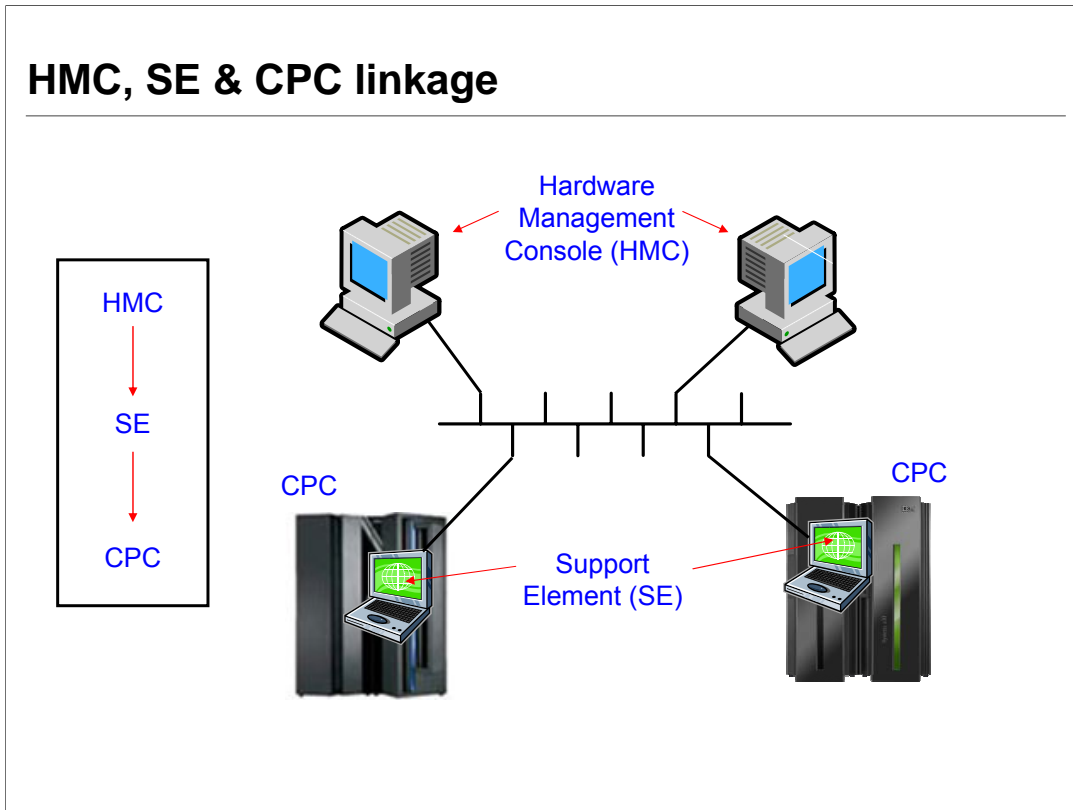
Today's System z systems support traditional SNA connectivity via Front End Processors (FEPS) or via OSA-Express .

TCP/IP on z/OS supports connectivity from IP networks to traditional SNA 3270 based applications as well as to TCP/IP based applications such as the z/OS Web Server, Lotus Domino server and FTP servers.



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

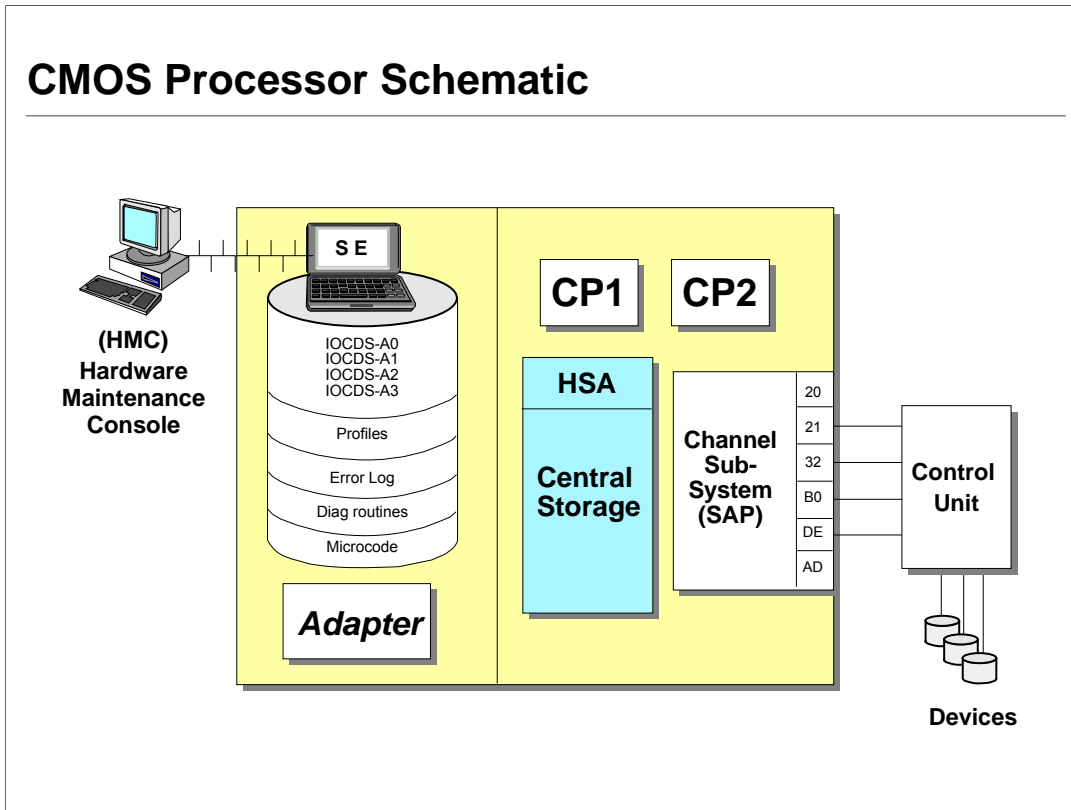
4.1



The Hardware Management Console (HMC) is used to manage, monitor, and operate all defined CPCs and images. The HMC is attached to a LAN; also attached to the same LAN is each CPC's Support Element (SE).

Multiple HMCs may exist. The recommendation is to have a minimum of two HMCs.

Note: The SE is behind the CPC covers.



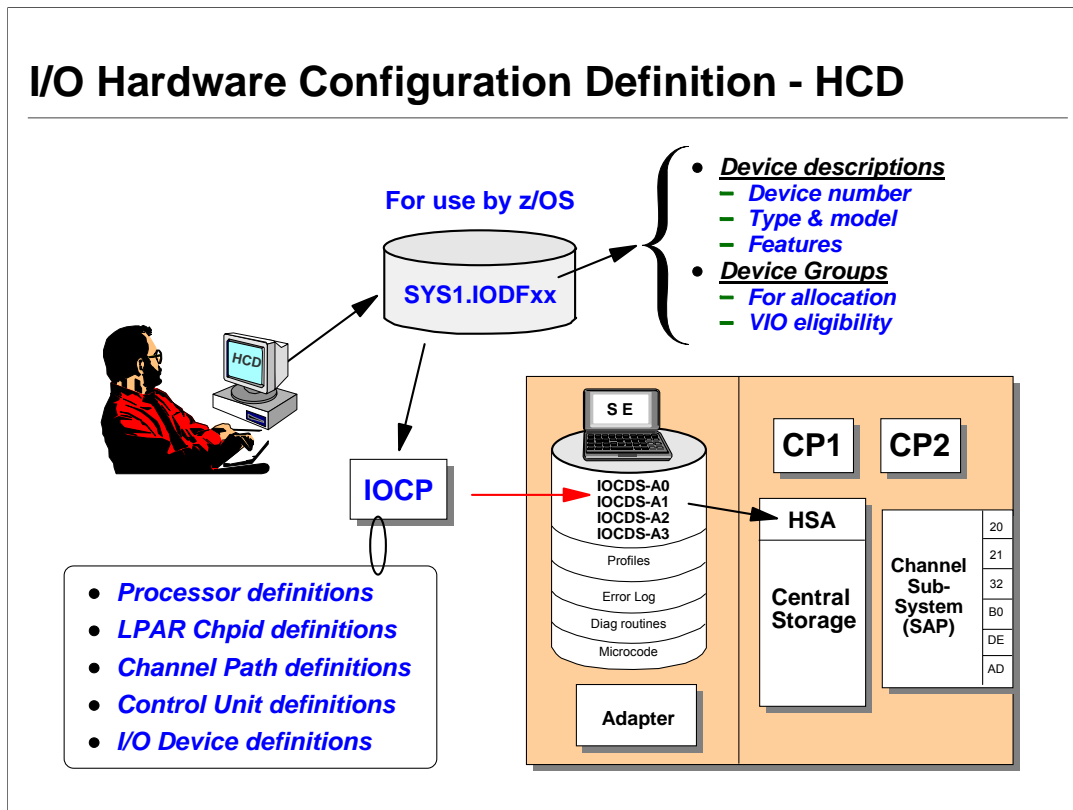
The Service Element together with a special adapter, makes up the 'process controller' function of each CPC

This function monitors and controls the CPC and provides a maintenance interface to the system. The SE holds a variety of information including those shown in the foil.

I/O control datasets (IOCDS), built by the HCD process, stores configuration details which are loaded into a part of storage (HSA) for use by the channel subsystem.

Profiles contain information relating to LPAR setup and IPL.

The HMC is used to configure system requirements, via profiles, as well as being used to power on, activate LPARs, and IPL.



Hardware Configuration Definition (HCD) is an ISPF application which is used to define I/O configuration to the System z hardware (channel subsystem) and the z/OS operating system.

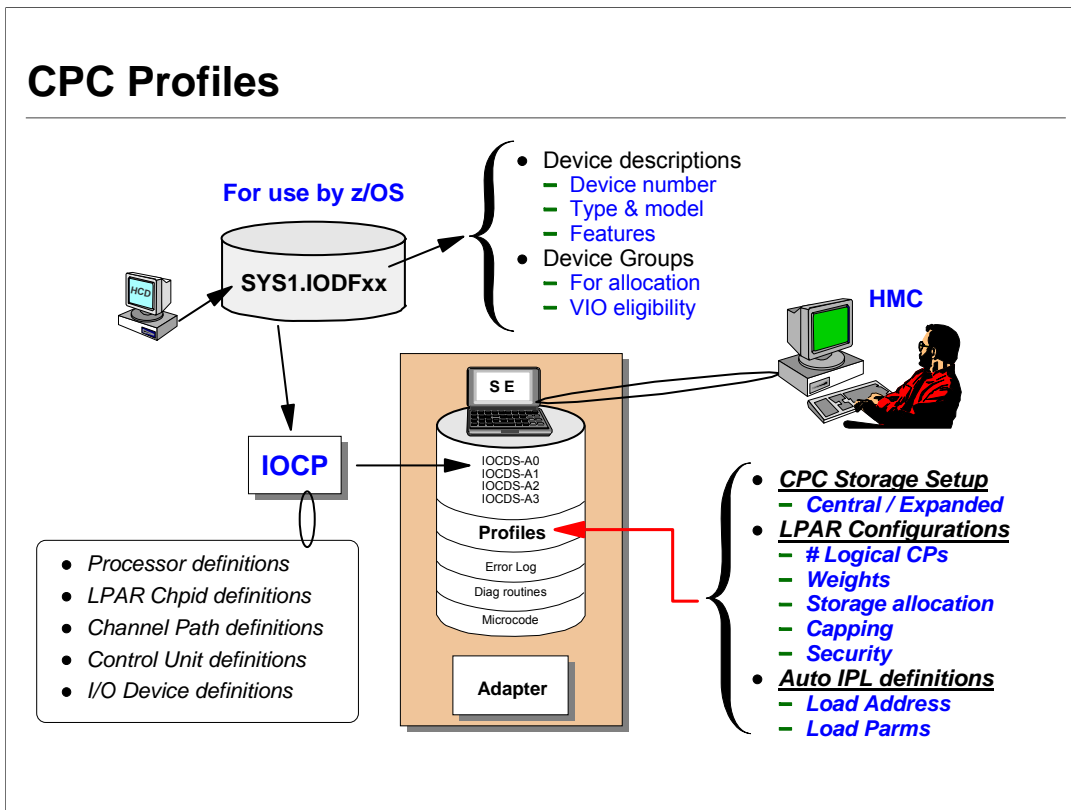
HCD generates an Input/Output Definition File (IODF) from details configured by the HCD user, typically a systems programmer.

z/OS accesses the IODF during the IPL process in order to build its own I/O configuration (control blocks).

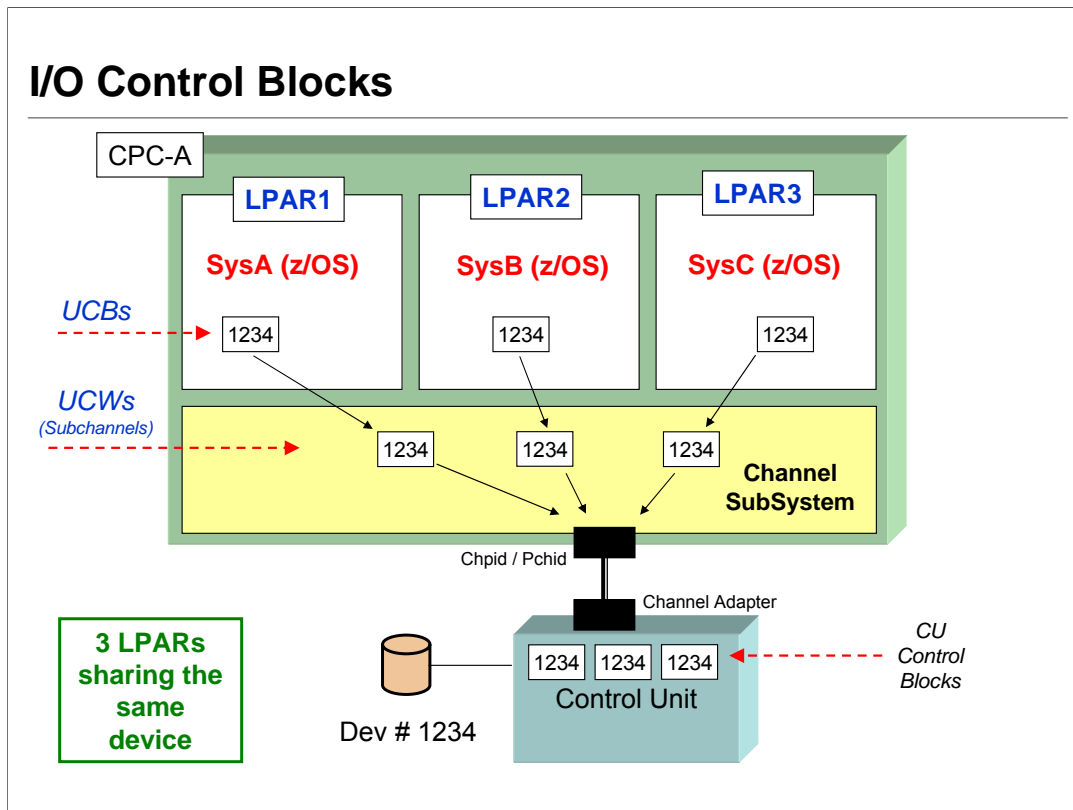
An additional part of the HCD process uses data from the IODF to create the I/O Configuration Data Set (IOCDs) for use by the Channel Subsystem. (HCD invokes the I/O Configuration Program - IOCP, to perform this function).

At Power-on-reset (POR) the appropriate IOCDs is loaded into the Hardware Systems Area (HSA) where it is used by the channel subsystem. The HSA is a portion of central storage and varies in size depending on the number of configured devices, chpids, and so on.





To complete the configuration of the CPC, a system program will use the HMC to define 'profiles' that further describe the LPARs defined in HCD, and hence in the IOCDs.

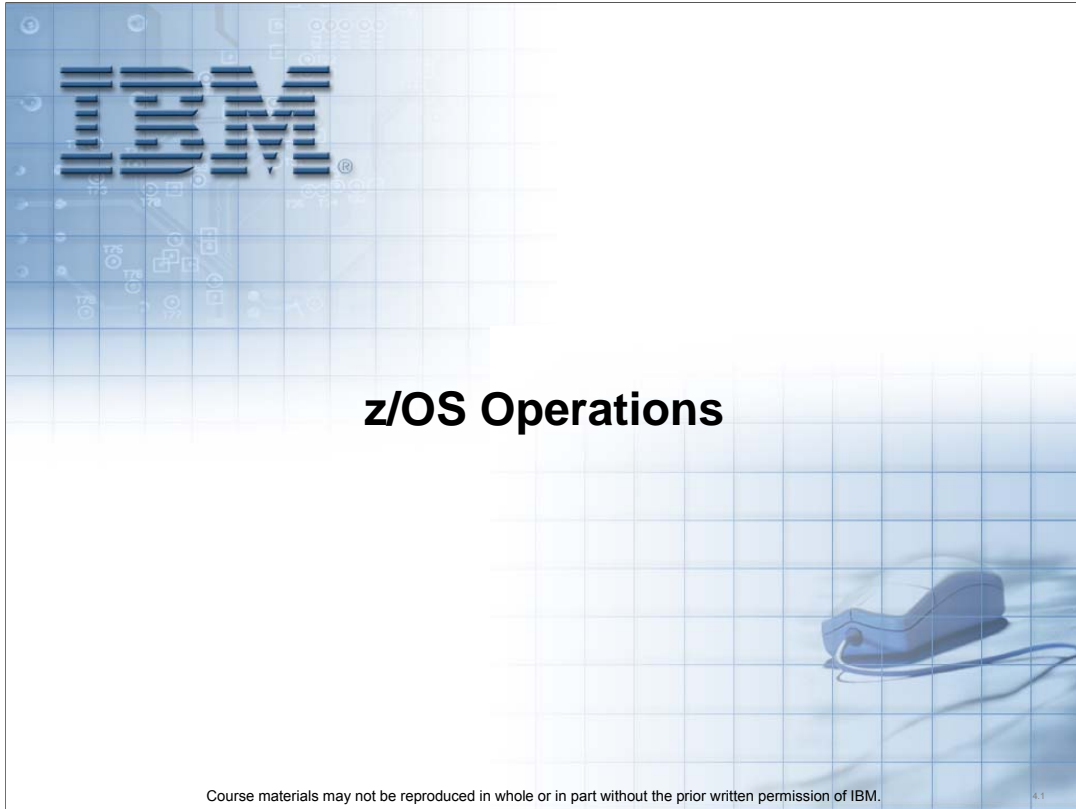


Each part of the system needs to 'manage' access to I/O.

In the example configuration above, device 1234 needs to be known and tracked by each z/OS. z/OS uses an area of storage (RAM) called a UCB (Unit Control Block) for this purpose. z/OS will have a UCB for every device it is configured to use.

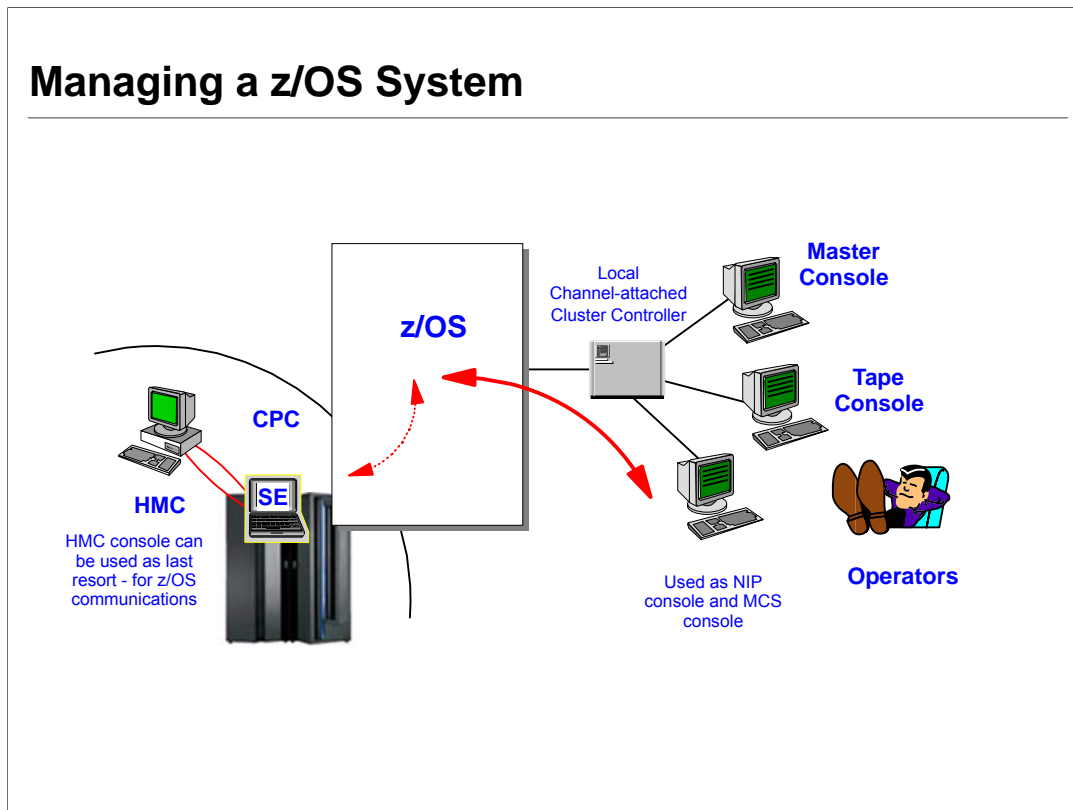
The channel subsystem manages each device, per LPAR, in a UCW (Unit Control Word - which is mapped to a subchannel).

Likewise every control unit needs a 'control block' to manage each device, per LPAR.



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.1



We have already looked at the Hardware Maintenance Console (HMC) which is used to power on, activate and IPL the z/OS system. It is not normally used to communicate with the z/OS system but can be used in this mode as a last resort.

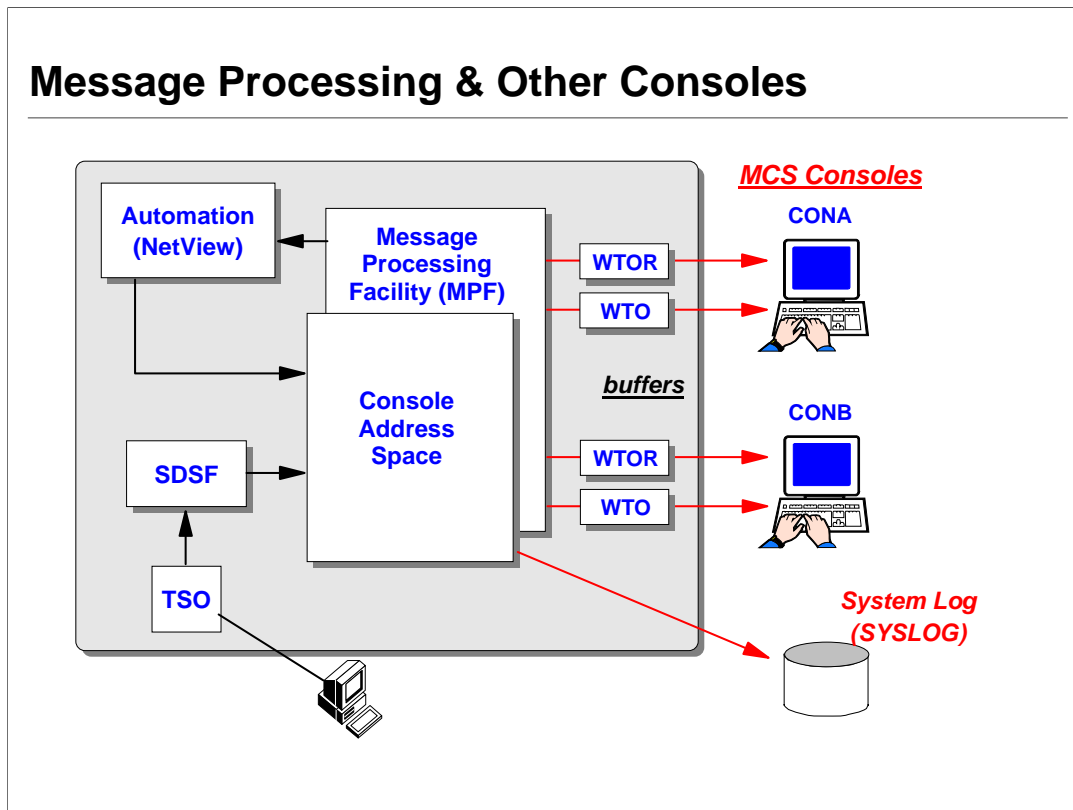
Local channel attached terminals are used for z/OS operational communication. These terminals are used during the Nucleus Initialisation program (NIP) phase and for normal console operations.

Routing codes and message levels can control message flow. They are specified in the CONSOLxx member of PARMLIB (normally SYS1.PARMLIB). They are used to specify which sort of messages should go to each console, e.g. only tape messages to the tape librarian's console.

routing code 1,2,3,9,10 for OS/390 master console

routing code 3 and 5 for tape control

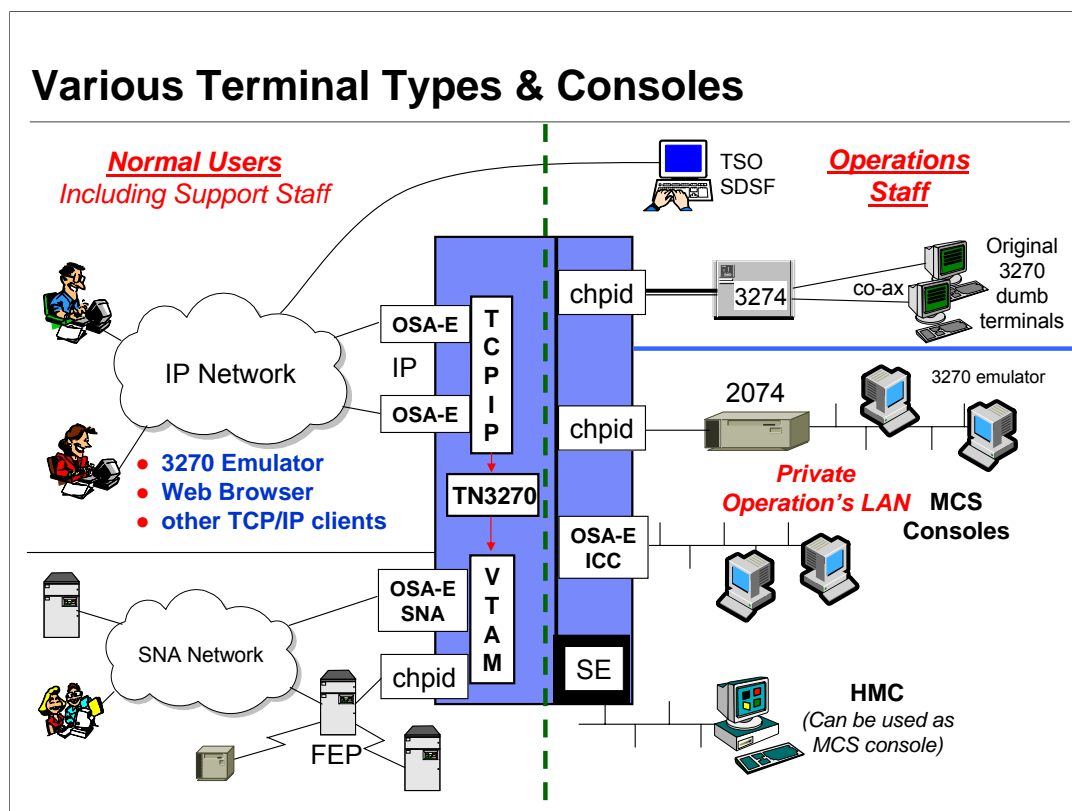
routing code 7 for printers



With MPF (Message Processing Facility), a standard feature of z/OS, incoming messages can be filtered so that only important ones reach operations consoles. In some z/OS installations up to 70% of messages may be suppressed. Automation products such as NetView can also be used to suppress and respond to specific messages. Normally all messages are written to the system log.

Each MCS console has a specific name. Each has its own WTO (write to operator -info) and WTOR (write to operator - response required) buffers. If there is a problem with a console or it has been left in a specific mode, these buffers can fill up and cause problems.

SDSF is a facility that allows TSO users to interactively monitor and control jobs, queues and system resources. It also allows TSO users, with the appropriate RACF authority, to issue operator commands.



It can be quite confusing working out what type of terminal is required for specific functions and how they need to connect into the CPC and/or z/OS system.

Most general users, including system programmers and other support staff, will connect in over a TCP/IP network. Some customers may still have SNA networks, as shown, so may use this type of connectivity. However, such 'external' SNA networks are rapidly disappearing.

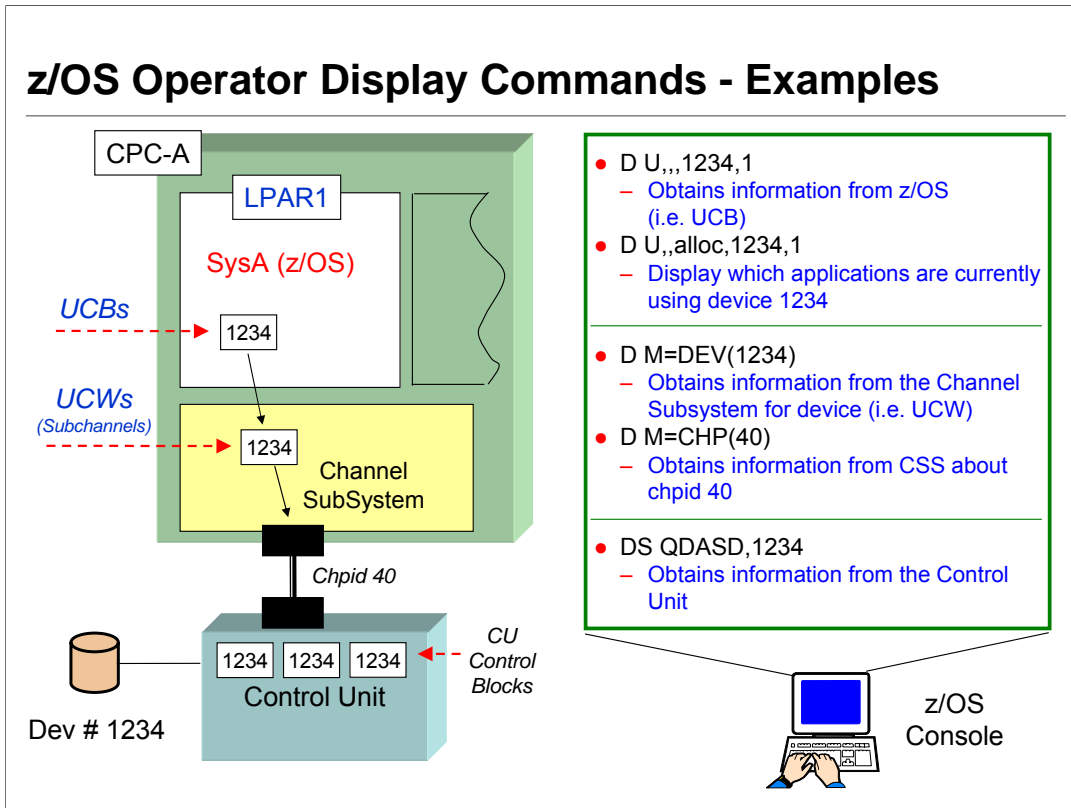
System programmers and other support staff will need access to TSO. TSO is an SNA application and therefore needs to be accessed via VTAM. As the diagram shows, access to VTAM from a TCP/IP network can be accomplished by way of a TN3270 server as described earlier.

Operator consoles need to be 'directly' connected to the system. They cannot be dependent on networking software such as TCP/IP or SNA. As shown above, the original operator consoles were dumb screens connected by co-ax cables to a cluster controller (3274, 3174) which was channel attached to the CPC. During the initial IPL process these consoles act as NIP consoles (Nucleus Initialisation Process consoles). Once this phase has completed the consoles then function as MCS (Master Console System) consoles.

The 3174 cluster controller and 3270 dumb screen function was superseded by the IBM 2074. The 2074 emulated the operation of the 3174 controller and provided TCP/IP connectivity to PCs running 3270 emulators. The 2074 is still 'channel connected' so from the perspective of the z/OS system is 'directly' connected. That is, there is no dependence on TCP/IP or VTAM running on z/OS.

An alternative approach is to configure a port on an appropriate OSA Express card as an ICC (Integrated Cluster Controller). In this case the OSA port functions as an *inboard* 2074.

Once the system has been IPLed and the network is fully functional, operators may make use of TSO connected terminal functions.



For a full description of z/OS commands refer to the IBM publication

MVS System Commands – SA22-7627

The syntax of the `==> D U` command may seem somewhat confusing. The options for this command are as follows

`D U,<device type>,<device status>,<device number>,<number of devices>`

If I'm not concerned about the device type or device status then I leave those fields blank, which is how we end up with the first command string shown above.

In most cases, if an entry field is not specified then default values are used.

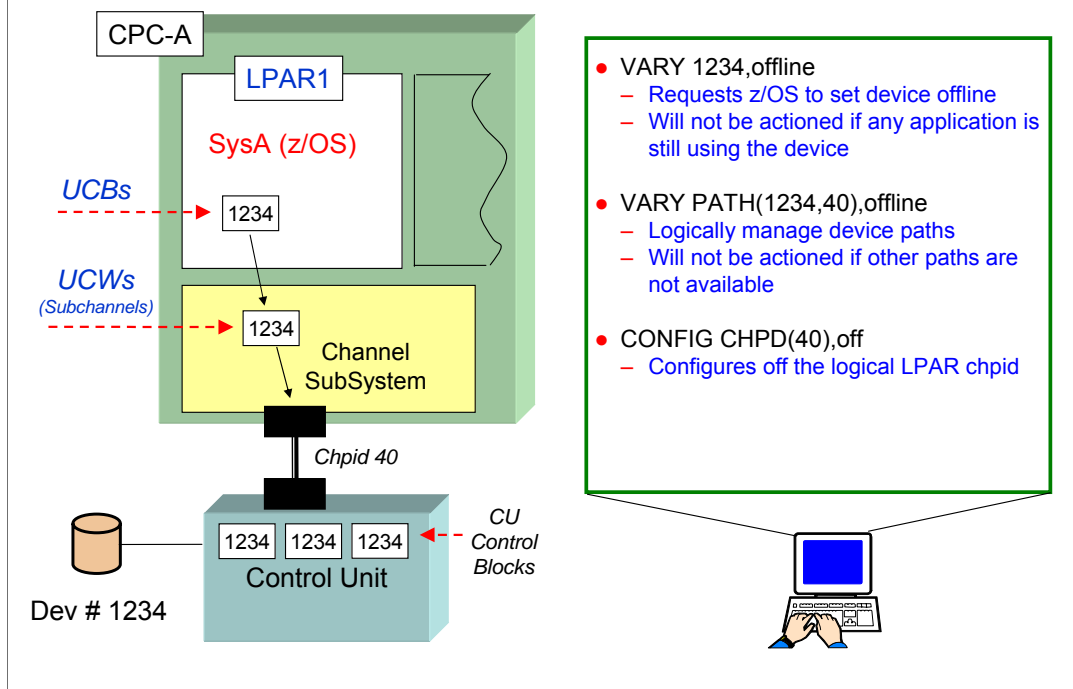
For example

<device type>	default =	all device types
<device status>	default =	any device status
<device number>	default =	the first configured device number
<no. of devices>	default =	16

So the command `==> D U,DASD,ONLINE`

would show the first 16 dasd (disk) devices that were 'online'

## z/OS Operator Commands - 2





## **z/OS Display Command Examples - Summary**

---

- **D U,DASD,ONLINE** – Show the first 16 disk devices online
- **D U,DASD,ONLINE,,64** – Show the first 64 disk devices online
- **D U,TAPE** – Display details of first 16 tape/cartridge devices
- **D U,,,1234,1** – Display details of the single device number 1234
- **D U,,,1234,4** – Display details of 4 devices starting from dev 1234
- **D U,,ALLOC,1234,1** – Display which jobs/apps are using device 1234
- **D M=DEV(1234)** – Display more detailed information about device 1234
- **D M=CHP** – Show information about all chpids
- **D M=CHP(40)** – Show information only about chpid number 40
- **D M=STOR** – Display details of central storage (RAM)
- **D M=CPU** – Display details of processors (engines) installed
- **D M=HSA** – Display how much HSA is in this LPAR
- **D M** – Display all configuration information
- **DS QDASD,1234** – Interrogate disk device number 1234
- **DS QTAPE,9876** – Interrogate cartridge device number 9876
- **D IOS,CONFIG** – Display HCD configuration information
- **D IPLINFO** – Display IPL information
- **D XCF,yyy** – Display Sysplex items .. many different options

## **z/OS Alter Command Examples - Summary**

---

- **Vary 100,OFFline**
  - **V 100-1FF,OFF**
  - **V 100,OFF,FORCE**
  - **V PATH(1234,40),OFF**
  - **CONFIG CHP(40),OFF**
  - **CONFIG CHP(40),OFF,UNCOND**
  - **CONFIG CHP(40),OFF,FORCE**
  - **Start xxx**
  - **stoP xxx**
  - **modiEy xxx**
  - **Cancel xxx**
  - **Force xxx**
  - **SET xxx**
  - **SETyyy zzz**
  - **SETXCF xxx,yyy**
- Vary device 100 offline, if inactive
  - Vary devices 100 to 1FF offline, if inactive
  - Vary off device 100, even if active
  - Vary off logical path 40 to device 1234, if not last
  - Configure chpid 40 offline to this LPAR
  - Config, even if normal device type is still active
  - Config off force, for all except master console
  - Start 'server' type application, e.g. S CICS
  - Shutdown appl (address space), e.g. P TCPIP
  - Send command to address space (if supported)
  - Cancel job/task if Stop doesn't work
  - Force job/task .. Use with utmost caution
  - Change different system settings dynamically
  - Similar to above, but for different functions/apps
  - Modify various items connected with Sysplex

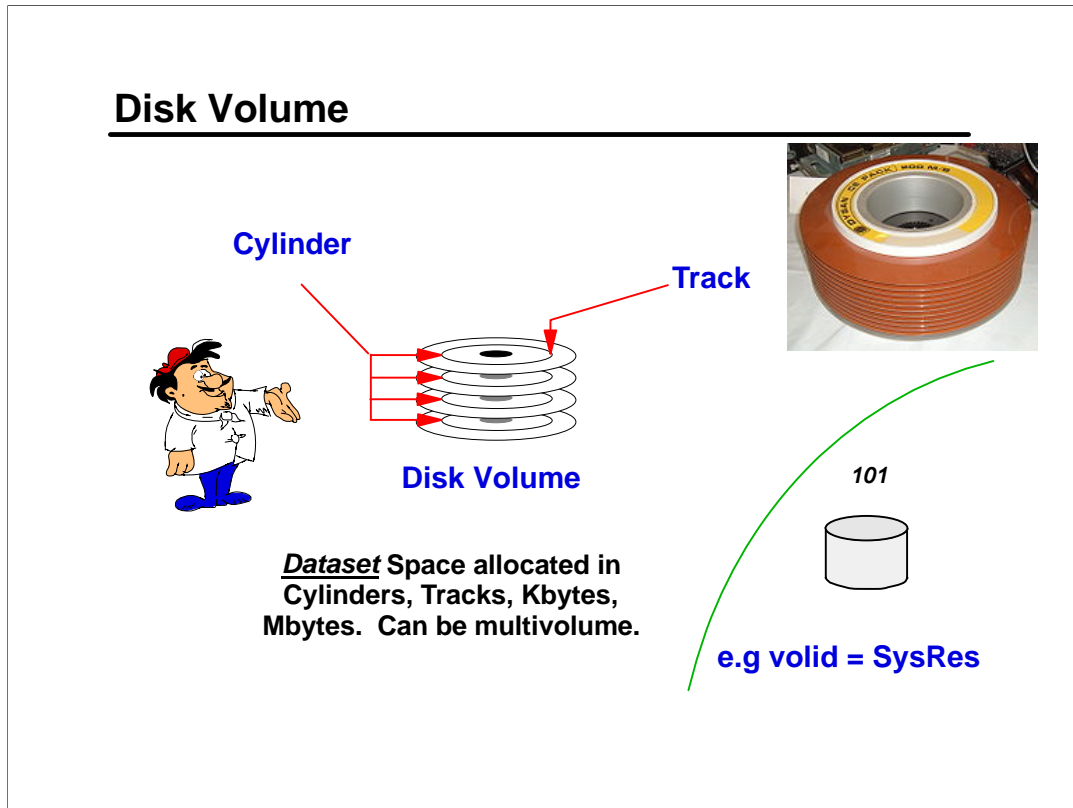


## Objectives - Datasets

---



- Describe different types of data sets and data set organisations
- Define the terms such as data, record, data set, block and VTOC
- Describe z/OS Catalogs
- Introduce the DFSMS family of products



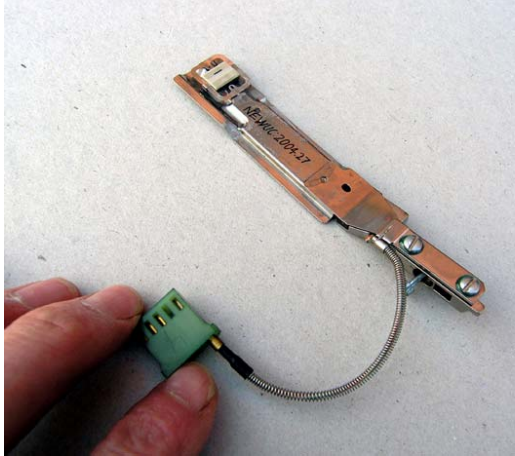
Traditional System z disk systems contained many disk drives, each with its own read/write multi-head mechanism called an actuator. The disk was made up from 8 platters. The actuator contained a read/write head on the topside and a read/write head on the underside for each platter, giving a total of 16 head/disk surfaces. One of these heads was used by the device itself leaving 15 heads for system use. Each platter is divided up into multiple tracks on which data can be written.

The combination of the same numbered track on each platter is called a cylinder. To locate a particular track the actuator 'seeks' to a specific cylinder, then switches in the appropriate head. Consequently a specific data track is known by its 'cylinder' and 'head' position (CCCHH).

The inset picture is of a 3330 demountable disk pack without its protective cover.

## Demountable 3330 Disk Drive

Single 3330 read/write head



3330 disk pack with cover

## Logical and Physical Records

### *Physical & Logical Record*



### *Physical Record*



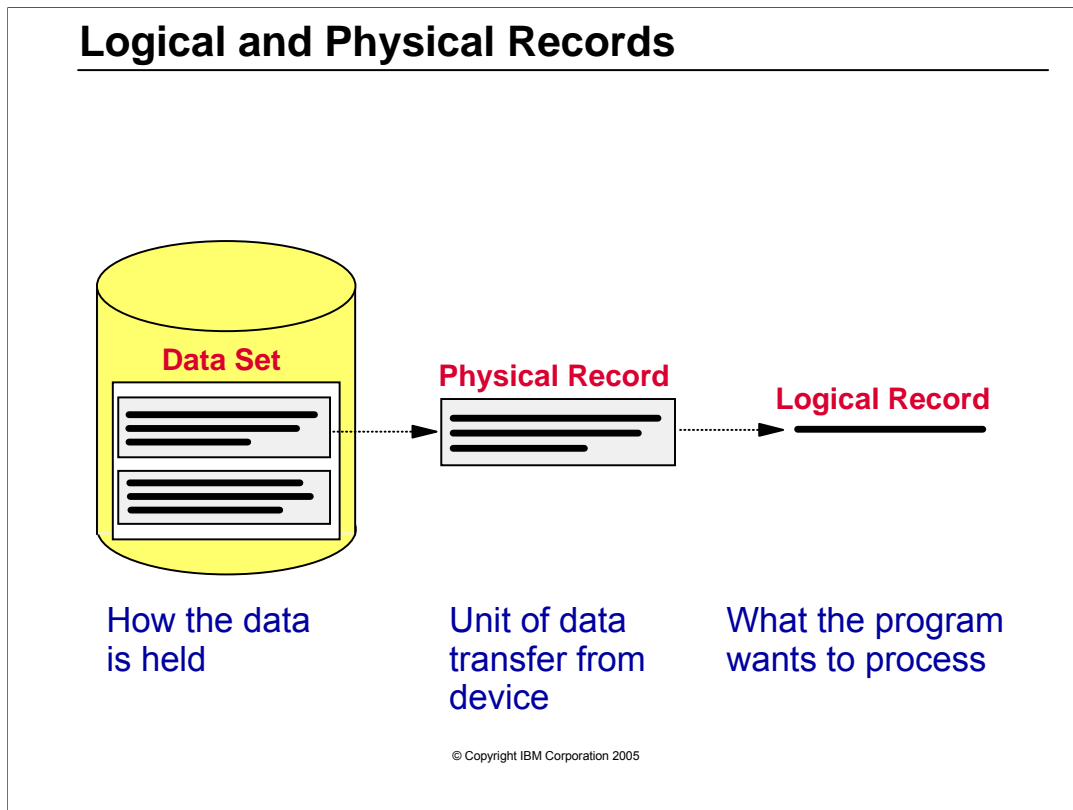
There are two distinct types of records.

A logical record.

One line of text/data the program expects to process.

A physical record.

The amount of data transferred in an I/O operation.



Data sets in z/OS do not only differ in their content, name, length, and data set type, but also vary internally in the length and format of their records.

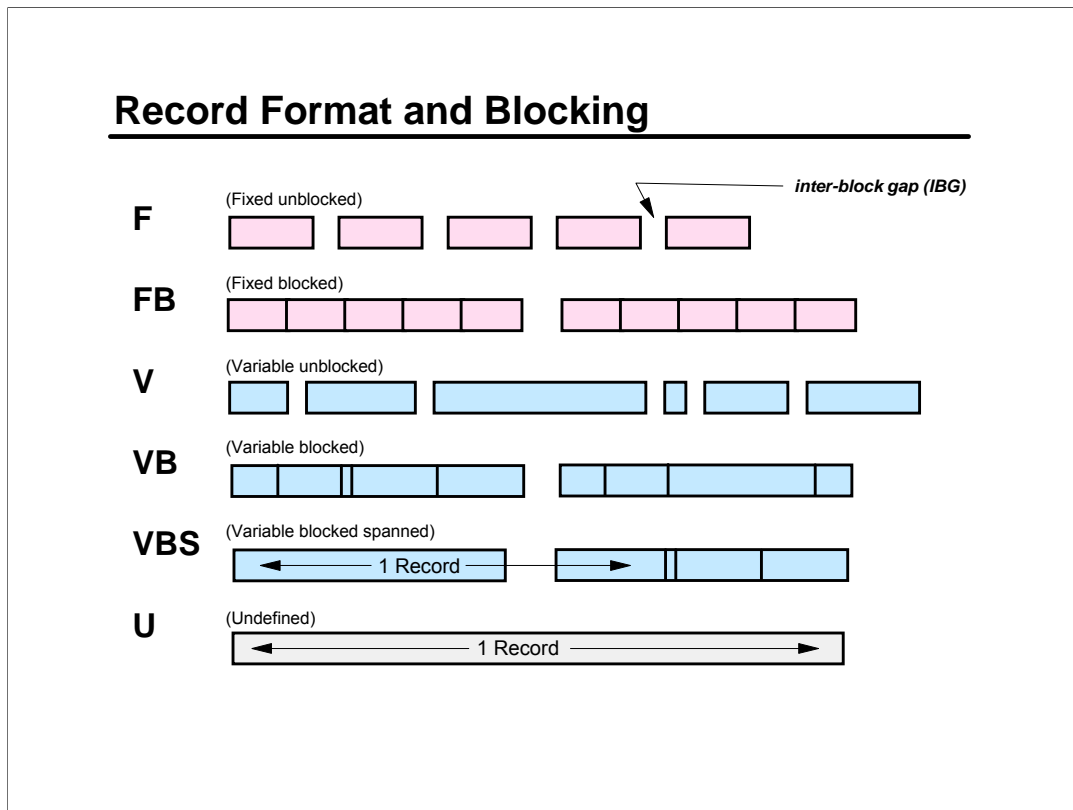
z/OS data sets may have both **logical records** and **physical records**.

A **logical** record is the unit of data that the program wants to process. It usually consists of a number of fields concatenated together.

A **physical** record describes the amount of data transferred in an I/O operation.

Combining multiple logical records in a single physical record (called blocking) has certain advantages over storing and retrieving a logical record as a physical record. These include a more efficient space usage as well as an improved performance when transferring data





Upon allocation of a new data set, users have to specify a record format. The most commonly used record formats are the **fixed-length record format** and the **variable-length record format**.

A fixed-length record format tells the system that all records within the data set are of the same length, that is, they all require the same amount of space, whether they contain only a single character or many characters.

The variable-length record consists of records that vary in length according to their content. This means records that hold little data require less storage than those containing a lot of data. The actual length of a record is specified within it as part of the record.

Both fixed as well as variable format records are commonly grouped in blocks, thus saving space and requiring fewer I/O operations than unblocked record storage. **Blocking** is the process of storing multiple data set records in a single block. Whenever an I/O operation takes place, an entire block rather than a single record is transferred.

## MVS Data Set Types

Emp#	Name	Dept	Salary
------	------	------	--------

A Logical Record: The unit of data that the program will process

Record 1	Record 2	Record 3	Record 4	Record ...	Record n	EOF
----------	----------	----------	----------	------------	----------	-----

### A Sequential Data Set:

- Records in the sequence they were entered, with an End of File marker at the end
- Records can't be deleted, and can only be inserted at the end
- Records must be read in sequence
  - ▶ For example, to read record 3, records 1 and 2 must be read first

© Copyright IBM Corporation 2005

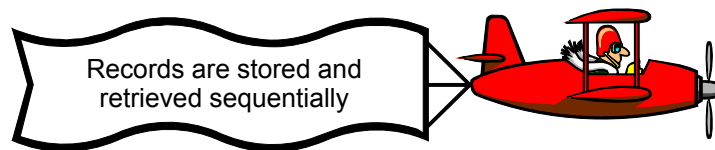
### Sequential data set

In a sequential data set, also referred to as **physical sequential (PS)**, records are arranged sequentially in the order in which they are written. New records are appended at the end of the data set. In sequential data sets, data is organized for sequential access. To retrieve the fifth record of the data set, the system first has to read the preceding four. Sequential data sets can be stored on DASD or on tape. They are required for the use of magnetic tape devices or printed output.

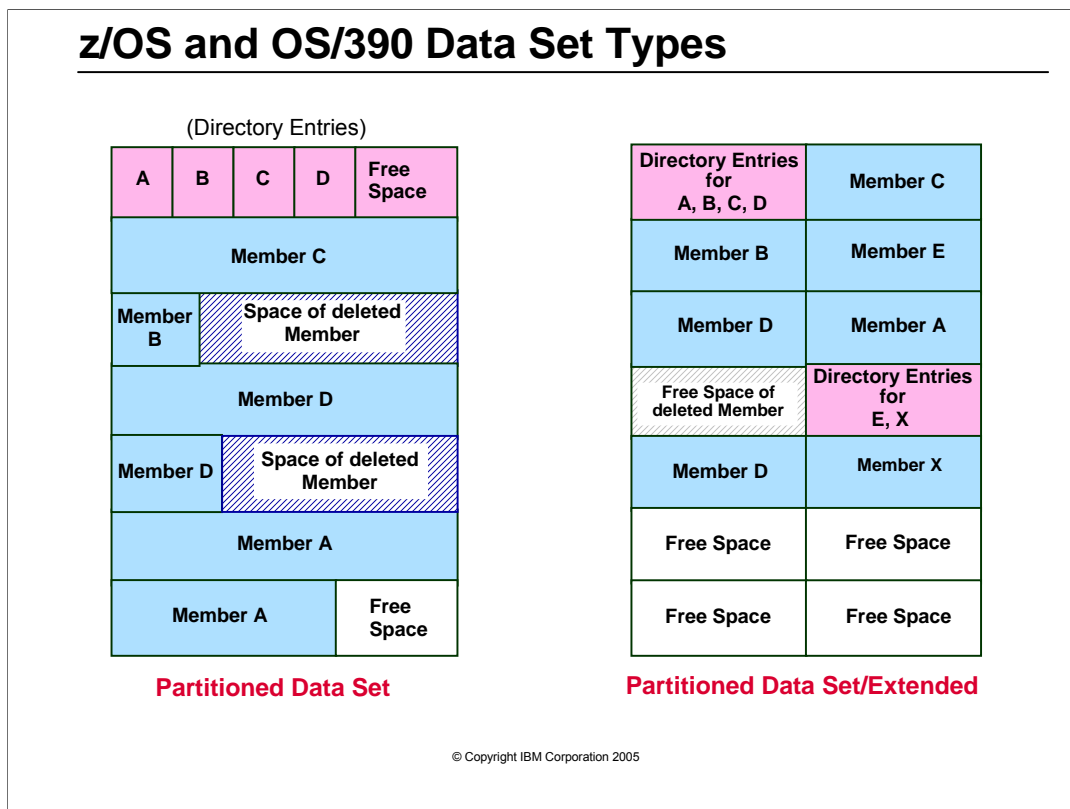
## Sequential Datasets

### ***Dataset xx.yy.zz***

Record 1	Record 2	Record 3	Record 4
Record 5	Record 6	Record x	free space



The records of sequentially organized datasets are stored and retrieved sequentially, i.e. when reading them, programs start from the beginning and get access to records in the order they were written and when a new record is written, it is appended at the end of the dataset. Datasets residing on tapes are sequential by physical necessity. Datasets on disk may, but don't have to be, sequential.



### Partitioned data set

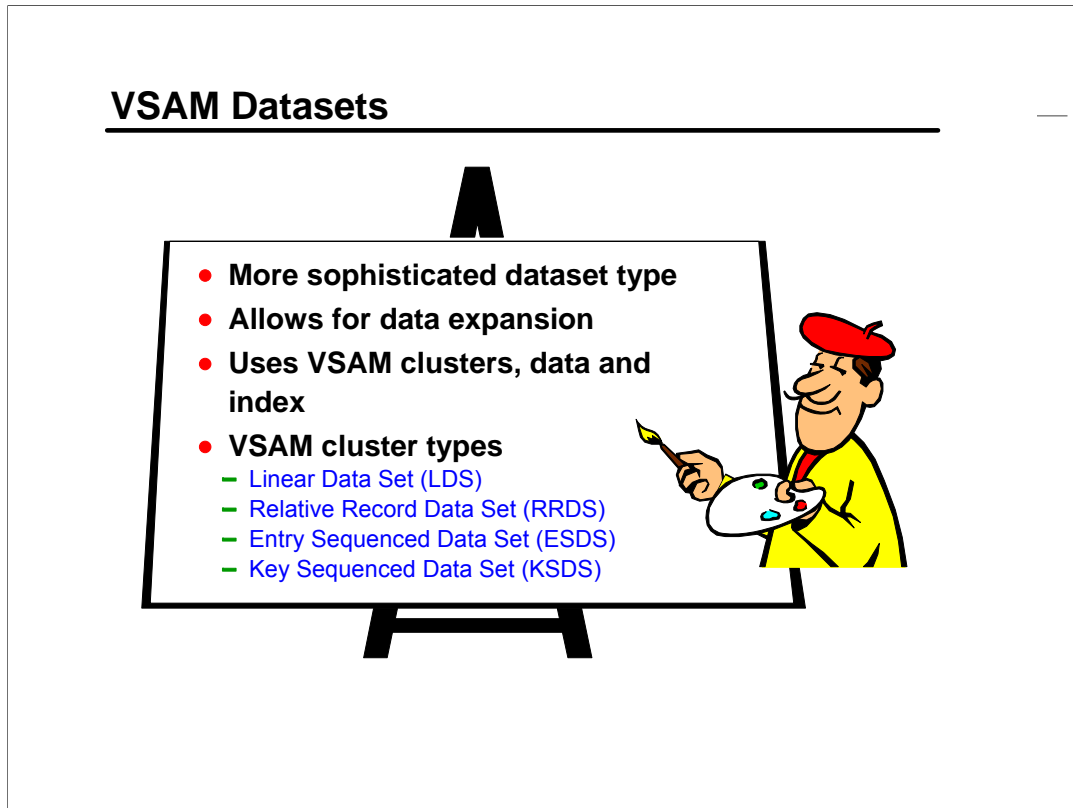
A partitioned data set (PDS) is divided into sequentially organized **members**, each of which can have one or more records. Each member has a unique name, stored in a **directory** that is part of the data set. The directory is located at the beginning of the data set and contains an entry for each member. Each directory entry contains the member name and the starting location of the member within the data set. The directory entries are arranged by name in alphanumeric collating sequence. The main advantage of using a partitioned data set is that, without searching the entire data set, you can retrieve any individual member after the data set is opened.

Individual members can be added or deleted as required. A deleted member is removed from the directory. Its space cannot be reused until the data set is reorganized using a system utility.

### Partitioned data set extended

In appearance, a Partitioned Data Set Extended (PDSE) is very similar to a partitioned data set. For accessing a partitioned data set directory or member, most PDSE interfaces are indistinguishable from PDS data set interfaces. However, PDSEs have a different internal format, which gives them increased usability.

The main advantage of using a PDSE over a PDS is that PDSEs use DASD space much more efficiently. The size of a PDS directory is fixed regardless of the number of members in it, while the size of a PDSE directory is flexible and expands to fit the members stored in it. Also, the system reclaims space automatically whenever a member is deleted or replaced, and returns it to the pool of space available for allocation to other members of the same PDSE. The space can be reused without having to reorganize the data set.



As the foil shows, VSAM datasets are a far more sophisticated dataset type and a full coverage of them is beyond the scope of this course. However a brief description of some of the VSAM dataset types follows:-

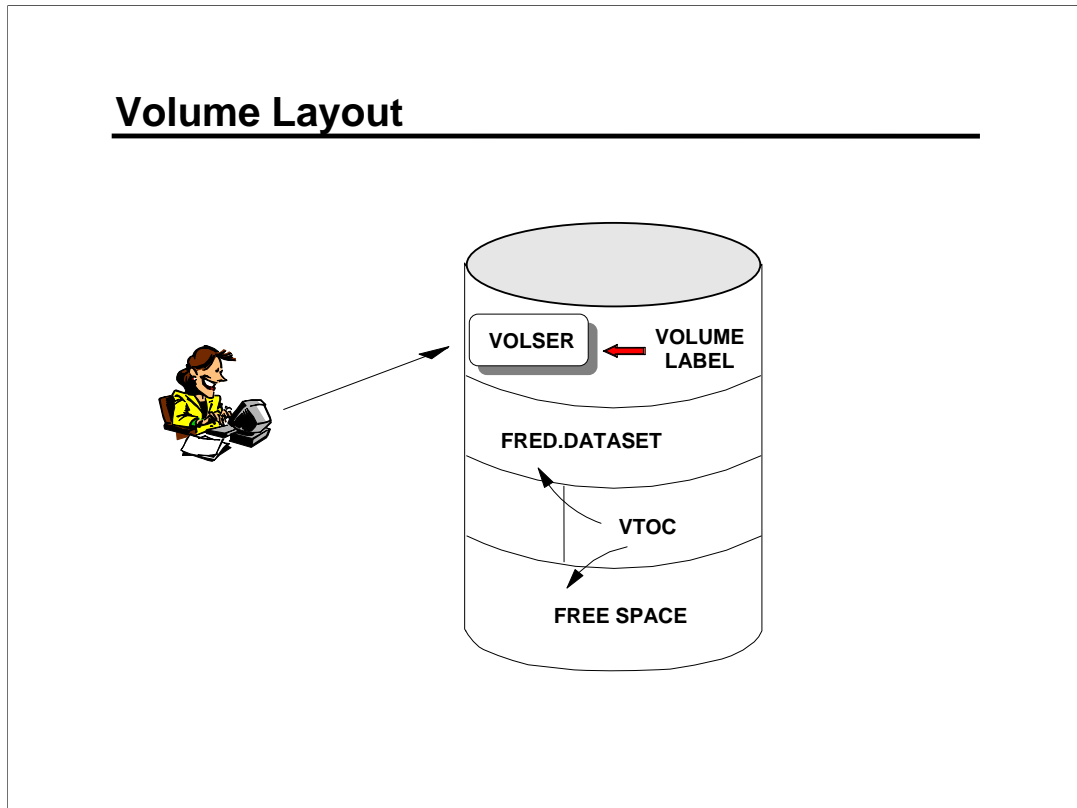
### ESDS

An **entry-sequenced data set** is comparable to a sequential (non-VSAM) data set. Records are sequenced by the order of their entry in the data set, rather than by a key field in the logical record. Records are only added at the end of the data set. Existing records cannot be deleted, only flagged as inactive.

### KSDS

In a **key-sequenced data set**, logical records are placed in the data set in ascending collating sequence by a field, called the **key**. The key contains a unique value, such as an employee number or part number, which determines the record's collating position in the data set. The key must be in the same position in each record, the key data must be contiguous, and each record's key must be unique. After it is specified the value of the key cannot be altered, but the entire record can be erased or deleted. When a new record is added to the data set, it is inserted in its collating sequence by key. A key-sequenced data set always has an **index** that relates key values to the relative locations of the logical records in a data set. This index is called the *prime index*, or simply index. It has two uses:

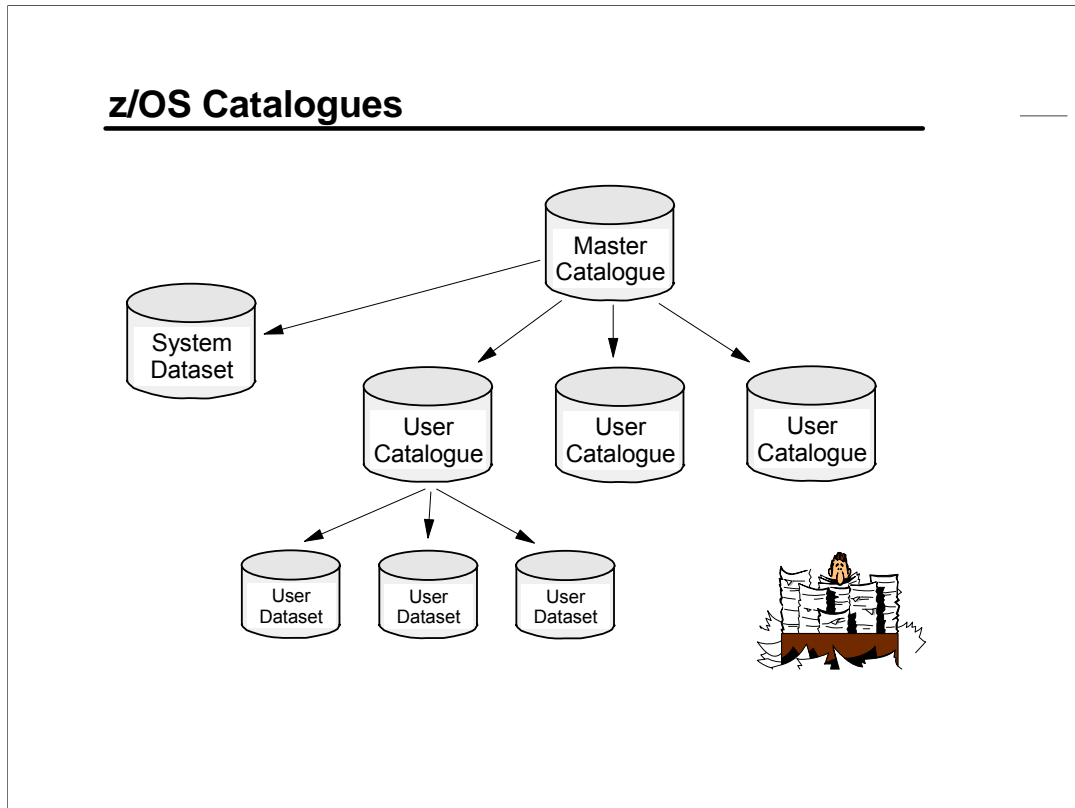
- To locate the collating position when inserting records
- To locate records for retrieval



The Device Support Facility (ICKDSF) initialises a disk volume so that it can be used by z/OS. This process creates the volume label and the Volume Table of Contents (VTOC) on the volume. The volume label contains the volume serial number (VOLSER) and a pointer to the VTOC. A VOLSER is used to identify each volume in the system.

The VTOC keeps track of the free space and the space occupied by data set extents.

To increase performance when locating a data set, VTOCs should be indexed. All SMS managed volumes must use indexed VTOCs. ICKDSF can also be used for media maintenance (i.e. to detect and correct data checks on the volume).



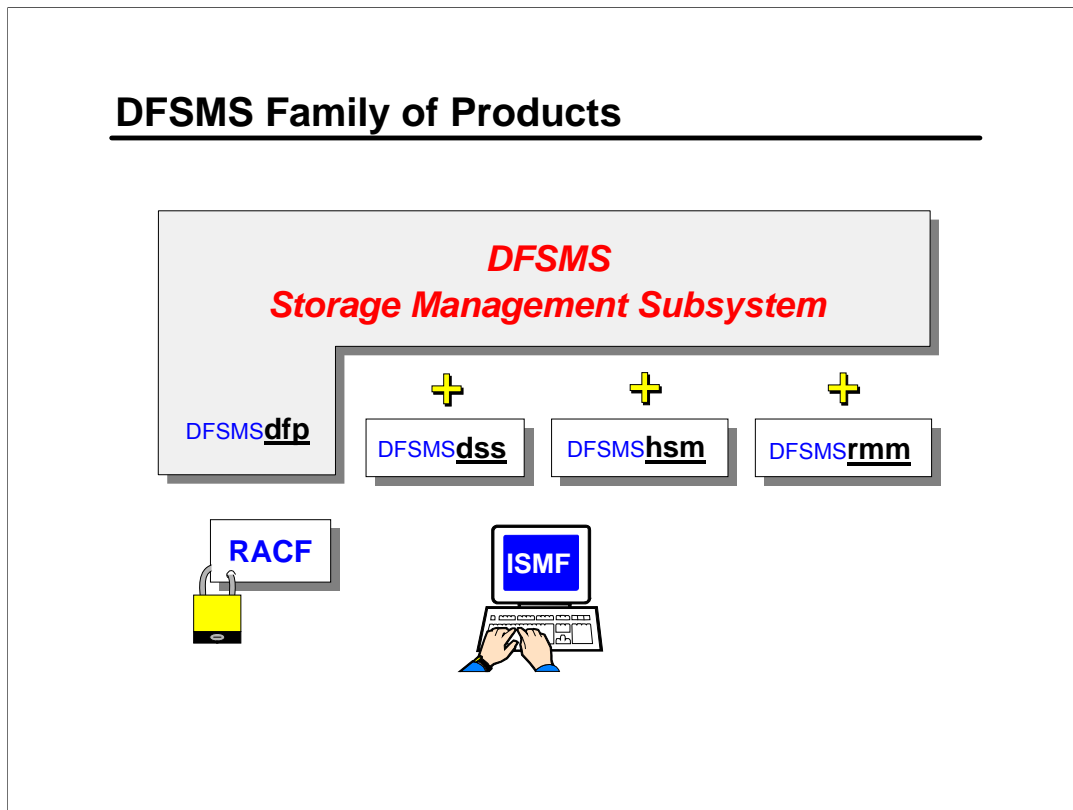
The master catalog should contain only system-related entries. The master catalog contains pointers to system data sets and user catalogs.

User catalogs contain pointers to user VSAM data sets and user non-VSAM data sets.

The catalog management modules and control blocks are located in the private area of the Catalog Address Space. At system initialization the Alias Table and Catalog Pointers are loaded into the catalog address space. Later modifications of this table or pointers will be noted both in the master catalog and the catalog address space.

The Catalog Address Space also keeps a cache which contains master catalog and selected user catalog information. This reduces the overhead of I/O when searching for the location of a dataset.

It is recommended that only master catalog data is cached in the catalog address space. User catalog data can be selectively cached in data spaces managed by VLF (Virtual Lookaside Facility).



In z/OS the group of products listed in the foil are used to control and access data transferred between the operating system and its I/O devices. These products form part of the DFSMS family and include:

#### DFSMSdfp

This product is the interface between z/OS and all attached I/O devices. Its services include creating and deleting data sets on disk or tape, as well as providing the 'access methods' used by programs to read or write to these data sets. It also contains the Program Binder (Linkage Editor), used to prepare program modules for execution, as well as I/O support for printers, graphics terminals and so on. It also includes the code necessary to initialize and control the Storage Management Subsystem.

#### DFSMShsm

A product providing automated space management and availability management.

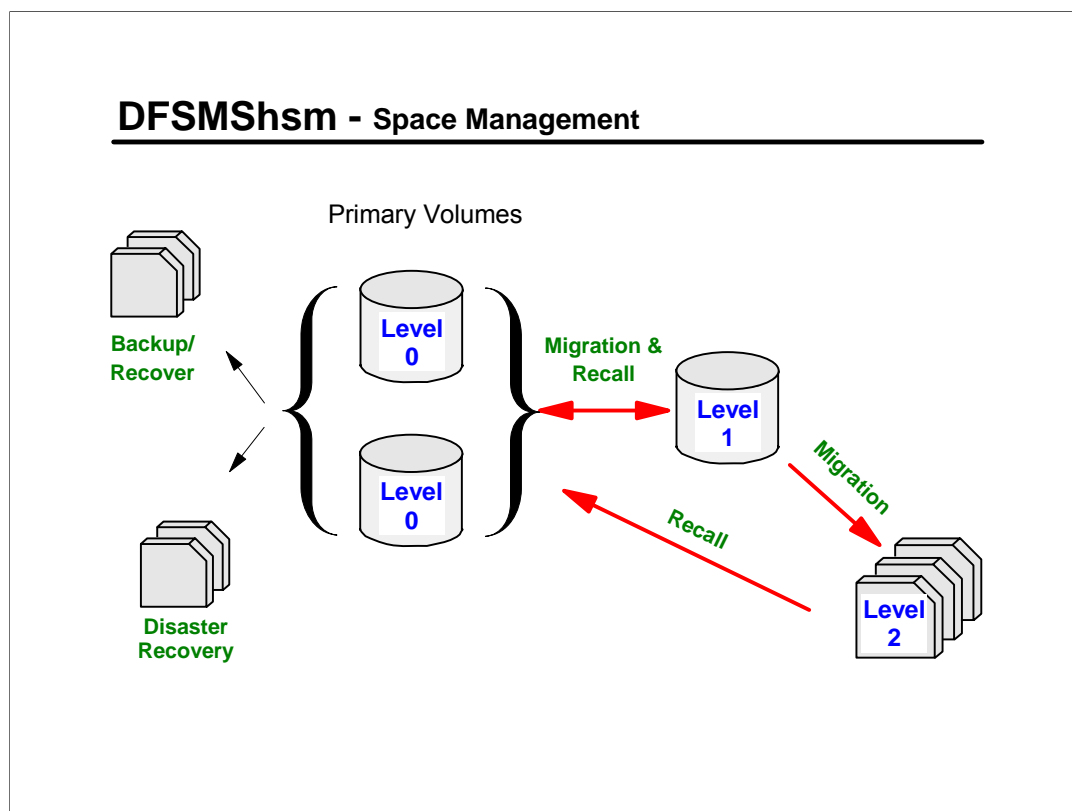
#### DFSMSdss

A high speed 'data mover' product used to manipulate data on disk or tape.

#### RACF

Resource Access Control Facility is a security product that protects data, and other z/OS resources, against unauthorized access, modification or destruction.





DFSMSHsm is a program that automatically performs space management and availability management. It also provides commands to invoke these functions manually. These commands can be issued through the interactive storage management facility (ISMF) or by explicit entry of the command.

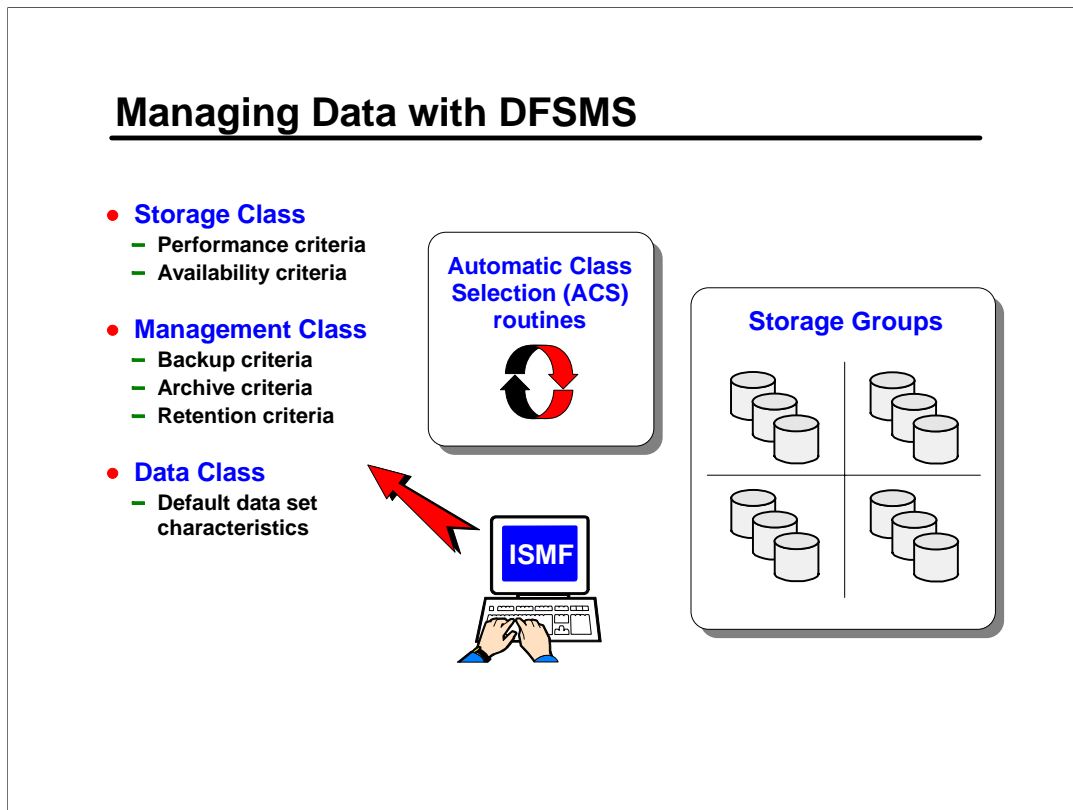
**Space Management** is a process that ensures there is enough storage space on user volumes. DFSMSHsm recreates available space on user volumes by (a) freeing over-allocated space (b) deleting expired data sets and (c) migrating data sets, i.e. moving eligible data sets not used recently (non-active data) to a lower-cost-per-byte storage device in compressed format.

DFSMSHsm records the location of each migrated data set in its control data set. Automatic Migration prepares the computing system for the daily addition of new data by freeing space on DFSMSHsm-managed volumes. Automatic Recall returns a migrated data set to a DFSMSHsm managed volume. If users need to recall migrated data sets they can do so without knowing where the data has been put.

**Availability Management.** This process ensures that there are current backup versions of data sets should recovery be necessary. DFSMSHsm automatically copies new or changed data sets to disk or tape and records the location of each backup version in its control data set. If DFSMS has not been implemented then DFSMSHsm attributes are defined at the volume level. With SMS they can be defined at the data set level.

**Automatic Incremental Backup** ensures that a current copy of new and changed data exists in case the original data sets are damaged or deleted by accident. During this automatic backup, also referred to as an incremental backup, only new or changed data sets are backed up. DFSMSHsm, together with SMS, uses the management class attributes to determine which data gets backed up, how often they get backed up, how many backup versions to maintain and how long they should be kept.

**Automatic Dump** is the process of copying all data from a disk volume to a dump tape volume.



Systems Managed Storage is the z/OS solution to managing storage resources in an efficient manner. With this approach the system determines data placement and automatically manages data availability, performance, space usage and security. The basis for Systems Managed Storage is the Storage Management Subsystem (SMS).

Data set service requirements are defined by the storage administrator, saved in a data set and used during the allocation or movement of data sets. During allocation specific service attributes are selected and assigned to individual data sets.

These attributes include allocation, performance, availability, backup, migration and expiration characteristics.

The user only needs to be concerned with the logical information about the data set, e.g. data set name, record length, space required.

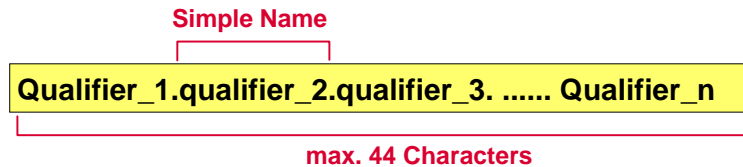
## Data Set Naming Rules

### Data Set Name:

- Simple Names joined by periods(.), with a maximum of 44 characters

### Simple Name:

- 1-8 characters long
- 1st char is A thru Z @ # \$
- 2-8th char is A thru Z @ # \$ and/or 0-9



### Examples:

A  
 SYS1.PROCLIB  
 PETE.TEST  
 ACCT.B100.DATA  
 PAYROLL.DAILY.BACKUP.G0012V00

What is the maximum number of qualifiers?

© Copyright IBM Corporation 2005

Before allocating a new data set, we briefly discuss the data set naming rules.

- A data set name consists of one or more parts, called **qualifiers**. They are connected by periods (.).
- Each qualifier must be a simple name.
  - A simple name is up to eight characters long.
  - The first character of a simple name may be alphabetic (A-Z) or a national character (@,#,¢).
  - The second through eighth character of a simple name may be alphabetic (A-Z), a national character (@,#,¢), or a numeric (0-9).
- The maximum length of a data set name is 44 characters (including the connecting periods). Qualifiers must be simple names. The leftmost qualifier is known as the high-level qualifier (hlq), and the rightmost qualifier is known as the low-level qualifier (llq).

## Partitioned Data Set Naming Rules

### To reference the whole Partitioned Data Set:

Dataset.Name.Up.To.Forty.Four.Char.acters

Dataset Name - max. 44 Characters

**Examples:** SYS1.PROCLIB  
AUES102.PAYROLL.COBOL

### To reference a member of a Partitioned Data Set:

Member Name - max 8 characters

Dataset.Name.Up.To.Forty.Four.Char.acters (Membrane)

Dataset Name - max. 44 Characters

**Examples:** SYS1.PROCLIB(COB2UCLG)  
AUES102.PAYROLL.COBOL(CALCTAX)

© Copyright IBM Corporation 2005

The partitioned and PDS/E data sets are named exactly the same as other z/OS data sets.

PDSs and PDS/Es are host to members, which are really just (small) sequential data sets.

To allocate a member of a PDS or PDS/E, we add the membername (enclosed in brackets) to the end of the data set name.

The maximum length of a data set name including the member name and brackets is 54 characters.

## **Summary**

---

- **Described different types of data sets and data set organisations**
- **Defined the terms such as data, record, data set, block and VTOC**
- **Described z/OS Catalogs**
- **Introduced the DFSMS family of products**

## Time for Lab Exercise

---

### **Exercise 3** Allocate Datasets



© Copyright IBM Corporation 2005

---



**Using  
ISPF**

## Objectives - ISPF

- Edit an existing data set using the ISPF/PDF editor
- Copy or Move an entire data set or selected members
- Rename a data set or member
- Delete an entire data set or a single member
- Work with data set lists (ISPF/PDF option 3.4)





## ISPF Primary Option Menu

```

Menu  Utilities  Compilers  Options  Status  Help
-----
ISPF Primary Option Menu
Option ==>
0 Settings      Terminal and user parameters      User ID . . : MVO4MST
1 View          Display source data or listings   Time . . . : 10:24
2 Edit          Create or change source data      Terminal . . : 3278
3 Utilities     Perform utility functions         Screen . . . : 1
4 Foreground   Interactive language processing   Language . . : ENGLISH
5 Batch         Submit job for language processing Appl ID . . : PDF
6 Command      Enter TSO or Workstation commands TSO logon . : IJACCNT
7 Dialog Test  Perform dialog testing           TSO prefix : MVO4MST
8 LM Facility  Library administrator functions   System ID . : MVSL
9 IBM Products IBM program development products MVS acct . : ACCT#
10 SCLM        SW Configuration Library Manager Release . . : ISPF 4.5
11 Workplace  ISPF Object/Action Workplace
B BookManager  BookManager READ(Online Manuals)
QW QuickRef    Online Quickreference Tool
SD SDSF        System Display and Search Facility

Enter X to Terminate using log/list defaults

F1=Help      F2=Split      F3=Exit      F4=Return    F7=Backward  F8=Forward
F9=Swap      F10=Actions   F12=Cancel

```

This shows the ISPF Primary Option Menu. The main option we will look at in this topic will be option 3 (Utilities).

## Utility Selection Panel

```

Menu  Help
-----
Utility Selection Panel
Option ==>
1 Library      Compress or print data set.  Print index listing.  Print,
                rename, delete, browse, edit or view members
2 Data Set     Allocate, rename, delete, catalog, uncatalog, or display
                information of an entire data set
3 Move/Copy    Move, copy, or promote members or data sets
4 Dslist       Print or display (to process) list of data set names.
                Print or display VTOC information
5 Reset        Reset statistics for members of ISPF library
6 Hardcopy     Initiate hardcopy output
7 Download     Download ISPF C/S, VA for ISPF, transfer map, or data set.
8 Outlist      Display, delete, or print held job output
9 Commands     Create/change an application command table
* Reserved     This option reserved for future expansion.
11 Format       Format definition for formatted data Edit/Browse
12 SuperC      Compare data sets (Standard Dialog)
13 SuperCE     Compare data sets Extended (Extended Dialog)
14 Search-For  Search data sets for strings of data (Standard Dialog)
15 Search-ForE Search data sets for strings of data Extended (Extended Dialog)

F1=Help      F2=Split      F3=Exit      F4=Return    F7=Backward  F8=Forward
F9=Swap      F10=Actions   F12=Cancel

```

Selecting option 3 from the primary menu brings up the utility menu. This menu provides access to facilities to set up and maintain libraries, library members and sequential datasets. It also provides options to work with VSAM datasets.

## Library Utility Panel

---

Menu RefList Utilities Help

---

Library Utility

---

Option ==> \_\_\_\_\_

blank Display member list	I Data set information	B Browse member
C Compress data set	S Short data set information	D Delete member
X Print Index listing	E Edit member	R Rename member
L Print entire data set	V View member	P Print member

Enter "/" to select option  
/ Confirm Member Delete

ISPF Library:

Project . . . . .	}	
Group . . . . .	}	
Type . . . . .	}	
Member . . . . .	}	(If B, D, E, P, R, V, or blank selected)
New name . . . . .		(If R selected)

Other Partitioned or Sequential Data Set:

Data Set Name . . . . . **2** \_\_\_\_\_ (If not cataloged)

Volume Serial . . . . . \_\_\_\_\_ (If not cataloged)

Data Set Password . . . . . \_\_\_\_\_ (If password protected)

F1=Help    F2=Split    F3=Exit    F4=Return    F7=Backward    F8=Forward  
F9=Swap    F10=Actions    F12=Cancel

Choosing option 1 (Library) from the Utility selection panel brings up the Library Utility Panel. As shown in the foil this allows datasets to be compressed (PDS only), printed, renamed, etc;

Note that the data set name can be entered in one of two ways. As shown highlighted as item (1) on the foil, the data set name (and member) can be divided up into PROJECT, GROUP and TYPE (and MEMBER if PDS or PDS/E) and entered into the (1) part of the panel. For example for a data set name of FRED.THIS.THAT then PROJECT = FRED, GROUP = THIS and TYPE = THAT.

Alternatively the data set name may be entered into the area shown as (2) on the foil. If you want to specify the full data set name you MUST enclose the name in single quotes. If you don't enclose the name in quotes then the system will add your prefix HLQ (normally your user ID) in front of the name you specify. For example if your userid (and prefix value) was FRED then you could specify either of the names below against the data set name field.

'FRED.THIS.THAT' --results in FRED.THIS.THAT  
THIS.THAT --results in FRED.THIS.THAT

If you specified FRED.THIS.THAT -- this would result in FRED.FRED.THIS.THAT

## Data Set Utility Panel

```

Menu  RefList  Utilities  Help
-----
                                Data Set Utility
Option ==> _____

  A Allocate new data set          C Catalog data set
  R Rename entire data set        U Uncatalog data set
  D Delete entire data set        S Data set information (short)
blank Data set information        M Enhanced data set allocation
                                V VSAM Utilities

ISPF Library:
Project . . . _____ } 1
Group . . . _____ }
Type . . . _____ }

Other Partitoned, Sequential or VSAM Data Set:
Data Set Name . . . _____ } 2
Volume Serial . . . _____ (If not cataloged, required for option "C")
Data Set Password . . . _____ (If password protected)

F1=Help   F2=Split   F3=Exit   F4=Return   F7=Backward  F8=Forward
F9=Swap   F10=Acti ons  F12=Cancel

```

Choosing option 2 from the Utilities panel brings up the data set utilities selection. Note that you could go directly to this panel from the main ISPF menu by typing 3.2 (i.e. option 3, then option 2). If you are not at the main ISPF panel, but using some other option then you can enter '3.2' which will take you straight to the data set utilities panel. Use this approach to get to other panels that you use frequently.

Note the options that can be performed from this menu.

## Data Set Utility Panel - Allocate Dataset

```

Menu  RefList  Utilities  Help
-----
Allocate New Data Set
Command ==>
Data Set Name . . . . : MVO4MST. SECOND. THIR D           More:  +
Management class . . . STANDARD      (Bl ank for defaul t management class)
Storage class . . . . SCDEFLT        (Bl ank for defaul t storage class)
Volume serial . . . .                (Bl ank for system defaul t volume) **
Device type . . . .                  (Generic unit or devi ce address) **
Data class . . . . SRCFLIB           (Bl ank for defaul t data class)
Space units . . . . TRACK            (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
Average record unit                (M, K, or U)
Primary quanti ty . . 1              (In above uni ts)
Secondary quanti ty . . 4            (In above uni ts)
Di rectory bl ocks . . 3              (Zero for sequenti al data set) *
Record format . . . . FB
Record length . . . . 80
Block size . . . . 0
Data set name type : LIBRARY         (LI BRARY, HFS, PDS, or bl ank) *
(Y Y/MM/DD, Y YYY/MM/DD
YY.DDD, Y YYY.DDD in Julian form
DDDD for retention period in days
or bl ank)
Expiration date . . .
Enter "/" to select option
Allocate Multiple Volumes

F1=Help      F2=Split   F3=Exit    F4=Return   F7=Backward F8=Forward
F9=Swap      F10=Action s F12=Cancel

```

Access to this is via option 3 from utilities panel, or 3.2 or =3.2 as described in previous foil.

This panel is used to specify the function you wish to perform (move, copy, etc;) and the name of the 'source' data set. After typing in the appropriate values and pressing enter you will be presented with another panel asking for details about the 'target' data set. Note that this process will not automatically ALLOCATE the target data set for you.

The target data set must already exist. If not, then you will have to use option 3.2 (Data Set Utility) selecting 'allocate' before any copy/move function can be performed.

## Data Set Information

```

                                Data Set Information
Command ==>
Data Set Name . . . : AUES100.TEST.DATA

General Data                      Current Allocation
Management class . . . : STANDARD    Allocated tracks . : 5
Storage class . . . : BASE           Allocated extents . : 1
Volume serial . . . : USER09        Maximum dir. blocks : 5
Device type . . . . : 3390

Data class . . . . :
Organization . . . : PO              Current Utilization
Record format . . . : FB             Used tracks . . . . : 1
Record length . . . : 80             Used extents . . . : 1
Block size . . . . : 27920          Used dir. blocks . : 1
1st extent tracks . : 5              Number of members . : 0
Secondary tracks . : 1
Data set name type : PDS

Creation date . . . : 2000/10/27
Expiration date . . : ***None***

F1=Help    F2=Split    F3=Exit    F4=         F5=RFind    F6=RChange
F7=Up      F8=Down      F9=Swap    F10=Right  F11=Left    F12=Cursor

```

© Copyright IBM Corporation 2005

To confirm whether the data set has been allocated according to your specifications, select the *blank* option from the *Data Set Utility* panel, enter the name of the data set that has been allocated and press *Enter*. ISPF now displays the actual characteristics of the specified data set in its *Data Set Information* panel.

## Move/Copy Utility Panel

```
Menu RefList Utilities Help
-----
Move/Copy Utility
Option ==> _____

C Copy data set or member(s)      CP Copy and print
M Move data set or member(s)      MP Move and print
L Copy and LMF lock member(s)     LP Copy, LMF lock, and print
P LMF Promote data set or member(s) PP LMF Promote and print

Specify "From" Data Set below, then press Enter key

From ISPF Library:
Project . . . . _____ (--- Options C, CP, L, and LP only ---)
Group . . . . _____
Type . . . . _____
Member . . . . _____ (Blank or pattern for member list,
                          "*" for all members)

From Other Partitioned or Sequential Data Set:
Data Set Name . . . . _____
Volume Serial . . . . _____ (If not cataloged)

Data Set Password . . . . _____ (If password protected)

F1=Hel p      F2=Split      F3=Exi t      F4=Return      F7=Backward      F8=Forward
F9=Swap      F10=Acti ons      F12=Cancel
```

## Data Set List Utility Panel

```

Menu  RefList  RefMode  Utilities  Help
-----
Data Set List Utility

Option ==>> _____

blank Display data set list          P Print data set list
V Display VTOC information           PV Print VTOC information

Enter one or both of the parameters below:
Dsname Level . . . . 1 FRED
Volume serial . . . .

Data set list options
Initial View . . . 1  1. Volume
                    2. Space
                    3. Attrib
                    4. Total

When the data set list is displayed, enter e
"/" on the data set list command field for
an ISPF line command, the name of a TSO command, CLIST, or REXX exec, o
"=" to execute the previous command.

F1=Help   F2=Split   F3=Exit   F4=Return   F7=Backward  F8=Forward
F9=Swap   F10=Action F12=Cancel
    
```

Can be

- HLQ - eg. MV04MST
- HLQ.2LQ - e.g. SYS1.PARMLIB
- HLQ+wildcard - e.g. SYS1.P\*
- or other combinations
- Note SYS1.\* only matches 2nd LQ.  
SYS1.\*\* matches all Qualifiers

This is one of the most useful panels from which you can invoke many functions. As shown at (1), enter an HLQ (High level qualifier), or multiple qualifiers (e.g. FRED.THIS), or a wild card character or set of characters. The system will then do a search for all dataset names matching your selection criteria and display them for further action. The foil shows some examples of use.



## Data Set List Utility cont...

```

Menu Options View Utilities Compilers Help
-----
DSLST - Data Sets Matching MV04MST                               Row 1 of 20
Command ==>>>                                                    Scrol l ==>> CSR

Command - Enter "/" to select action                               Message                               Volume
-----
MVO4MST                                                           *ALI AS
MVO4MST. DSS. ASM                                               MLS004
MVO4MST. EDI T. DATA                                           MLS004
MVO4MST. EX1. COBLI ST                                          MLS002
MVO4MST. HCD. MSGLOG                                            MLS006
MVO4MST.                                                       MLS001
MVO4MST.                                                       MLS001
MVO4MST.                                                       MLS002
MVO4MST.                                                       MLS002
MVO4MST.                                                       *VSAM*
MVO4MST.                                                       MLS001
MVO4MST.                                                       MLS001
MVO4MST.                                                       MLS004
MVO4MST.                                                       MLS001
MVO4MST.                                                       MLS003
MVO4MST.                                                       MLS004
MVO4MST. SPF TEMP1. CNTL                                        MLS004
MVO4MST. STUDLI B. COBOL                                       MLS001
MVO4MST. STUDLI B. DATA                                       MLS001

F1=Hel p      F2=Spli t      F3=Exi t      F4=Return    F7=Backward  F8=Forward
F9=Swap      F10=Acti ons  F12=Cancel
    
```

Once the selection list has been displayed you can use commands against the individual data set lines. The foil gives some examples of use.



## The LOCATE (L) command

Menu Functions Confirm Utilities Help

DSLIST SYS1.PARMLIB 00115

Command ==> L IEASYS

Name Prompt

EEEECO

ADYSET00

ADYSET01

ADYSET02

ARCCDMM

ARCCD00

ASAI PCSP

ASBI PCSP

ATBI PCSP

BLSCECT

BLSCECTX

BPXI PCSP

BPXPRMFS 01.00 99/01/23 08:08 12 12 0 SMCCPP2

BPXPRMOL 01.00 99/05/11 16:36 484 484 0 STEVE8

BPXPRMOO 01.04 99/06/03 13:58 499 484 0 STEVE8

CEEI PCSP

CLOCKOO 01.04 00/04/06 11:25 5 5 0 STEVE8

CLRPARM

CNLENUOO

CNLJPNOO

COFDLFOO

F1=Help F2=Split F3=Exit F4=Return F7=Backward F8=Forward

F9=Swap F10=Actions F12=Cancel

The Locate command is very useful for finding a specific member in a partitioned data set containing many members.

The Locate command can be used on other ISPF panels where lists of items are presented.

If you had entered 'm', for member list against a PDS or PDS/E you would be presented with a list of all the members in the dataset. In very large partitioned data sets, rather than scroll through the list to find the member you want to work with, you can use the LOCATE command to get there more quickly. The foil shows an example of its use.

## Member List

Menu Functions Confirm Utilities Help

Row 00081 of 00115

DSLI ST SYS1.PARMLIB Scroll ==> CSR

Command ==>

Name	Prompt	VV	MM	Changed	Size	Init	Mod	ID
I EASVC00		01.02	99/10/13	13:43	6	3	0	MASSIMO
I EASYSC1		01.03	99/02/12	12:18	2	44	0	STEVE8
I EASYSC2		01.02	99/02/16	15:14	3	1	0	STEVE8
I EAYSOL		01.00	00/04/05	09:59	48	48	0	STEVE8
I EAYS00		01.00	99/03/19	11:01	44	44	0	MASSIMO
I EAYS01		01.19	00/04/06	11:25	48	43	0	STEVE8
I EAYS0						2	0	SMCCPP2
I EAYS9						47	0	MASSIMO
I ECI0S0						2	0	MASSIMO
I EFSSNO						19	0	STEVE8
I FAPRDO								
I GDSMS0						12	0	IBMUSER
I GDSMS0						12	0	SMCCPP2
I KJTS00						108	0	STEVE8
JES2NOD						1442	0	MASSIMO
JES2PAR						755	0	MASSIMO
JES2PAR						754	0	STEVE8
JES2PAR						839	0	STEVE8
LOADBK		01.00	99/06/17	14:51	4	4	0	MASSIMO
LOAD00		01.20	99/10/21	10:21	5	6	0	STEVE8

Example command entries

- / to get pop up selection
- e to edit member
- b to browse member
- r to rename member
- d to delete member
- sub to submit JCL member

F1=Help F2=Split F3=Exit F4=Return F7=Backward F8=Forward  
 F9=Swap F10=Actions F12=Cancel

Having found the member that you wish to work with you can use a variety of line commands against it. This foil shows some examples of use.

## ISPF Editor - ways to create new member

```

Menu  Options  View  Utilities  Compilers  Help
-----
DSLIST - Data Sets Matching MVO4MST                               Row 1 of 20
Command ==>                                                       Scrol | ==> CSR
-----
Command - Enter "/" to select action                               Message           Vol ume
-----
MVO4MST                                                           *ALIAS
MVO4MST. DSS. ASM
MVO4MST. EDI T. DATA
MVO4MST. EX1. COBLIST
MVO4MST. HCD. MSGLOG
MVO4MST. HCD. TERM
MVO4MST. HCD. TRACE
MVO4MST. IODFOO. WORK
MVO4MST. IODFOO. WORK
MVO4MST. IODFOO. WORK
MVO4MST. I SPF. I SP PROF
MVO4MST. MASTER. JCL (newfile)
MVO4MST. MVO4. CRSE
MVO4MST. MVO4. JCL
MVO4MST. RMFOS260. ADMGDF
MVO4MST. RMFOS260. I SPTABLE
MVO4MST. SPFTMP1. CNTL
MVO4MST. STUDLI B. COBOL
MVO4MST. STUDLI B. DATA
MVS002
MLS002
*VSAM*
MLS001
MLS001
MLS004
MLS001
MLS003
MLS004
MLS004
MLS001
MLS001
F1=Hel p      F2=Spl it      F3=Exi t      F4=Return      F7=Backward  F8=Forward
F9=Swap      F10=Acti ons  F12=Cancel

```

If you have just allocated a new PDS or PDS/E you can create the first member (or subsequent new members) by entering an 'e' against the data set name, in the command line entry area, and append the name of the member to the data set name as shown on the foil. Alternatively you could have used option 2 (edit) from the initial panel.

You could also use the 'select' and 'create' primary commands - see the ISPF manual for further details. If you have just allocated a new PDS or PDS/E you can create the first member (or subsequent new members) by entering an 'e' against the data set name, in the command line entry area, and append the name of the member to the data set name as shown on the foil. Alternatively you could have used option 2 (edit) from the initial panel.

## Scrolling

---

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==>> Scroll ==>> PAGE
-----
EDIT      AUES100.TEST.CNTL(AMS) - 01.04      Column 00072
*****  ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //          CLASS=E,
000003 //          MSGLEVEL=(1,1),
000004 //          NOTIFY=AUES100,
000005 //          MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=T
000008 //SYSIN DD *
000009 LISTDATA STATUS VOLUME(USER11) UNIT(3390)
000010 /*
*****  ***** Bottom of Data *****

```

SCROLL  
AMOUNT

**Press F7 to move "UP" the data (that is, towards the first record)**  
**Press F8 to move "DOWN" the data (that is, towards the last record)**  
**The useful SCROLL AMOUNTS are:**

Cursor	Page	Half
--------	------	------

**To change move cursor to SCROLL AREA and enter first character of the amount. For example, C or P or H**

F1=Help	F2=Split	F3=Exit	F4=	F5=RFind	F6=RChange
F7=Up	F8=Down	F9=Swap	F10=Right	F11=Left	F12=Cursor

© Copyright IBM Corporation 2005

The editor allows you to change your view by scrolling up and down as well as left and right. *Columns* lists the columns you are currently editing.

To scroll up and down use the function keys **F7** and **F8** or enter **up** or **down** with or without a numeric value, indicating the number of lines to scroll. If you do not specify a value or use the function keys to scroll, ISPF scrolls by the amount indicated in the *Scroll* ==>> field.

You can change its setting to *PAGE*, *HALF*, *CSR*, *MAX*, *DATA*, or to a numeric value.

Accordingly, ISPF scrolls by the page, half page, to the current cursor position, to the top/bottom of the data set or member, by a page minus a line, or by number of lines specified.

Most people prefer *CSR*. You should try *PAGE* and *HALF*.

The settings of *MAX*, *DATA* or a numeric value are not very useful in the scroll field.

## More Scrolling

```

  File Edit Confirm Menu Utilities Compilers Test Help
Command ==> M Scroll ==> PAGE
-----
EDIT          AUES100.TEST.CNTL(AMS) - 01.04          Columns 00001 00072
***** ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //                      CLASS=E,
000003 //                      MSGLEVEL=(1,1),
000004 //                      NOTIFY=AUES100,
000005 //                      MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=T
000008 //SYSIN DD *
000009 LISTDATA STATUS VOLUME(USER11) UNIT(3390)
000010 /*
***** ***** Bottom of Data *****

```

**Press F7 to move "UP" the data (that is, towards the first record)**  
**Press F8 to move "DOWN" the data (that is, towards the last record)**  
**Alternatively, override SCROLL AMOUNT for the next action but entering an M (for Maximum) or a number you choose (for example, 8) in the command area.**

```

F1=Help   F2=Split   F3=Exit   F4=         F5=RFind   F6=RChange
F7=Up     F8=Down     F9=Swap   F10=Right  F11=Left  F12=Cursor

```

© Copyright IBM Corporation 2005

The scroll amount can be adjusted for the next move by coding *M* for *MAX*, or a numeric value on the command line.

## Scrolling Left and Right

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> _____ Scroll ==> PAGE
-----
EDIT          AUES100.TEST.CNTL(AMS) - 01.04          Columns 00001 00072
***** ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //                CLASS=E,
000003 //                MSGLEVEL=(1,1),
000004 //                NOTIFY=AUES100,
000005 //                MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=T
000008 //SYSIN DD *
000009 LISTDATA STATUS VOLUME(USER11) UNIT(3390)
000010 /*
***** ***** Bottom of Data *****

```

**Press F10 to move "RIGHT"**  
**Press F11 to move "LEFT"**

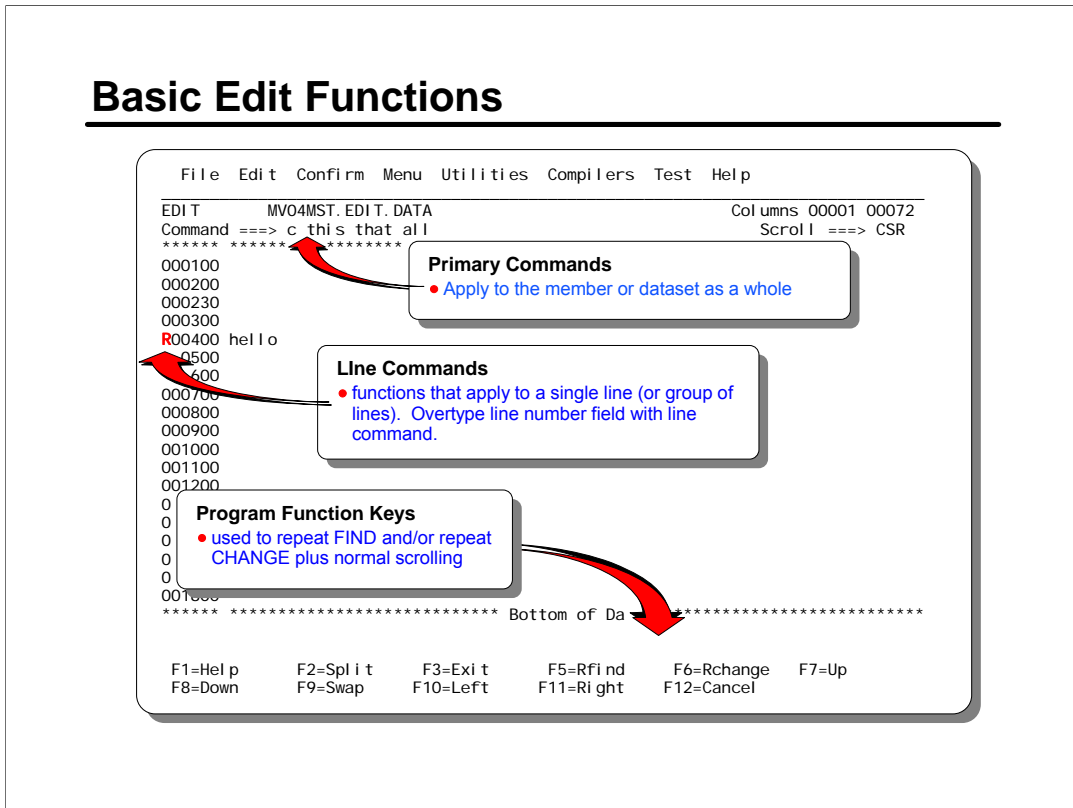
F1=Help	F2=Split	F3=Exit	F4=	F5=RFind	F6=RChange
F7=Up	F8=Down	F9=Swap	F10=Right	F11=Left	F12=Cursor

© Copyright IBM Corporation 2005

Note that you might also need to scroll to the right or left, not just up and down.



## Basic Edit Functions



When editing a file you have various commands at your disposal and these are shown on the foil.

Line commands are entered as shown on the foil.

Primary commands are entered against the command prompt field which may be at the top or bottom of your display. This depends on your ISPF setup. Two of the most useful primary commands are 'c' (change) and 'f' (find).

The format of the change command is

Change oldstring newstring ...plus options.

If a string consists of more than one word, includes special characters or contains a blank, then enclose the string in single quotes. e.g. c 'this thing' 'that item'.

## ISPF Editor - Line commands

The diagram illustrates the ISPF Editor interface with two callout boxes explaining line and block commands.

**Example command entries:**

- **i** to insert line
- **r** to replicate a line
- **d** to delete line
- **c** to copy line, then put 'a' on line after which line is to be copied...or 'b' for before
- **m** to move line (then 'a' or 'b' as above)

**Block commands:**

- example shows block copying of data. **CC** - **CC** on the lines that make up the block, then 'a' for after, or 'b' for before to place block to be copied.
- **rr** - **rr**, repeats a block of lines
- **dd** - **dd**, deletes a block of lines
- **mm** - **mm**, moves a block of lines
- etc;

The editor screen shows a list of lines with command characters in red: **C**00300, **B**00600, **CC**1200, **CC**1600, and **A**01900. Arrows point from the callout boxes to these specific lines.

The foil gives some examples of line commands.

To insert one line enter 'i' in the command line.

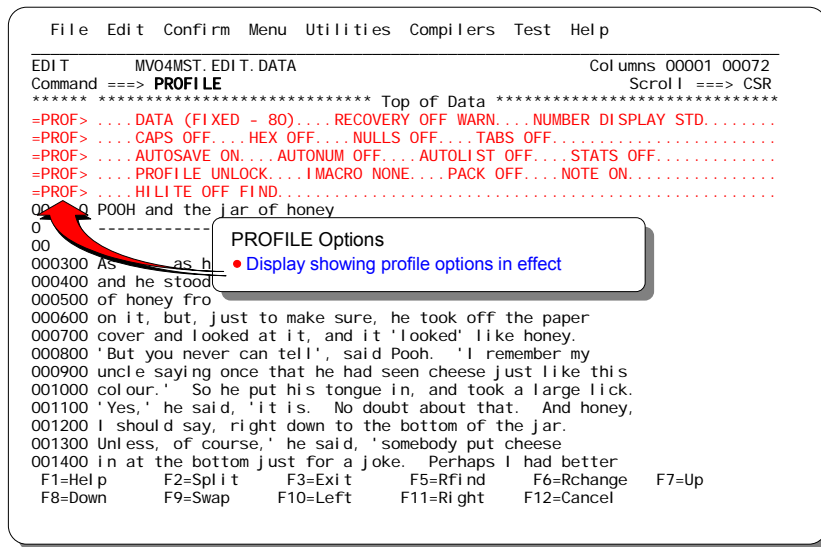
To insert 9 lines, enter '9i' in the command line area, likewise for the 'r' (replicate), 'c' (copy) and 'm' (move) line commands.

To copy one line, type a 'c' in the command line area of the line you wish to copy. Then type an 'a' against the line AFTER which you want to copy the line. Alternatively type a 'b' against the line BEFORE which you want to copy the line. Likewise for the copy and move commands.

To work with a block of lines use a repeat of the command character to enclose the block as shown on the foil.

To help determine column numbers type 'col' in the command line area. To reset the column number line, type RESET on the primary command area.

## ISPF Editor - PROFILE settings



```

File Edit Confirm Menu Utilities Compilers Test Help
EDIT      MV04MST.EDIT.DATA                      Columns 00001 00072
Command ==> PROFILE                               Scroll ==> CSR
***** Top of Data *****
=PROF> ... DATA (FIXED - 80)... RECOVERY OFF WARN... NUMBER DISPLAY STD...
=PROF> ... CAPS OFF... HEX OFF... NULLS OFF... TABS OFF...
=PROF> ... AUTOSAVE ON... AUTONUM OFF... AUTOLIST OFF... STATS OFF...
=PROF> ... PROFILE UNLOCK... IMACRO NONE... PACK OFF... NOTE ON...
=PROF> ... HILITE OFF FIND...
000000 POOH and the jar of honey
000000 -----
000000 as h
000300 As
000400 and he stood
000500 of honey fro
000600 on it, but, just to make sure, he took off the paper
000700 cover and looked at it, and it 'looked' like honey.
000800 'But you never can tell', said Pooh. 'I remember my
000900 uncle saying once that he had seen cheese just like this
001000 colour.' So he put his tongue in, and took a large lick.
001100 'Yes,' he said, 'it is. No doubt about that. And honey,
001200 I should say, right down to the bottom of the jar.
001300 Unless, of course,' he said, 'somebody put cheese
001400 in at the bottom just for a joke. Perhaps I had better
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right     F12=Cancel

```

Each data set/member has specific settings which cover such things as

- whether all entries should be automatically translated to uppercase
- whether highlighting of specific file types should be turned on
- if autosave is in effect and so on.

Type the word 'PROFILE' on the command line to see the settings that are in effect. To change a setting issue the appropriate command on the command line. For example to keep data in the case it is typed enter ==> CAPS OFF. See the documentation for details of other options.

## ISPF Editor - Using Labels

The screenshot shows the ISPF Editor interface with a menu bar (File, Edit, Confirm, Menu, Utilities, Compilers, Test, Help) and a command line. The command entered is `c honey jam .here .there all`. The text being edited is a paragraph about honey. Two labels, `.HERE` and `.THERE`, are placed at the beginning of lines 000700 and 001800 respectively. A callout box titled **'Change' command using labels** provides the following information:

- says 'change' all occurrences of the word 'honey' to 'jam' between lines with labels of '.here' and '.there'. The 'dot' signifies a label.
- Setup labels by overtyping the line number with the label. Label must begin with a dot.
- Clear labels by blanking them out.
- To clear highlighting and other screen effects type **RESET** on the command line.
- In built labels
  - .ZF = First line in file
  - .ZL = Last line in file
  - .ZCSR = Line where cursor is positioned

You can enter labels in the line number area of a file. This can be used to easily locate a line or for use with the change command.

The foil shows an example of using two labels to restrict the scope of the change command. If you were working with a large file and wanted to mark a line (e.g. with label `.here`) then you could use the Primary command - Locate, as mentioned earlier.

## Save and Cancel

---

<u>Command:</u>	<u>Action:</u>
Type: save	<ul style="list-style-type: none"> <li>• Saves Data Set</li> <li>• Remains in Edit-Mode</li> </ul>
Press: F3	<ul style="list-style-type: none"> <li>• Saves Data Set</li> <li>• Returns to previous panel</li> </ul>
Type: cancel/can or Press: F12 (If set to Cancel)	<ul style="list-style-type: none"> <li>• Data Set changes are not saved</li> <li>• Returns to previous panel</li> </ul>

© Copyright IBM Corporation 2005

To save the contents of an altered data set or member enter the **Save** primary command on the command line and press *Enter*. ISPF saves the contents of the edited data set or member, but remains in Edit mode.

To save the contents of an edited data set or member while quitting the editor, press **F3**. ISPF saves the changes and displays the previously shown panel.

To quit the ISPF editor without saving any changes since the last save, press **F12** or enter **cancel** or **can** on the command line and press *Enter*. ISPF returns to the previous panel.

## Split Screen (1 of 2)

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==>> Scroll ==>> PAGE
-----
EDIT      AUES100.TEST.CNTL(AMS) - 01.04      Columns 00001 00072
*****  ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //          CLASS=E,
000003 //          MSGLEVEL=(1,1),
000004 //          NOTIFY=AUES100,
000005 //          MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=T
000008 //SYSIN DD *
000009 LISTDATA STATUS VOLUME(USER11) UNIT(3390)
000010 /*
*****  ***** Bottom of Data *****

```

-  
Note the Cursor position

**Press F2 to "SPLIT" the screen into TWO screens**

F1=Help	F2=Split	F3=Exit	F4=	F5=RFind	F6=RChange
F7=Up	F8=Down	F9=Swap	F10=Right	F11=Left	F12=Cursor

© Copyright IBM Corporation 2005

Notice the PF2 function key (split). Pressing this allows you to bring up an additional screen. You should note that the way the two screens are displayed depends upon where you have positioned the cursor before pressing the PF2 key. If you move the cursor to the very top line of the screen and then press PF2 (split) you will get two full screens. Use PF9 to swap between the two screens.

However, if you position the cursor somewhere in the middle of the screen and then press PF2 (swap) your display will show a portion of both screens, their size being dependent on where you had your cursor before pressing PF2. (Use PF9 to swap between them). This mode may be useful when copying information from one session to another but generally you may find it better to use the full screen sessions as described above.

## Split Screen (2 of 2)

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> _____ Scroll ==> PAGE
-----
EDIT      AUES100.TEST.CNTL(AMS) - 01.04          Columns 00001 00072
*****  ***** Top of Data *****
000001  //AUES100A JOB AUES100,
000002  //                               CLASS=E,
000003  //                               MSGLEVEL=(1,1),
000004  //                               NOTIFY=AUES100,
000005  //                               MSGCLASS=T
000006  //STEP1 EXEC PGM=IDCAMS
000007  //SYSPRINT DD SYSOUT=T
000008  //SYSIN   DD *
F1=Help  F2=Split  F3=Exit  F4=          F5=RFind  F6=RChange
F7=Up    F8=Down   F9=Swap  F10=Right F11=Left  F12=Cursor
-----
Menu Utilities Compilers Options Status Help
-----
ISPF Primary Option Menu
Option ==> _____
0 Settings      Terminal and user parameters      User ID . . : AUES100
1 View          Display source data or listings          Time . . . : 16:57
2 Edit          Create or change source data             Date . . . : 3278
3 Utilities     Perform utility functions                Day . . .  : 1
4 Foreground    Interactive language processing          Language . : ENGLISH
5 Batch         Submit job for language processing       Job class  : ISR
6 Command       Enter TSO or Workstation commands       Job name   : LOGON
7 Dialog Test   Perform dialog testing                  TSO prefix: AUES100
8 LM Facility   Library administrator functions        System ID  : SYS1
F1=Help  F2=Split  F3=Exit  F4=          F5=RFind  F6=RChange
F7=Up    F8=Down   F9=Swap  F10=Right  F11=Left  F12=Cursor

```

© Copyright IBM Corporation 2005

To change the split, position the cursor to the new split line and press F2.  
 You can change the split position at any time.

## The Resplit Screen

```

  File Edit Confirm Menu Utilities Compilers Test Help
Command ==> _____ Scroll ==> PAGE
-----
  Menu Utilities Compilers Options Status Help
-----
                          ISPF Primary Option Menu
Option ==>
0 Settings      Terminal and user parameters      User ID . : AUES100
1 View          Display source data or listings             Time. . . : 16:57
2 Edit          Create or change source data             Terminal. : 3278
3 Utilities     Perform utility functions                   Screen. . : 1
4 Foreground   Interactive language processing           Language. : ENGLISH
5 Batch        Submit job for language processing          Appl ID . : ISR
6 Command      Enter TSO or Workstation commands          TSO logon : LOGON
7 Dialog Test  Perform dialog testing                    TSO prefix: AUES100
8 LM Facility  Library administrator functions           System ID : SYS1
9 IBM Products IBM program development products        MVS acct. : 5820
10 SCLM        SW Configuration Library Manager         Release . : ISPF 4.8
11 Workplace   ISPF Object/Action Workplace

S SDSF        System Display and Search Facility

Enter X to Terminate using log/list defaults

F1=Help      F2=Split      F3=Exit      F4=          F5=RFind     F6=RChange
F7=Up        F8=Down       F9=Swap     F10=Right   F11=Left    F12=Cursor

```

© Copyright IBM Corporation 2005



## Swap the Screen

```

  File Edit Confirm Menu Utilities Compilers Test Help
Command ==> =----- Scroll ==> PAGE
-----
EDIT          AUES100.TEST.CNTL(AMS) - 01.04          Columns 00001 00072
***** ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //          CLASS=E,
000003 //          MSGLEVEL=(1,1),
000004 //          NOTIFY=AUES100,
000005 //          MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=T
000008 //SYSIN DD *
F1=Help      F2=Split    F3=Exit      F4=          F5=RFind     F6=RChange
F7=Up        F8=Down     F9=Swap     F10=Right   F11=Left    F12=Cursor
-----
  Menu Utilities Compilers Options Status Help
-----
                                ISPF Primary Option Menu
Option ==>
0 Settings      Terminal and user parameters   User ID . : AUES100
1 View          Display source data or listings Time . . . : 16:57
2 Edit          Create or change source data   Terminal . : 3278
3 Utilities     Perform utility functions     Screen . . : 1
4 Foreground    Interactive language processing Language . : ENGLISH
5 Batch         Submit job for language processin Appl ID . : ISR
6 Command       Enter TSO or Workstation commands TSO logon : LOGON
7 Dialog Test   Perform dialog testing        TSO prefix: AUES100
F1=Help      F2=Split    F3=Exit      F4=          F5=RFind     F6=RChange
F7=Up        F8=Down     F9=Swap     F10=Right   F11=Left    F12=Cursor

```

© Copyright IBM Corporation 2005

F9 swaps the cursor from one screen to the other.

## The FIND Command

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> Find ACF Scroll ==> PAGE
EDIT AUES100.OS390.DATA(REQ) - 01.01 Columns 00001 00072
***** Top of Data *****
000100 MINIMUM VERSION AND RELEASE REQUIREMENTS FOR IBM PRODUCTS THAT RUN WITH
000110
000120 +-----+-----+-----+-----+
000130 | PRODUCT AND PRODUCT NUMBER | MVS/ESA | OS/390 | OS/390 | OS/39
000140 | | SP 5.2.2 | V1 R1 | V1 R2 | V1 R3
000150 +-----+-----+-----+-----+
000160 | ACF/NETWORK CONTROL | ANY CUR- | ANY CUR- | ANY CUR- | ANY CU
000170 | PROGRAM (ACF/NCP) | RENTLY | RENTLY | RENTLY | RENTLY
000180 | (5668-854, 5668-738, | SUPPORTED | SUPPORTED | SUPPORTED | SUPPOR
000190 | 5668-231, AND 5648-063) | RELEASE | RELEASE | RELEASE | RELEAS
000191 +-----+-----+-----+-----+
000192 | ACF/SYSTEM SUPPORT | ANY CUR- | ANY CUR- | ANY CUR- | ANY CU
000193 | PROGRAM (ACF/SSP) | RENTLY | RENTLY | RENTLY | RENTLY
000194 | (5665-338 AND 5665-041) | SUPPORTED | SUPPORTED | SUPPORTED | SUPPOR
000195 | | RELEASE | RELEASE | RELEASE | RELEAS
000196 +-----+-----+-----+-----+
000197 | ACF/TELECOMMUNICATIONS | V2R4 WITH | V2R4 WITH | V2R4 WITH | V2R4 W
000198 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS
000199 | (5735-RC3) | UY51381, | UY51381, | UY51381, | UY5138
000200 | | UY99771, | UY99771, | UY99771, | UY9977
000201 | | UZ42855, | UZ42855, | UZ42855, | UZ4285
000202 | | AND | AND | AND | AND
000203 | | UW16886 | UW16886 | UW16886 | UW1688
000204 +-----+-----+-----+-----+
000205 | ACF/TELECOMMUNICATIONS | V3R1 WITH | V3R1 WITH | V3R1 WITH | V3R1 W
000206 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS
F1=Help F2=Split F3=Exit F4= F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Right F11=Left F12=Cursor

```

© Copyright IBM Corporation 2005

The **FIND** primary command is used to locate and display the occurrences of a specified character string in the data set or member you are currently editing.

FIND can be abbreviated as **F** or **f**.

Along with the FIND command you always have to specify a character string.

FIND *characterstring* locates and displays the next occurrence of *characterstring*.

Here are some examples of the FIND command:

**FIND** *characterstring* Locates and displays the next occurrence of *characterstring*.

**FIND** *characterstring* **ALL** Locates and displays all occurrences of *characterstring*.

**FIND** '*characterstring*' Locates and displays the next occurrence of *characterstring*, but allows *characterstring* to contain blanks, commas, or keywords of the FIND command, for example, FIND 'first ALL'.

**FIND C**'*characterstring*' Locates and displays the next occurrence of *characterstring*, but is case-sensitive, for example, FIND C'test' will find 'test' but not 'TEST'.

**FIND** *characterstring* *startcolumn endcolumn* Locates the next occurrence of *characterstring* between columns *startcolumn* and *endcolumn* and displays it.

**FIND** *characterstring* **NEXT** | **PREV** | **FIRST** | **LAST**

## The FIND Command

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==>> Scroll ==>> PAGE
EDIT AUES100.OS390.DATA(REQ) - 01.01 CHARS 'ACF' found
***** Top of Data *****
000100 MINIMUM VERSION AND RELEASE REQUIREMENTS FOR IBM PRODUCTS THAT RUN WITH
000110
000120 +-----+-----+-----+-----+-----+
000130 | PRODUCT AND PRODUCT NUMBER | MVS/ESA | OS/390 | OS/390 | OS/3
000140 | SP 5.2.2 | V1 R1 | V1 R2 | V1 R
000150 +-----+-----+-----+-----+-----+
000160 | ACF/NETWORK CONTROL | ANY CUR- | ANY CUR- | ANY CUR- | ANY C
000170 | PROGRAM (ACF/NCP) | RENTLY | RENTLY | RENTLY | RENTL
000180 | (5668-854, 5668-738, | SUPPORTED | SUPPORTED | SUPPORTED | SUPPO
000190 | 5668-231, AND 5648-063) | RELEASE | RELEASE | RELEASE | RELEA
000191 +-----+-----+-----+-----+-----+
000192 | ACF/SYSTEM SUPPORT | ANY CUR- | ANY CUR- | ANY CUR- | ANY C
000193 | PROGRAM (ACF/SSP) | RENTLY | RENTLY | RENTLY | RENTL
000194 | (5665-338 AND 5665-041) | SUPPORTED | SUPPORTED | SUPPORTED | SUPPO
000195 | RELEASE | RELEASE | RELEASE | RELEA
000196 +-----+-----+-----+-----+-----+
000197 | ACF/TELECOMMUNICATIONS | V2R4 WITH | V2R4 WITH | V2R4 WITH | V2R4
000198 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS
000199 | (5735-RC3) | UY51381, | UY51381, | UY51381, | UY513
000200 | | UY99771, | UY99771, | UY99771, | UY997
000201 | | UZ42855, | UZ42855, | UZ42855, | UZ428
000202 | | AND | AND | AND | AND
000203 | | UW16886 | UW16886 | UW16886 | UW168
000204 +-----+-----+-----+-----+-----+
000205 | ACF/TELECOMMUNICATIONS | V3R1 WITH | V3R1 WITH | V3R1 WITH | V3R1
000206 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS
F1=Help F2=Split F3=Exit F4= F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Right F11=Left F12=Cursor =>

```

© Copyright IBM Corporation 2005

Cont...

**NEXT** Locates and displays the next occurrence of *characterstring* starting from the current cursor position. NEXT is the default for FIND.

**PREV** Starts a search for the previous occurrence of *characterstring* starting at the line preceding the first line being displayed. If the FIND command reaches the top of the data displayed it wraps around to the bottom.

**FIRST** Locates and displays the first occurrence of *characterstring* starting from the top of the data being edited.

**LAST** Locates the last occurrence of *characterstring* within the data set or member.

**RFIND** Repeats the action of the previous FIND command

**Note:** For additional parameters of the FIND command refer to: *ISPF User's Guide Vol I*.

## The BOUNDS and EXCLUDE Commands

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> Bounds 1 10; Exclude ACF all          Scroll ==> PAGE
EDIT      AUES100.OS390.DATA(REQ) - 01.01        Columns 00001 00072
*****  ***** Top of Data *****
000100  MINIMUM VERSION AND RELEASE REQUIREMENTS FOR IBM PRODUCTS THAT RUN WITH
000110
000120  +-----+-----+-----+-----+-----+-----+
000130  | PRODUCT AND PRODUCT NUMBER | MVS/ESA | OS/390 | OS/390 | OS/390 | OS/390 |
000140  |                               | SP 5.2.2 | V1 R1  | V1 R2  | V1 R3  |
000150  +-----+-----+-----+-----+-----+-----+
000160  | ACF/NETWORK CONTROL        | ANY CUR- | ANY CUR- | ANY CUR- | ANY CU
000170  | PROGRAM (ACF/NCP)          | RENTLY   | RENTLY   | RENTLY   | RENTLY
000180  | (5668-854, 5668-738,      | SUPPORTED | SUPPORTED | SUPPORTED | SUPPOR
000190  | 5668-231, AND 5648-063)   | RELEASE  | RELEASE  | RELEASE  | RELEAS
000191  +-----+-----+-----+-----+-----+-----+
000192  | ACF/SYSTEM SUPPORT         | ANY CUR- | ANY CUR- | ANY CUR- | ANY CU
000193  | PROGRAM (ACF/SSP)         | RENTLY   | RENTLY   | RENTLY   | RENTLY
000194  | (5665-338 AND 5665-041)   | SUPPORTED | SUPPORTED | SUPPORTED | SUPPOR
000195  |                               | RELEASE  | RELEASE  | RELEASE  | RELEAS
000196  +-----+-----+-----+-----+-----+-----+
000197  | ACF/TELECOMMUNICATIONS    | V2R4 WITH | V2R4 WITH | V2R4 WITH | V2R4 W
000198  | ACCESS METHOD (ACF/TCAM)   | PTFS     | PTFS     | PTFS     | PTFS
000199  | (5735-RC3)                | UY51381, | UY51381, | UY51381, | UY5138
000200  |                               | UY99771, | UY99771, | UY99771, | UY9977
000201  |                               | UZ42855, | UZ42855, | UZ42855, | UZ4285
000202  |                               | AND      | AND      | AND      | AND
000203  |                               | UW16886  | UW16886  | UW16886  | UW1688
000204  +-----+-----+-----+-----+-----+-----+
000205  | ACF/TELECOMMUNICATIONS    | V3R1 WITH | V3R1 WITH | V3R1 WITH | V3R1 W
000206  | ACCESS METHOD (ACF/TCAM)   | PTFS     | PTFS     | PTFS     | PTFS
F1=Help   F2=Split   F3=Exit   F4=      F5=RFind   F6=RChange
F7=Up     F8=Down    F9=Swap   F10=Right F11=Left  F12=Cursor

```

© Copyright IBM Corporation 2005

Use the **BOUNDS** primary command to specify the range of columns to which other commands such as, FIND, EXCLUDE, CHANGE and so on apply. Columns outside the specified area remain unaffected.

**BOUNDS** *leftcolumn rightcolumn* sets the boundaries for all following editor commands to columns *leftcolumn* and *rightcolumn*.

The **EXCLUDE** primary command which can also be entered as **X** excludes specific lines of the data set or member being edited from display.

**EXCLUDE** *characterstring ALL* excludes all lines containing the specified *characterstring* from display.

## The BOUNDS and EXCLUDE Commands

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> Scroll ==> PAGE
EDIT AUES100.OS390.DATA(REQ) - 01.01 4 CHARS 'ACF'
***** ***** Top of Data *****
000100 MINIMUM VERSION AND RELEASE REQUIREMENTS FOR IBM PRODUCTS THAT RUN WITH
000110
000120 +-----+-----+-----+-----+-----+
000130 | PRODUCT AND PRODUCT NUMBER | MVS/ESA | OS/390 | OS/390 | OS/390 |
000140 | | SP 5.2.2 | V1 R1 | V1 R2 | V1 R3 |
000150 +-----+-----+-----+-----+-----+
- - - - - 1 Line(s) not Displayed
000170 | PROGRAM (ACF/NCP) | RENTLY | RENTLY | RENTLY | RENTLY |
000180 | (5668-854, 5668-738, | SUPPORTED | SUPPORTED | SUPPORTED | SUPPORT |
000190 | 5668-231, AND 5648-063) | RELEASE | RELEASE | RELEASE | RELEAS |
000191 +-----+-----+-----+-----+-----+
- - - - - 1 Line(s) not Displayed
000193 | PROGRAM (ACF/SSP) | RENTLY | RENTLY | RENTLY | RENTLY |
000194 | (5665-338 AND 5665-041) | SUPPORTED | SUPPORTED | SUPPORTED | SUPPORT |
000195 | | RELEASE | RELEASE | RELEASE | RELEAS |
000196 +-----+-----+-----+-----+-----+
- - - - - 1 Line(s) not Displayed
000198 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS |
000199 | (5735-RC3) | UY51381, | UY51381, | UY51381, | UY5138 |
000200 | | UY99771, | UY99771, | UY99771, | UY9977 |
000201 | | UZ42855, | UZ42855, | UZ42855, | UZ4285 |
000202 | | AND | AND | AND | AND |
000203 | | UW16886 | UW16886 | UW16886 | UW1688 |
000204 +-----+-----+-----+-----+-----+
- - - - - 1 Line(s) not Displayed
000206 | ACCESS METHOD (ACF/TCAM) | PTFS | PTFS | PTFS | PTFS |
F1=Help F2=Split F3=Exit F4= F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Right F11=Left F12=Cursor

```

© Copyright IBM Corporation 2005

Use the **BOUNDS** primary command to specify the range of columns to which other commands such as, FIND, EXCLUDE, CHANGE and so on apply. Columns outside the specified area remain unaffected.

Here are some examples of BOUNDS:

**BOUNDS** *leftcolumn rightcolumn* Sets the boundaries for all following editor commands to columns *leftcolumn* and *rightcolumn*.

**BOUNDS** *leftcolumn* \* Sets the left boundary for all following editor commands to column *leftcolumn*.

**BOUNDS** Resets the boundaries to their default setting.

The **EXCLUDE** primary command which can also be entered as **X** excludes specific lines of the data set or member

**EXCLUDE** *characterstring* Excludes the line containing the next occurrence of the specified *characterstring* starting from the current cursor position.

**EXCLUDE** *characterstring startcolumn endcolumn* Excludes the next line containing an occurrence of the specified *characterstring* within the columns entered with the command.

**EXCLUDE** *characterstring ALL* Excludes all lines containing the specified *characterstring* from display.

**EXCLUDE** '*characterstring*' Locates and displays the next occurrence of *characterstring*, but allows *characterstring* to contain blanks, commas, or keywords of the EXCLUDE command, for example, EXCLUDE 'x ALL'.

**EXCLUDE** "\*" *column* Excludes all lines containing any character in column *column* from display.

## The CREATE Command

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> CREATE JOBCARD Scroll ==> PAGE
EDIT AUES100.TEST.CNTL(JOB19) - 01.0 Columns 00001 00072
***** ***** Top of Data*****
CC0001 //AUES100A JOB AUES100,
000002 // TIME=(,15),
000003 // CLASS=E,
000004 // MSGLEVEL=(1,1),
000005 // NOTIFY=AUES100,
CC0006 // MSGCLASS=T
000007 //STEP1 EXEC PGM=IEBGENER
000008 //SYSUT1 DD DSN=AUES100.TEST.EXEC,DISP=SHR
000009 //SYSUT2 DD DSN=AUES100.TEST.EXEC1,LIKE=AUES100.TEST.EXEC,
000010 // DISP=(NEW,CATLG,CATLG)
000011 //SYSIN DD DUMMY
000012 //SYSPRINT DD SYSOUT=T
***** ***** Bottom of Data*****

```

Note the  
CCs

New Member JOBCARD

F1=Help F2=Split F3=Exit F4= F5= F6= F7=Up F8=Down F9=Swap F10=Right F11=Left F12=Right

© Copyright IBM Corporation 2005

The CREATE macro command creates a member of a partitioned data set from the data you are editing. This command cannot be used to create a sequential data set. CREATE adds a member to a partitioned data set only if a member with the same name does not already exist. Use REPLACE if the member already exists.

CREATE *membername* creates a new member from the data set you are editing. Use *cc* in the numeric prefix area of a line to mark the start and the end of the data to be copied into the new member.

## The CREATE Command

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ===> Scroll ==> PAGE
EDIT AUES100.TEST.CNTL(JOBCARD) - 01.00 Columns 00001 00072
***** ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 // TIME=(,15),
000003 // CLASS=E,
000004 // MSGLEVEL=(1,1),
000005 // NOTIFY=AUES100,
000006 // MSGCLASS=T
***** ***** Bottom of Data *****

```

New member  
JOBCARD created

```

F1=Help      F2=Split    F3=Exit     F4=         F5=RFind    F6=RChange
F7=Up        F8=Down     F9=Swap     F10=Right   F11=Left    F12=C

```

© Copyright IBM Corporation 2005

The CREATE macro command creates a member of a partitioned data set from the data you are editing. This command cannot be used to create a sequential data set. CREATE adds a member to a partitioned data set only if a member with the same name does not already exist. Use REPLACE if the member already exists.

Enter the command in one of the following forms:

**CREATE** *membername* Creates a new member from the data set you are editing. Use *cc* in the numeric prefix area of a line to mark the start and the end of the data to be copied into the new member.

**CREATE** *membername startline endline*

Creates a new member from the data set you are editing and copies the data in lines *startline* to *endline* into the new data set.

## The COPY Command

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> COPY JOBCARD Scroll ==> PAGE
EDIT AUES100.TEST.CNTL(JOB29) - 01.0 Columns 00001 00072
***** ***** Top of Data*****
00b007 //STEP1 EXEC PGM=IEBGENER
000008 //SYSUT1 DD DSN=AUES100.TEST.EXEC,DISP=SHR
000009 //SYSUT2 DD DSN=AUES100.TEST.EXEC1,LIKE=AUES100.TEST.EXEC,
000010 // DISP=(NEW,CATLG,CATLG)
000011 //SYSIN DD DUMMY
000012 //SYSPRINT DD SYSOUT=T
***** ***** Bottom of Data*****

```

New Member JOBCARD

```

F1=Help F2=Split F3=Exit F4= F5=
F7=Up F8=Down F9=Swap F10=Right F11=

```

© Copyright IBM Corporation 2005

The COPY command imports records into your edit session at the point you choose. Use **A** or **B** to indicate where the imported records are positioned.

The MOVE command does the same as the COPY command, except that MOVE deletes the original data after the records have been imported.

This example imports the whole member, and the member must be in the same PDS.

There are options which allow you to import from other sources. Refer to ISPF HELP for details.



## COPY'd

```

File Edit Confirm Menu Utilities Compilers Test Help
Command ==> Scroll ==> PAGE
EDIT AUES100.TEST.CNTL(JOB29) - 01.0 member JOBCARD copied
***** ***** Top of Data *****
000001 //AUES100A JOB AUES100,
000002 //          TIME=(,15),
000003 //          CLASS=E,
000004 //          MSGLEVEL=(1,1),
000005 //          NOTIFY=AUES100,
000006 //          MSGCLASS=T
000007 //STEP1 EXEC PGM=IEBGENER
000008 //SYSUT1 DD DSN=AUES100.TEST.EXEC,DISP=SHR
000009 //SYSUT2 DD DSN=AUES100.TEST.EXEC1,LIKE=AUES100.TEST.EXEC,
000010 //          DISP=(NEW,CATLG,CATLG)
000011 //SYSIN DD DUMMY
000012 //SYSPRINT DD SYSOUT=T
***** ***** Bottom of Data *****

F1=Help    F2=Split  F3=Exit   F4=       F5=RFind  F6=RChange
F7=Up      F8=Down   F9=Swap   F10=Right F11=Left  F12=Cursor

```

© Copyright IBM Corporation 2005

The COPY command imports the records you choose.

There are options which allow you to select the records to be imported. Use ISPF HELP for details.

## Additional Primary Commands

---

- Locate** - Displays a particular line in the data
- Reset** - Resets the editor display
- Submit** - Submits the edited data as a jobstream for background execution
- Renum** - Switches NUMBER mode on and renumbers the data
- Unnum** - Switches NUMBER mode off and blanks out sequence numbers
- Move** - Moves a member or data set into this edit session
- Edit** - Causes a recursive entry into edit
- Undo** - Undoes the changes of the last command
- Recovery** - Enables Undo and automatically backs up every time you press Enter. Highly recommended.

© Copyright IBM Corporation 2005

### Locate

The LOCATE primary command allows you to scroll up or down to a specified line, which is then displayed as the first line.

### Reset

The RESET primary command can restore line numbers in the line command area when those line numbers have been replaced by labels, pending line commands, error flags, and change flags. RESET can also delete special lines from the display, redisplay excluded lines, and temporarily disable the highlighting of FIND strings.

### Submit

The SUBMIT primary command submits the member or data set you are editing or a part of the member or data set as a batch job.

### Renum

RENUM activates the number mode and renumbers all lines, usually in increments of 100.

### Unnum

The UNNUMBER primary command sets all sequence fields to blanks, turns off number mode, and positions the data so that column 1 is the first column displayed.

### Move

MOVE specifies a member of the partitioned data set being edited to be moved into the data being edited.

### Edit

The EDIT primary command allows you to edit another sequential data set or partitioned data set member during your current edit session.

### Undo

If you enter an edit primary, line, or macro command, or type over existing data by mistake, you can restore your data with the UNDO primary command. To use the UNDO command, you must have either RECOVERY ON or SETUNDO on.

## More Primary Commands

---

- COLS** - Displays a protected non-scrollable line on the EDIT screen
- Hide** - Removes each **n Line(s) not displayed** message from the screen
- RESet Hide** - Redisplays all **n Line(s) not displayed** messages on the screen

© Copyright IBM Corporation 2005

### Cols

The Cols line differs from that displayed by the Cols *line* command in that line-command field is protected. This means that it can not be moved, copied, or deleted with line commands. This line does not scroll with the Data.

### Hide x

The Hide primary command removes each **n Line(s) not displayed** message from the display where lines have been hidden from display by the **Exclude** command. Instead the line-number field of the preceding line is underscored (if the terminal supports underscoring) to indicate that part of the data is not being displayed.

### RESet Hide

The **Hide operand** of the **Reset** command redisplay **n Line(s) not displayed** messages that were previously excluded via the **Hide command**.

## **Summary**

---

- **Described how ISPF can be used**
- **Illustrated some of the basic functions of ISPF**
- **Described how ISPF can be used with datasets**
- **Covered the basic use of the ISPF editor**

## Time for Lab Exercise

---

### **Exercise 5** ISPF Editor



© Copyright IBM Corporation 2005



## **Virtual Storage & Address Spaces**

In order for work to be processed, instructions and data must be in central storage.

On early operating systems any job/task that was to be run was allocated the full amount of storage it was likely to use for the full duration of the job/task.

This included space for the program(s) and data. This allocation was for 'real' storage.

Over time the speed of processors increased such that many more programs could be run concurrently than could be contained in central storage.

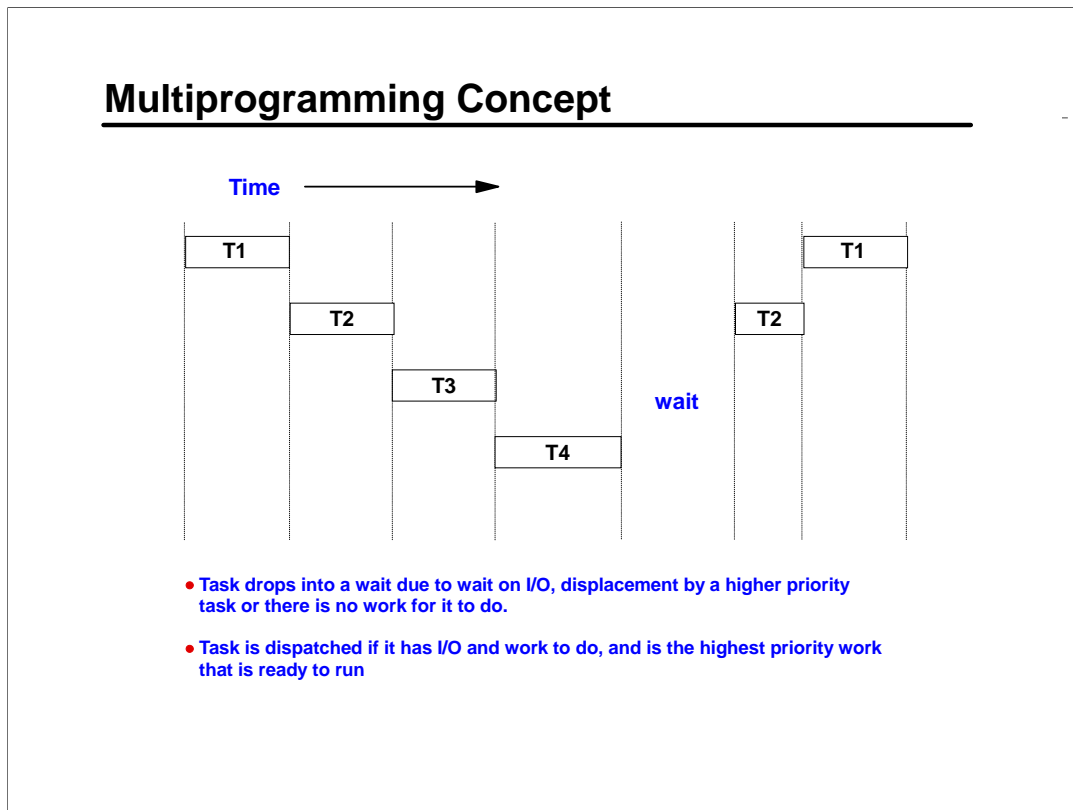
## Objectives

---



- Explain the virtual address space concept
- Describe the different areas of an address space
- Describe the differences between an address space and a data space
- Explain paging and swapping



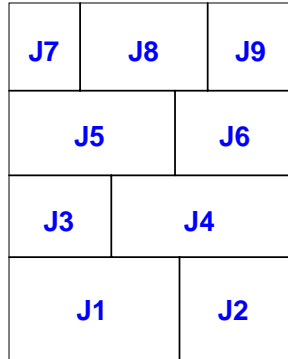


A program cannot process data unless it is brought into central storage. The time it takes to read or write data to/from an I/O device is very long compared to the speed at which a processor can execute instructions. A single processor (engine), on a current-day System z server executes in excess of 200 million instructions per second. Consequently a central processor is able to interleave many tasks while waiting for I/O operations to complete.

This 'interleaving' is called Multiprogramming. Multiprogramming will 'interrupt' the program requesting an I/O and present another task to be executed by the processor rather than wait for the I/O to complete. When interrupting the current program, z/OS stores all necessary information in order to reschedule this program later on when its I/O operation has completed.

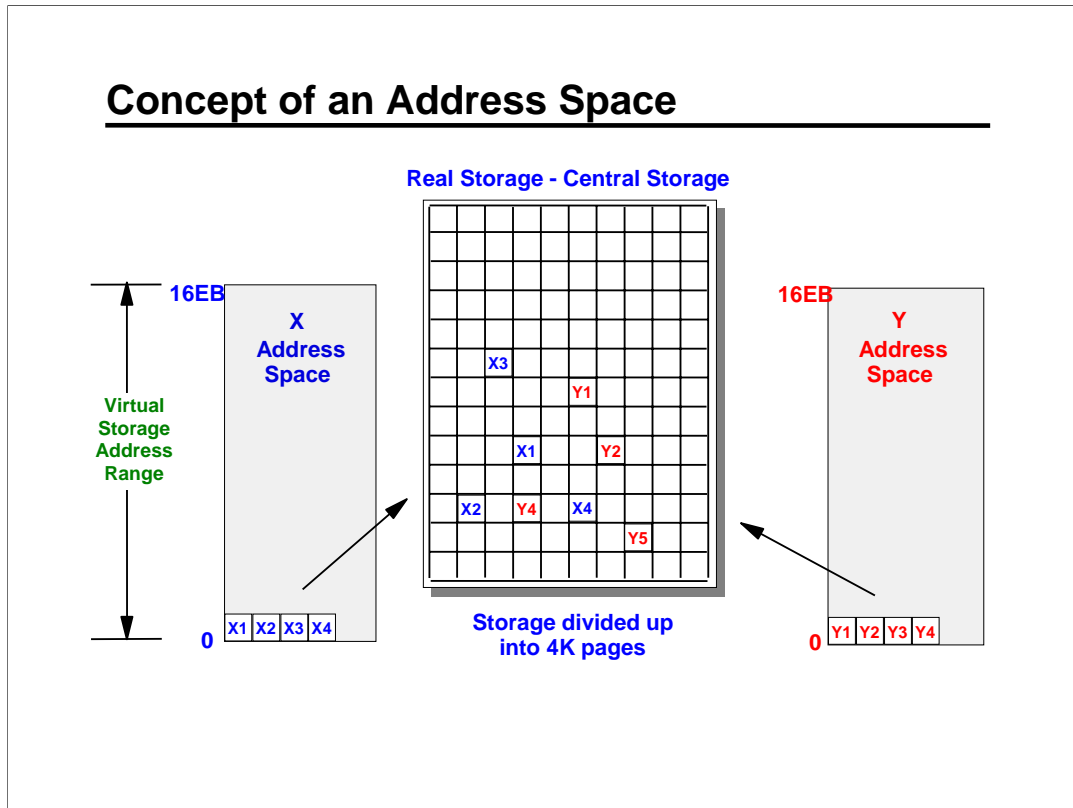
## Early Implementation - Fixed Storage Allocation

Real  
Storage



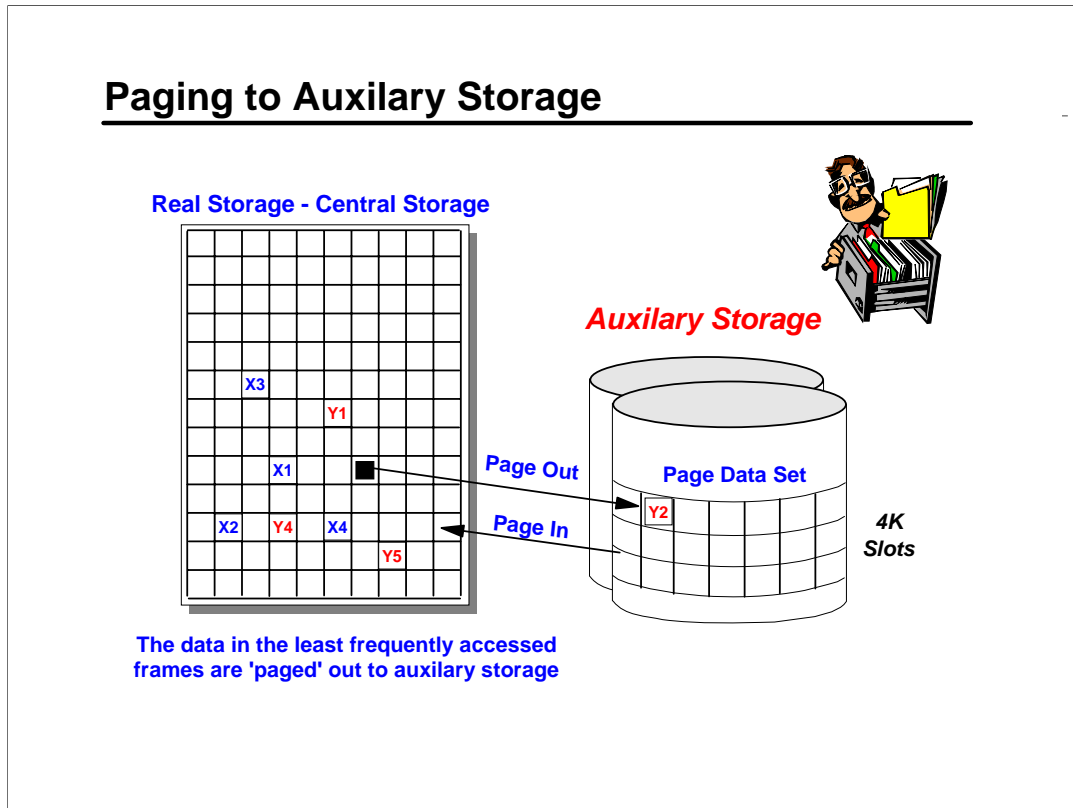
The level of multi-tasking was limited by the number of jobs that could fit in real storage

In order for work to be processed, instructions and data must be in central storage. On early operating systems any job/task that was to be run was allocated the full amount of storage it was likely to use for the full duration of the job/task. This included space for the program(s) and data. This allocation was for 'real' storage. Over time the speed of processors increased such that many more programs could be run concurrently than could be contained in central storage.



To overcome the limitation of tying up large amounts of real storage to individual jobs/tasks the concept of Virtual Storage was introduced. This is implemented by dividing up the real/central storage into 4K units, called pages and by 'pretending' that each job/task has access to up to 16 exabytes of storage (in an address space).

An address space is the basic unit of dispatching work in z/OS and describes the virtual storage address ranges available to a program. Storage is only allocated as required rather than the total amount that the program will require over the course of its execution. From the moment of loading a program till the end of its execution, the virtual addresses of the pages it references remain the same, whether these pages are located in real or auxiliary storage.



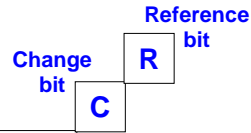
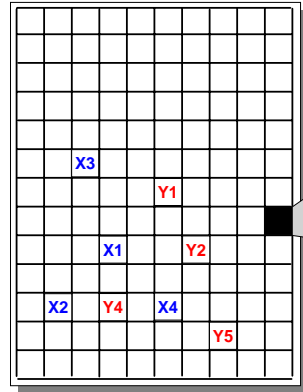
As work progresses on a system the demand for storage space will increase to the point where all of real storage has been allocated. This is quite normal because the concept of virtual storage allows for over-allocation of storage. In order to satisfy new requests for storage the system needs to free up space. It moves little used or least recently used data from central storage to auxiliary storage.

Auxiliary storage is made up of one or more 'page data sets' which reside on disk. Each page data set is divided up into 4k slots. The process of moving data between central storage and auxiliary storage is known as paging. The movement from central storage to auxiliary is known as a page out.

When a user/program tries to access a page of data that is not in central storage this causes a page fault. When this happens the z/OS paging routines 'page in' the required page from auxiliary storage.

## Central Storage Pages/Frames

Real Storage - Central Storage



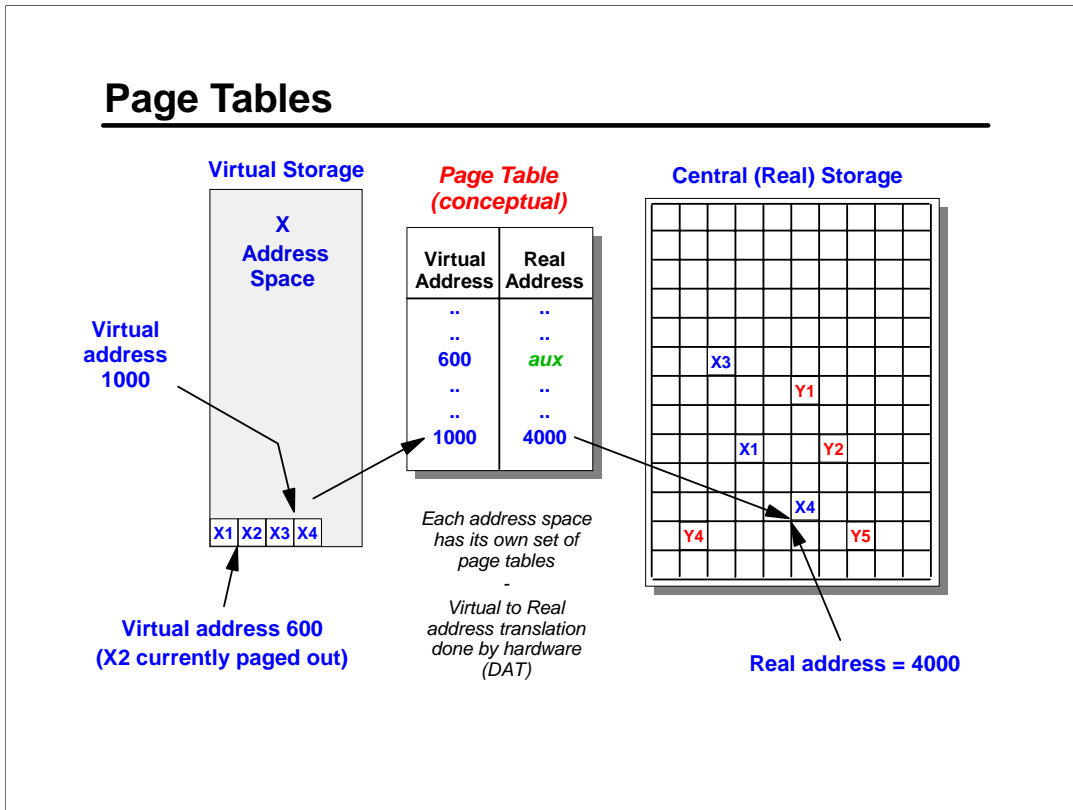
**Each 4K frame has a change bit and a reference bit. Both are set by hardware.**

*Also have access control bits and fetch protection bits for each 4K frame*

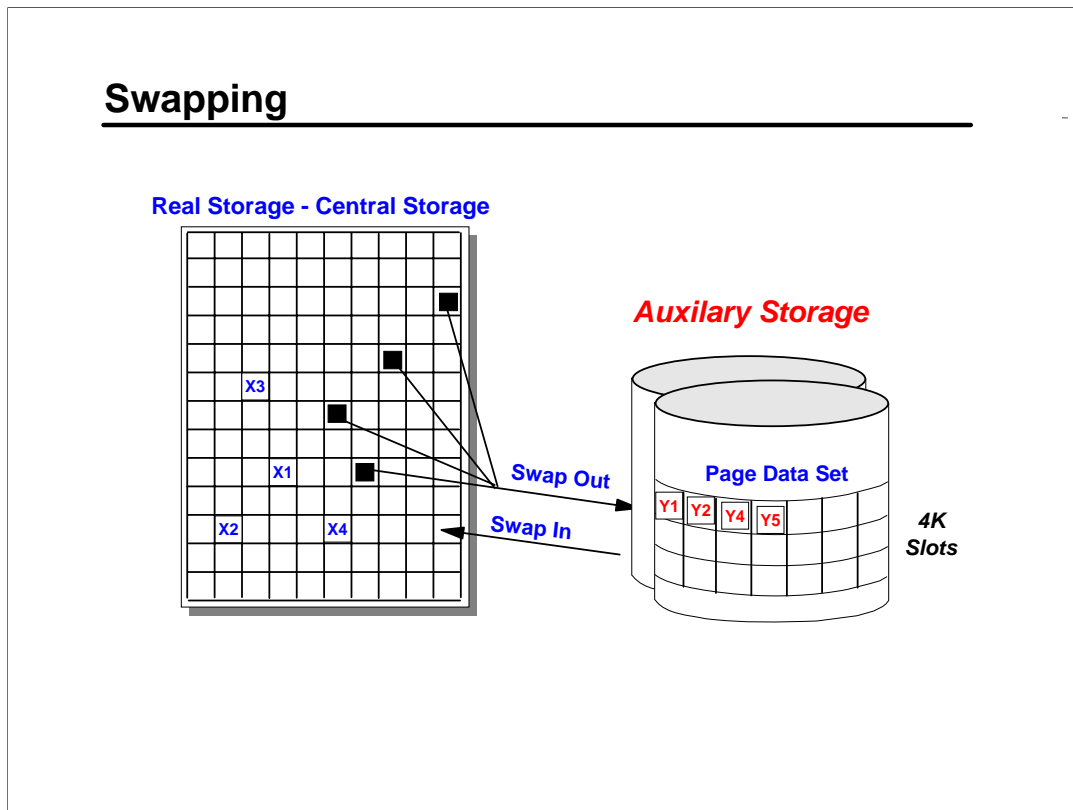
**Storage divided up into 4K frames.  
One page occupies one frame**

In order to determine which data is the least recently used, each physical 4k frame of central storage has a 'change bit' and a 'reference bit' associated with it which is set on by hardware. When the data in a given frame is read then just the reference bit is set on. When the frame is written to then both the reference and change bits are set on.

Part of the z/OS system (Real Storage Manager - RSM) periodically checks and resets these bits thereby building up a table of the least recently used frames. It uses this information to determine which data should be paged out. In fact, rather than work in 'panic mode' and wait till storage is completely full, RSM aims to maintain a certain percentage of free space at all times.

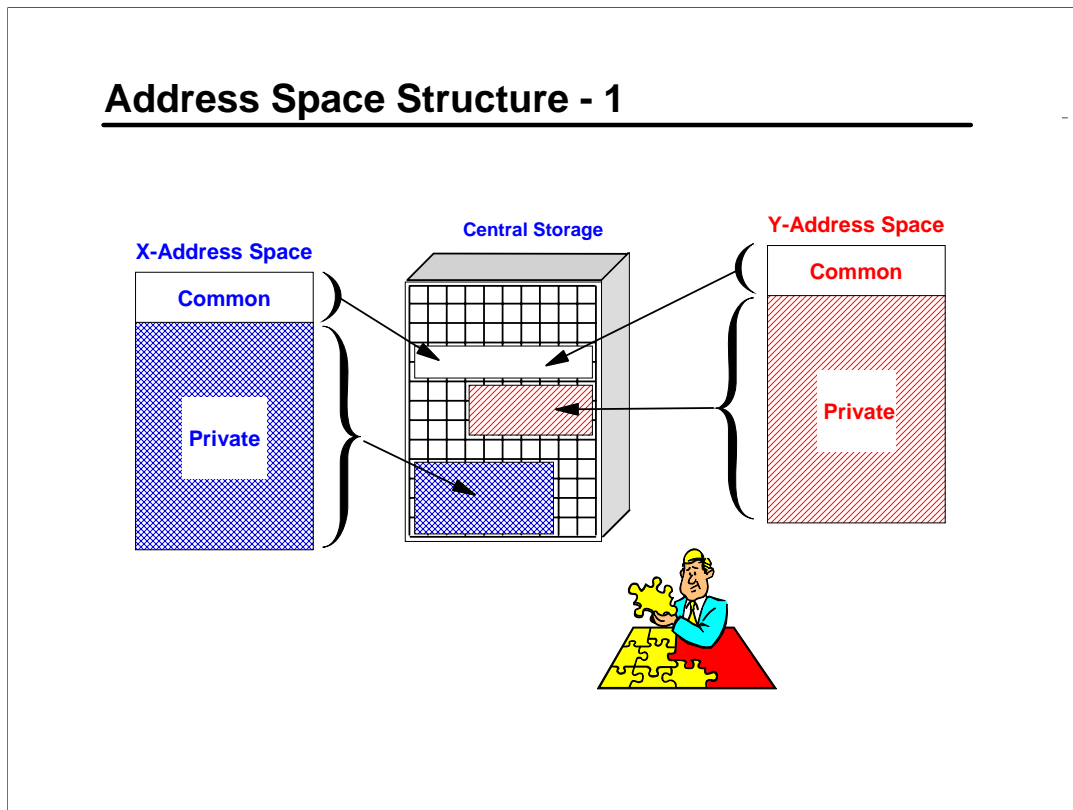


The mapping of virtual to real storage addresses is accomplished via page tables. Each address space has its own page tables. Page tables are built when an address space is created. It is then maintained by the Real Storage Manager as pages are created and moved between central and auxiliary storage.



So far we have looked at paging, which is the movement of inactive pages of programs and data between central storage and auxiliary storage. Swapping is the movement of the entire working set of an address space. Swapping is a common operation in the system and is mainly used to balance the use of resources. An address space is normally swapped out when it is known that it will be inactive for some time, e.g. a TSO address space which has sent output to the terminal and which is waiting for terminal input. Address spaces will also be swapped out when contention for system resources exceeds installation defined thresholds. The System Resource Manager (SRM) component of OS/390 determines the need for a swap out and selects the appropriate address space.

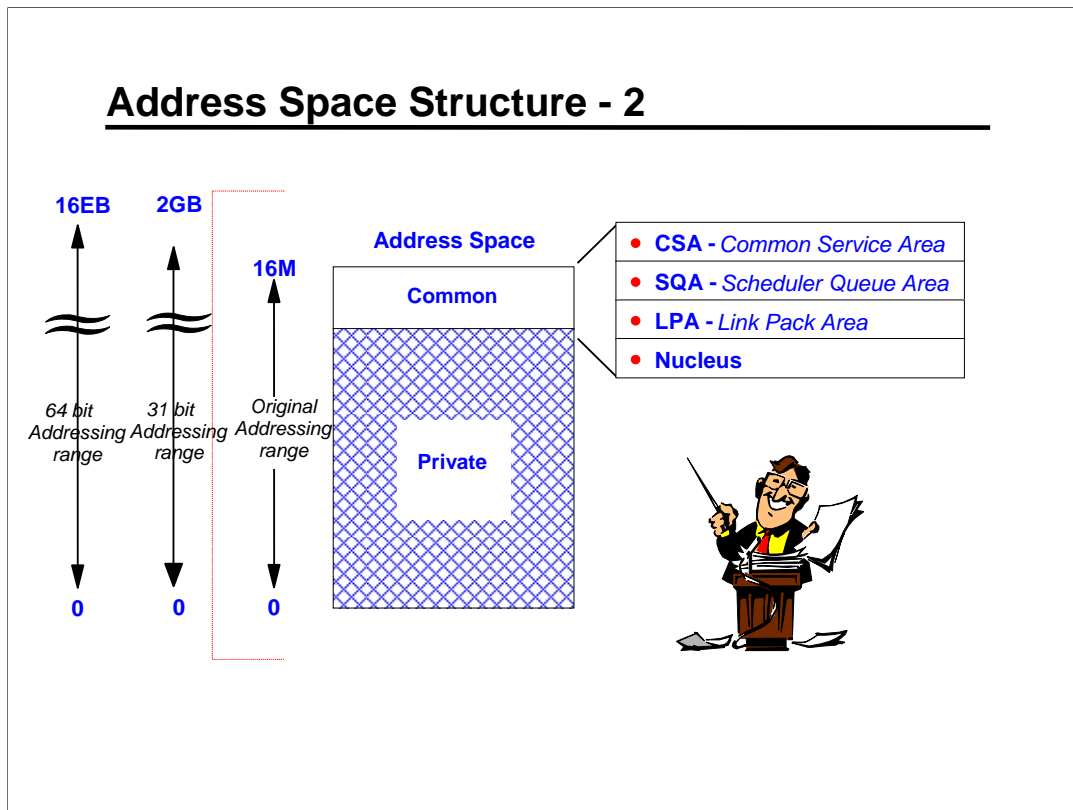
Both paging and swapping are essential to the operation of z/OS. However excessive paging and swapping can be detrimental to the smooth running of the system. The performance of important address spaces, such as system address spaces, CICS, IMS and DB2 would be drastically affected if they were swapped out. Consequently such address spaces will/should be set as non-swappable.



An address space is made up of two main areas:- The Private Area and the Common Area. The private area contains the program(s), data buffers and control information for the job/task running in the address space.

The common area maps the executable z/OS code, control blocks and work areas needed by ALL address spaces in the system. The data and programs in the private area of one address space cannot be accessed by any other address space (unless this has been specifically setup using 'cross-memory services').





The common area broken down into the areas shown.

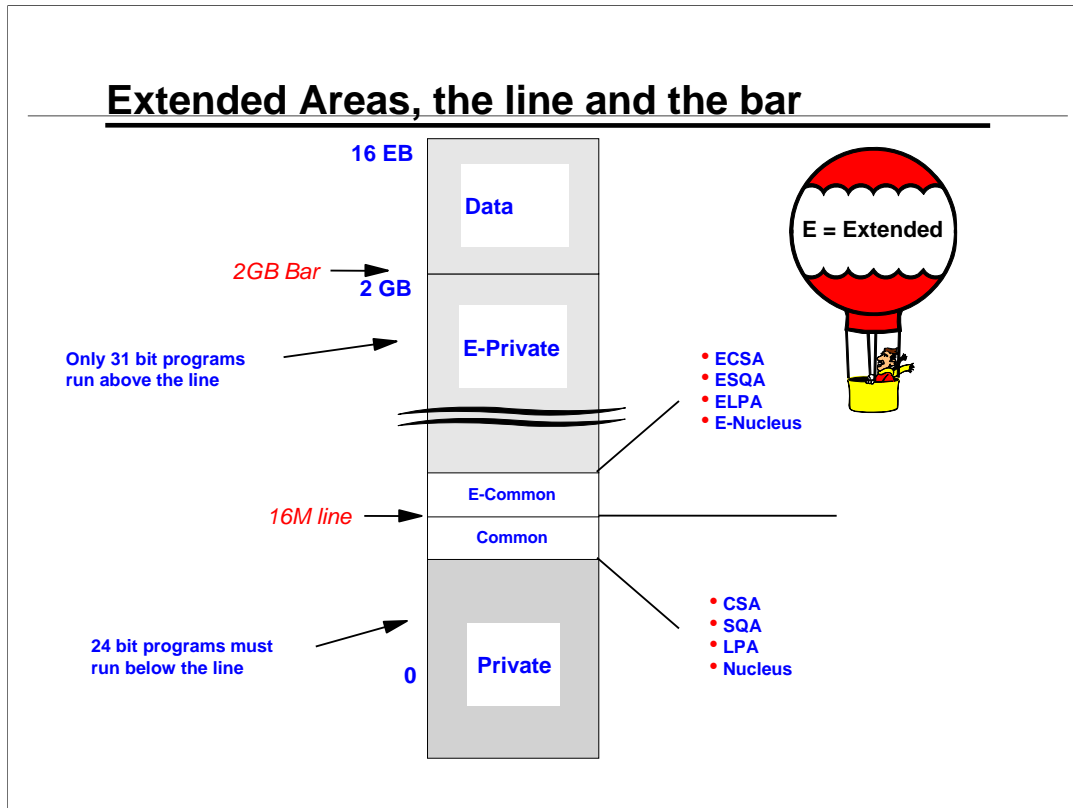
The Nucleus contains the heart of the z/OS operating system, although other parts are in the LPA and other system address spaces.

The Link Pack Area (LPA) contains program modules typically required by multiple address spaces, e.g. access methods, TSO commands.

The Scheduler Queue Area (SQA) contains system wide control blocks.

The Common Service Area (CSA) contains data required by more than one address space and is sometimes used as a communication area between one or more address spaces.

The original architected address range of an address space was 16 Megabytes. The current virtual address range is 2 Exabytes.

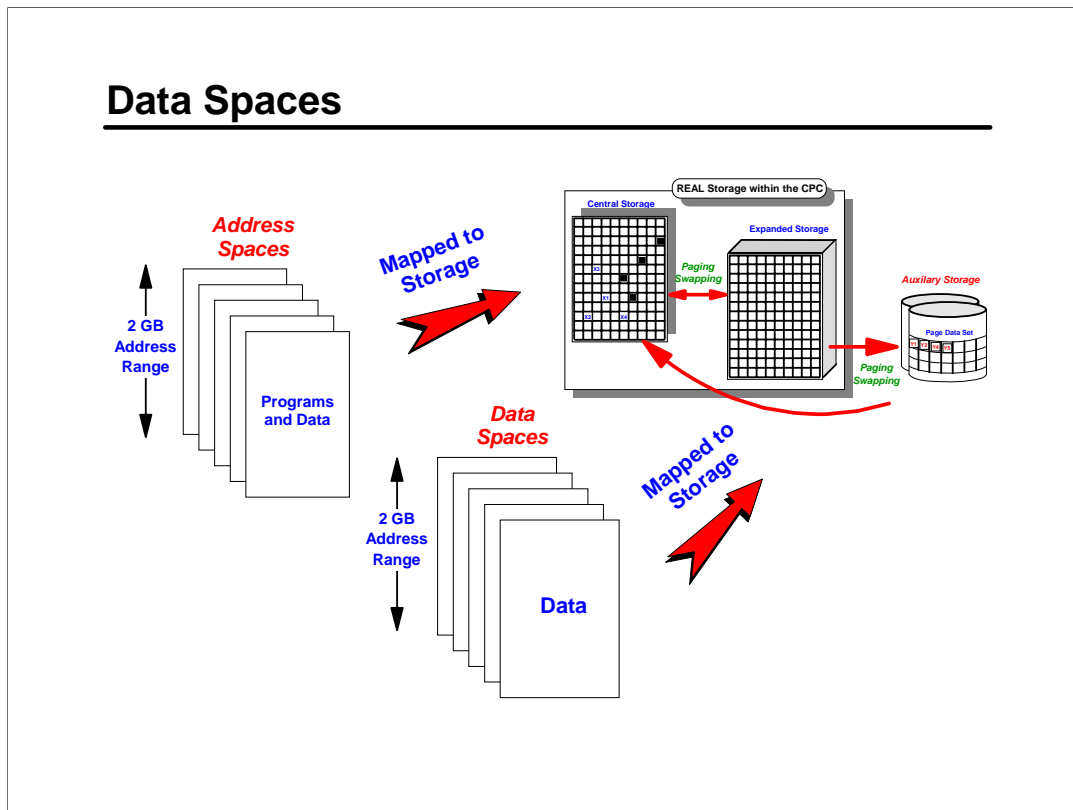


When the virtual storage addressing range was increased from 16 Megabytes to 2 Gigabytes the areas above the 16 Meg line were termed 'Extended'.

As the foil shows this applied to both the Private and Common areas. A program is designated as a 24-bit program if it can only access virtual storage addresses up to 16 megabytes. A 31-bit program can access virtual storage addresses up to 2 gigabytes. The addressing mode of a program (i.e. 24 or 31 bit) is set when the program source code is compiled. Programs written and compiled before 31 bit addressing was available/supported default to 24 bit programs and so will only run 'below the line'. When the virtual storage addressing range was increased from 16 Megabytes to 2 Gigabytes the areas above the 16 Meg line were termed 'Extended'.

As the foil shows this applied to both the Private and Common areas. A program is designated as a 24-bit program if it can only access virtual storage addresses up to 16 megabytes. A 31-bit program can access virtual storage addresses up to 2 gigabytes. The addressing mode of a program (i.e. 24 or 31 bit) is set when the program source code is compiled. Programs written and compiled before 31 bit addressing was available/supported default to 24 bit programs and so will only run 'below the line'.

64bit is now supported on z/OS, this brings about a new concept of "above the bar", which allows user programs to address memory up to 16 Exabytes (16 EB). Only data can be stored above the bar currently.



One of the ways to improve system performance is to get as much data in processor storage in order to avoid I/O operations. Data Spaces help in this area. Data spaces are horizontal storage expansions of the virtual storage concept. Data spaces, just like address spaces, are virtual storage areas but only contain data, not programs. A data space can be up to 2 GB in size and be used by one or multiple address spaces.

The entire addressing range in a data space is available for data. System areas are not mapped in a data space. Although a program may be 'stored' in a data space it must be moved to an address space for it to be executed. Data spaces are used by system address spaces such as VTAM, TCP/IP, RACF, VLF. VLF is in fact an address space designed to hold data in data spaces on behalf of other address spaces.

## Display List of Address Spaces (D A,L)

```

D A,L
IEE114I 21.45.20 96.081 ACTIVITY 566
JOBS      M/S      TS USERS  SYSAS  INITS  ACTIVE/MAX VTAM  OAS
00000    00008    00002    00019  00017  00002/00050  00000
  LLA      LLA      LLA      NSW S  VLF      VLF      VLF      NSW S
  APPC     APPC     APPC     NSW S  ASCH     ASCH     ASCH     NSW S
  ESCM     E          PSTEP01  OWT S  NET      NET      VTAM     NSW S
  JES2     JES2     IEFPROC  NSW S  TSO      TSO      STEP1    OWT S
  RSK      OWT      STEW     IN

```

```

IN = Swapped in
OUT = Swapped out, ready to run
OWT = Swapped out, waiting, not ready to run
OU* = In the process of being swapped out
IN* = In the process of being swapped in
NSW = Non-swappable

A = ATX
J = Job
M = Mount
S = Started task
* = System address space

```

The D A,L command will list the address spaces currently in the system.

JOBS - The number of address spaces running under initiators.

M/S - The number of address spaces created by a MOUNT or START command, but not an initiator.

TS USERS - The number of active Time Sharing Option Extensions (TSO/E) address spaces.

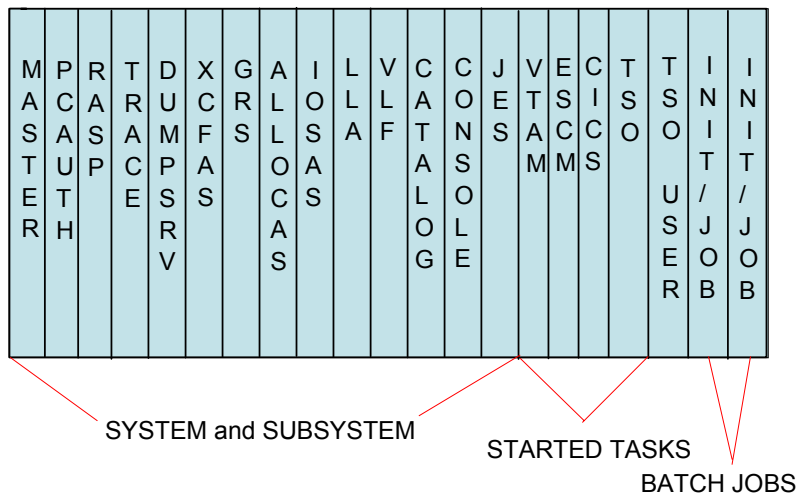
SYSAS - The number of system address spaces.

INITS - The number of started job entry subsystem (JES) and advanced program-to-program communications (APPC) initiators in the system.

ACTIVE/MAX VTAM - The number of active TSO/E address spaces using the Virtual Telecommunications Access Method (VTAM) and the maximum number of TSO/E address spaces that could use VTAM.

OAS - The total number of OpenMVS address spaces on this system.

## Types of Address Spaces



There are five different types of address spaces in z/OS. Each type is defined by the type of work performed and how the address space was created.

System address spaces - created by z/OS when the system is initialized.

Subsystem address spaces - created by z/OS during system initialization on the start command (JES, VTAM).

Started task programs are created using the START command.

TSO user address spaces are created at logon time.

Batch job address spaces are created with the START INIT command.

System and subsystem address space have more authority and capabilities than other address spaces.

## Display Address Space - Detailed Info

**D A, VTAM**

```

IEE115I 12.09.12 2007.150 ACTIVITY 667
JOBS  M/S  TS  USERS  SYSAS  INITS  ACTIVE/MAX  VTAM
00022 00033 00002 00035 00048 00002/00090 00026
VTAM  VTAM  VTAM  NSW  SO  A=0023  PER=NO  SMC=000
      PGN=N/A  DMN=N/A  AFF=NONE
      CT=21.56.21  ET=04681.04
      WUID=STC08374  USERID=SYSTASK
      WKL=OVERHEAD  SCL=STCFast  P=1
      RGP=N/A  SRVR=NO  QSC=NO
      ADDR SPACE  ASTE=07EC08C0
      DSPNAME=ISTAB37B  ASTE=635C8100
      DSPNAME=ISTF0D42  ASTE=7EB6DF80
      DSPNAME=IST08B3C  ASTE=7EB6D700
      DSPNAME=ISTFA316  ASTE=08CC4B80
      DSPNAME=IST2E97A  ASTE=08CC4F80
      DSPNAME=IST64F67  ASTE=635C8180
      DSPNAME=ISTFC3C2  ASTE=7EB6D480
      DSPNAME=ISTB89CA  ASTE=07FE4D80
    
```

Summary details row

Address space details  
PGN =performance Group  
CT = CPU time used  
ET = Elapsed Time

Details of Data Spaces managed by this address space

The D A,A command will list all active address spaces.

- \* System Address Space
- S Started Task
- I Initiator Address Space
- A Address space identifier (ASID), in hexadecimal.
- O OMVS
  
- PER YES or NO - PER trap is active or not active in the address space
- SMC Number of outstanding step-must-complete requests.
- PGN Performance group number – these went in z/OS 1.4
- DMN Domain number
- AFF The identifier of a specific processor (None=job can run on any processor).
- CT The processor time used by the address space, including the initiator.
- ET Elapsed time for the address space.

See message IEE115I in the System Message and Codes manual for more information.

## **Summary**

---

- **An Operating System is made up from multiple components**
- **Virtual Storage allows computer storage to be used more effectively**
- **The Address Space is the basic unit of work**
- **There are three types of storage**
  - **Central Storage**
  - **Expanded Storage**
  - **Auxiliary Storage**

## Time for Lab Exercise

---

### **Exercise 6** Address Spaces



© Copyright IBM Corporation 2005

### **Summary of Labs**

Use SDSF to look at the address spaces on our system



# JES2 Introduction

IBM provides two Job Entry Systems also called JESs from which to choose: JES2 and JES3. In an installation that has only one MVS image, JES2 and JES3 perform similar functions. That is, they read jobs into the system, convert them to internal machine-readable form, select them for processing, process their output, and purge them from the system.

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

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

JES2 controls output devices such as local and remote printers, punches, and card readers. You define each device to JES2 using JES2 initialization statements. All are directly under JES2's control with the exception of those printers that operate under the Print Services Facility (PSF). Printed and punched output can be routed to a variety of devices in multiple locations. The control JES2 exercises over its printers ranges from the job output classes and job names from which the printer can select work to such specifications such as the forms on which the output is printed.

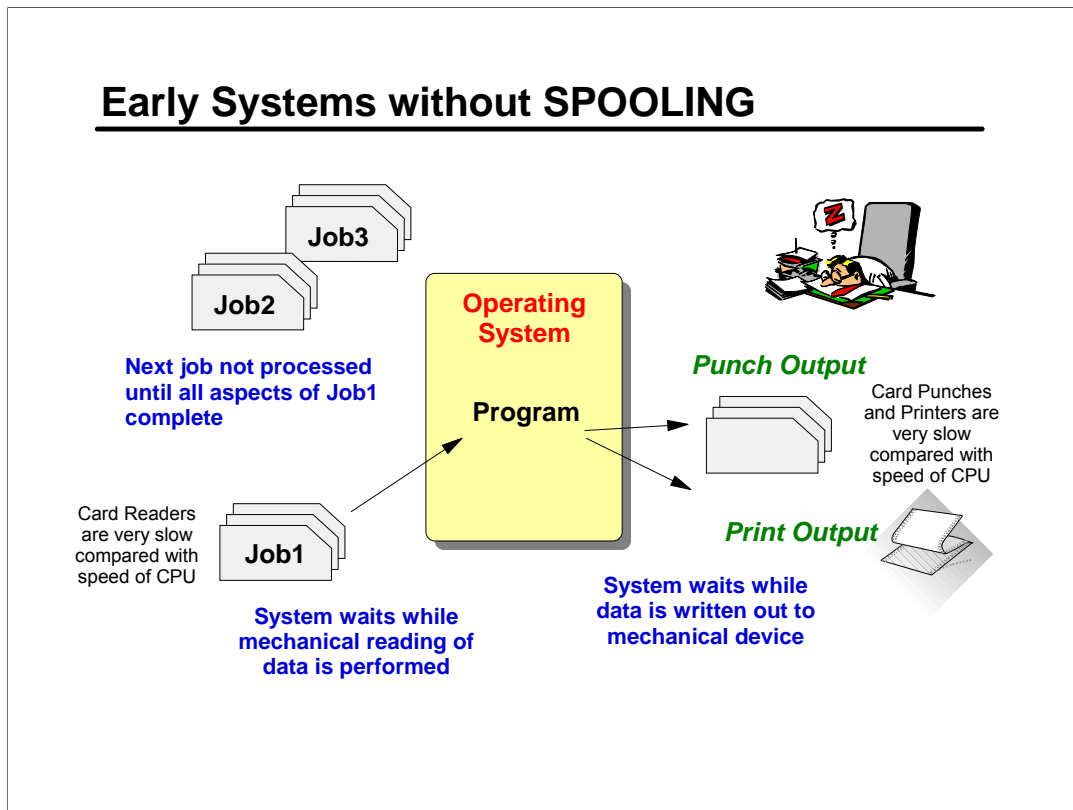
Card readers and card punches are things of the past. However 'readers' are still used with JES2 and these are called internal readers. An internal reader is a program that other programs can use to submit jobs, control statements, and commands to JES2. Any job running in MVS can use an internal reader to pass an input stream to JES2. JES2 can receive multiple jobs simultaneously through multiple internal readers. MVS uses internal readers, allocated during system initialization, to pass to JES2 the job control language (JCL) for started tasks, START and MOUNT commands, and TSO LOGON requests.

## **Objectives**

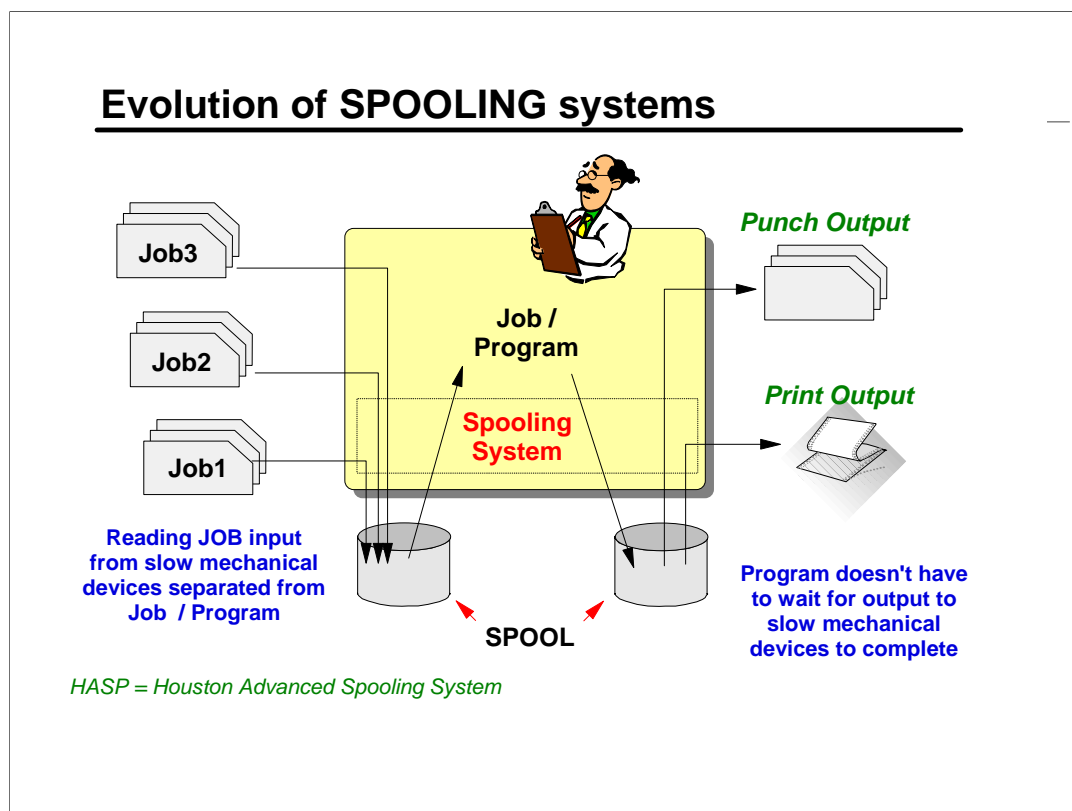
---



- Describe the various modes of operation
- Describe the main JES2 components and operation
- Describe the various modes of operation



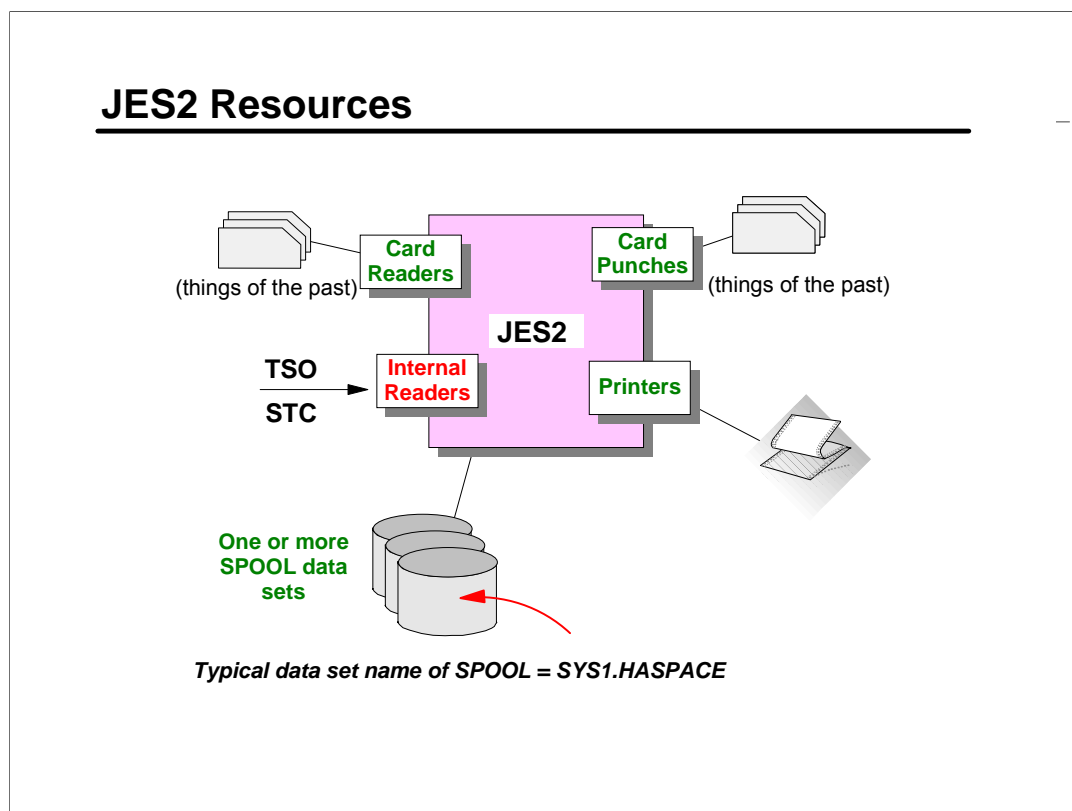
This foils shows the problems associated with very early systems before any spooling system was available.



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

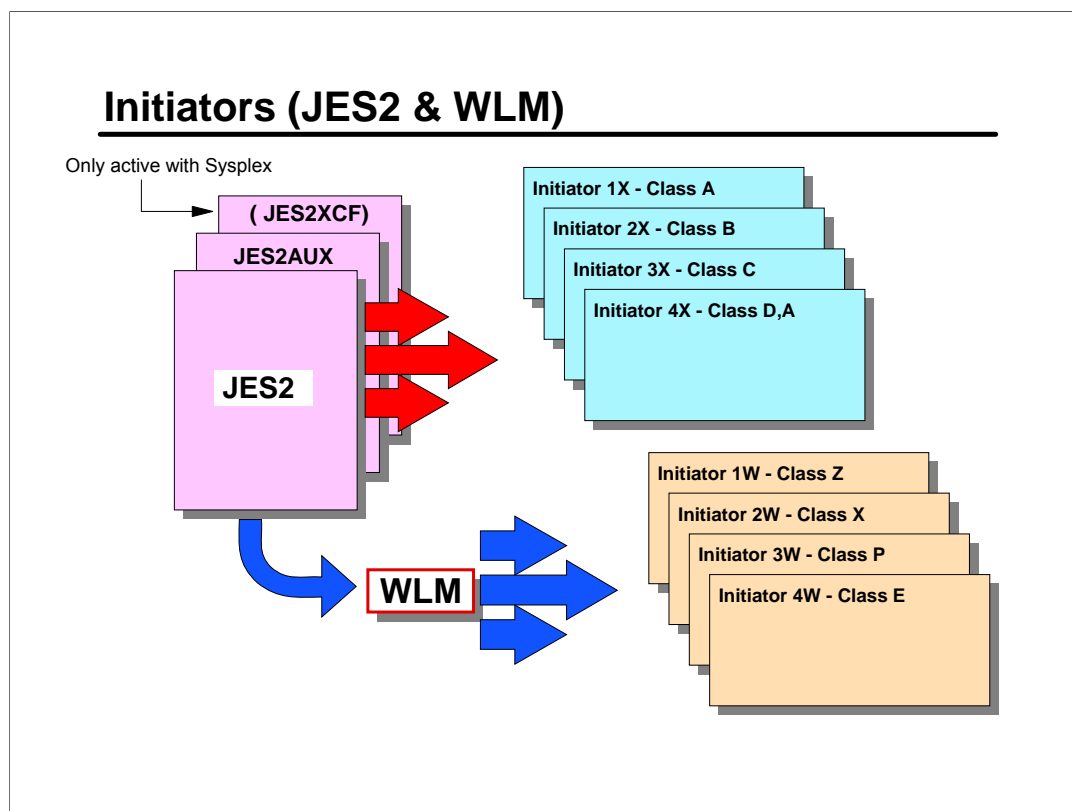
JES2, or Job Entry Subsystem 2, is a functional extension of the HASP II program that receives jobs into the system and processes all output data produced by the job. So what does all that mean? Simply stated, JES2 is the component of z/OS that provides the necessary functions to get jobs into, and output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

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



JES2 controls output devices such as local and remote printers, punches, and card readers. You define each device to JES2 using JES2 initialization statements. All are directly under JES2's control with the exception of those printers that operate under the Print Services Facility (PSF). Printed and punched output can be routed to a variety of devices in multiple locations. The control JES2 exercises over its printers ranges from the job output classes and job names from which the printer can select work to such specifications such as the forms on which the output is printed.

Card readers and card punches are things of the past. However 'readers' are still used with JES2 and these are called internal readers. An internal reader is a program that other programs can use to submit jobs, control statements, and commands to JES2. Any job running in MVS can use an internal reader to pass an input stream to JES2. JES2 can receive multiple jobs simultaneously through multiple internal readers. MVS uses internal readers, allocated during system initialization, to pass to JES2 the job control language (JCL) for started tasks, START and MOUNT commands, and TSO LOGON requests.

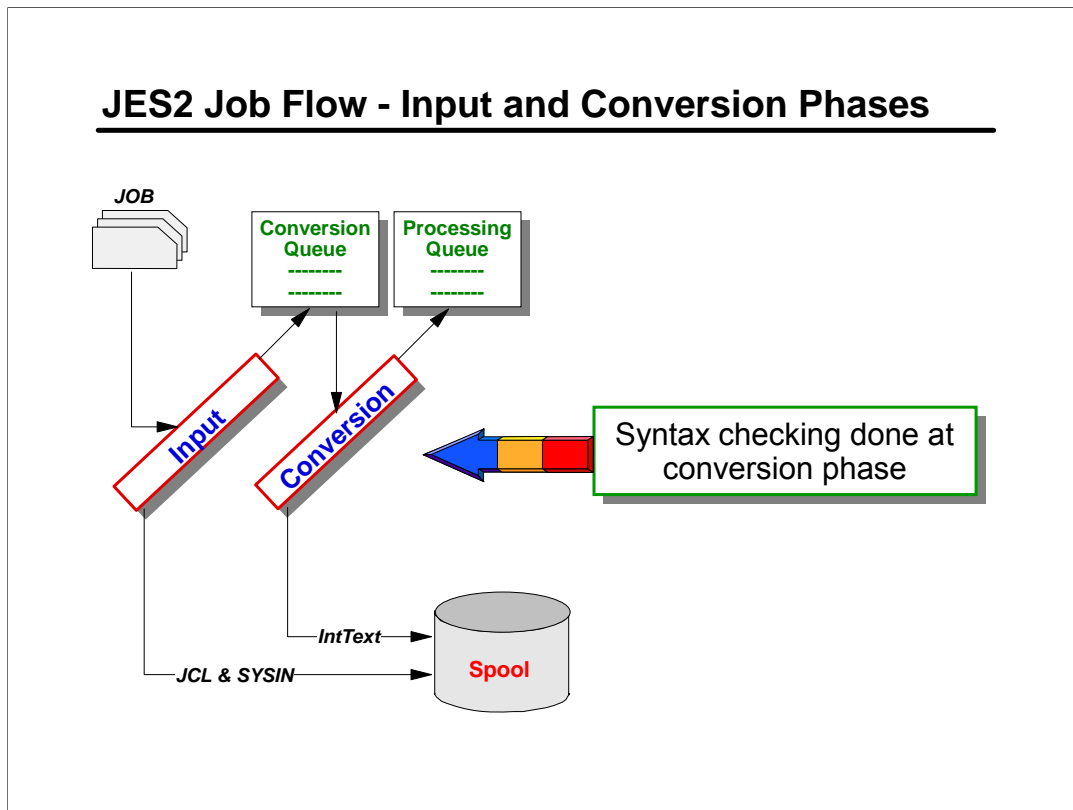


Batch jobs do not run in the JES2 address space but in other 'worker' address spaces called initiators. There are two types of initiators; traditional JES2 controlled initiators and the more recent WLM managed initiators.

An initiator is a system program that runs in its own address space and is controlled by JES or by WLM (in goal mode, with WLM Batch Initiator Management). JES2 initiators are defined to JES2 through JES2 initialization statements. JES2 initiators are started by the operator or by JES2 automatically when the system initializes. The installation associates each initiator with one or more job classes in order to obtain an efficient use of available system resources. Initiators select jobs whose classes match the initiator assigned class, obeying the priority of the queued jobs.

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

Jobs are routed to WLM initiators via a JOBCLASS JES2 initialization statement. In goal mode, with WLM Batch Management, a system can have WLM initiators and/or JES2 initiators.



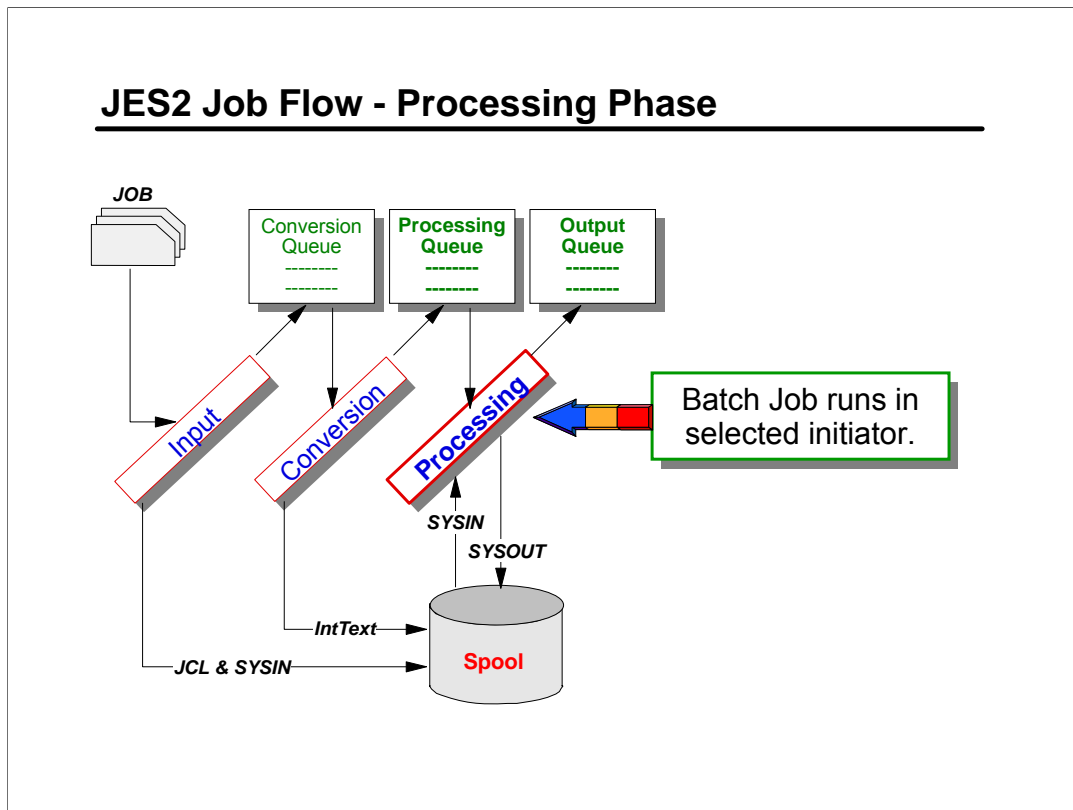
There are six JES2 phases, the first of which are the Input and Conversion Phases.

#### Input phase

JES2 accepts jobs, in the form of an input stream, from input devices, internal readers, other nodes in a job entry network, and from other programs. JES2 reads the input stream and assigns a job identifier to each JOB JCL statement. JES2 places the job's JCL, optional JES2 control statements, and SYSIN data onto the spool data sets. JES2 then selects jobs from the spool data sets for processing and subsequent running.

#### Conversion phase

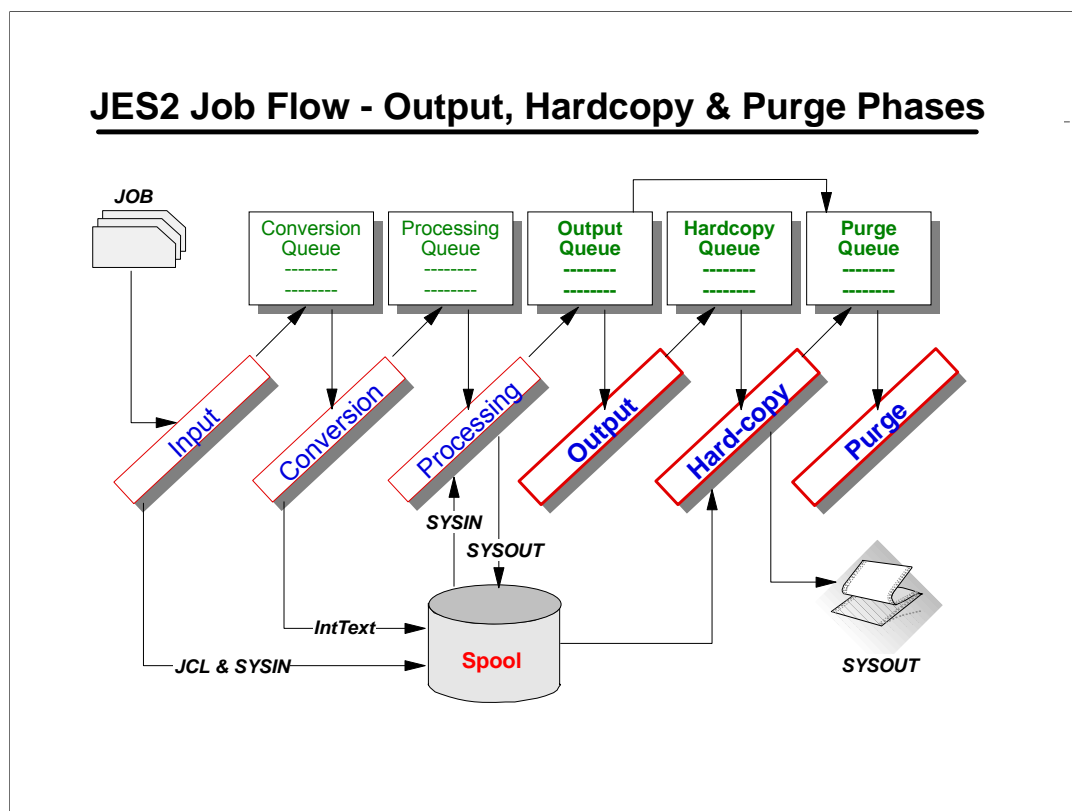
JES2 uses a converter program to analyze each job's JCL statements. The converter takes the job's JCL and merges it with JCL from a procedure library. Then, JES2 converts the composite JCL into converter/interpreter text that both JES2 and the job scheduler functions of MVS can recognize. Next, JES2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



In the processing phase, JES2 responds to requests for jobs from the MVS initiators. After a job is selected, the initiator invokes the interpreter to build control blocks from the converter/interpreter text that the converter created for the job. The initiator then allocates the resources specified in the JCL for the first step of the job. This allocation ensures that the devices are available before the job step starts running. The initiator then starts the program requested in the JCL EXEC statement.

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





#### Output phase

JES2 controls all SYSOUT processing. SYSOUT is a data set in a printer device format, that is, it is ready to be printed. Intermediately, a sysout file is stored in the spool data set. System messages related to job execution are directed to SYSOUT.

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

#### Hardcopy phase

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

#### Purge phase

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

## SDSF interface to JES2

```

S2 - [24 x 80]
File Edit View Communication Actions Window Help
Display Filter View Print Options Help
-----
HQX7708 ----- SDSF PRIMARY OPTION MENU -----
COMMAND INPUT ==>> _                               SCROLL ==>> CSR

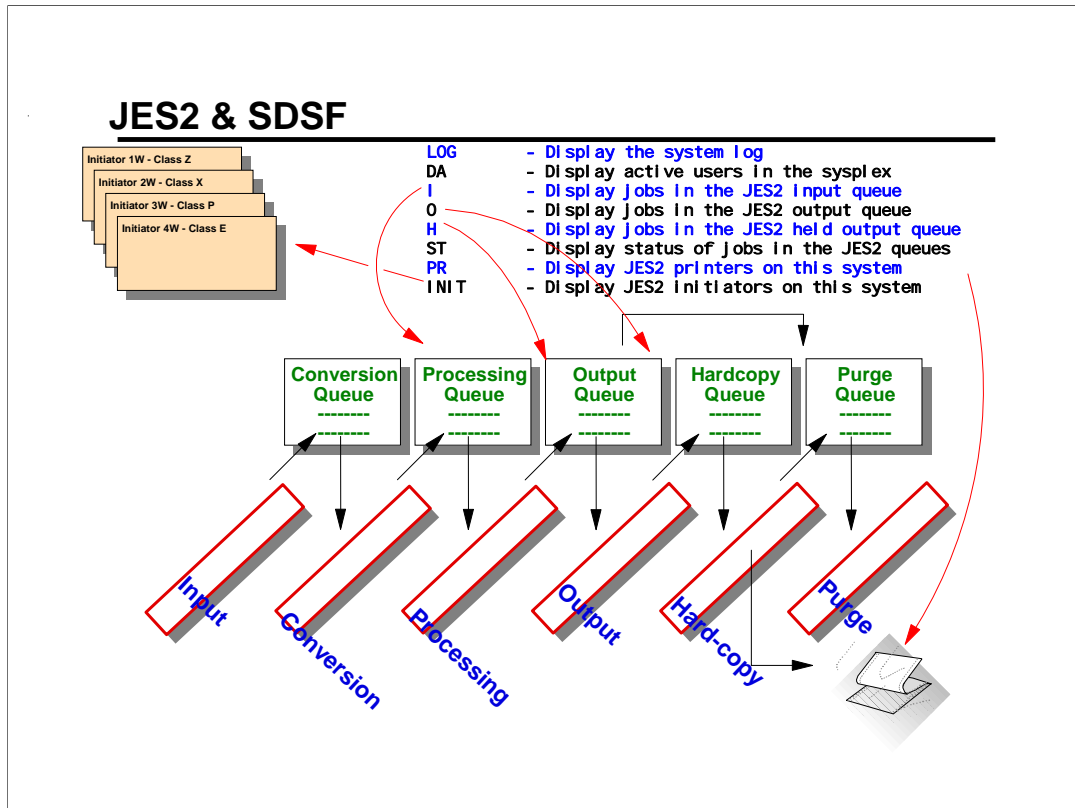
DA   Active users                                INIT  Initiators
I    Input queue                                 PR    Printers
O    Output queue                                PUN   Punches
H    Held output queue                           RDR   Readers
ST   Status of jobs                              LINE  Lines
                                           NODE  Nodes
LOG   System log                                 SO    Spool offload
SR   System requests                             SP    Spool volumes
MAS  Members in the MAS
JC   Job classes                                CK    Health checker
SE   Scheduling environments
RES  WLM resources                              ULOG  User session log

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1981, 2003. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

M.A. 04/021
Connected to remote server/host winmvss2 using lu/pool IYAVTC27 and port 23

```



This foil shows the approximate relationship of the SDSF panel options to the JES2 queues and resources.

A job will be in the input queue for one or more of the following reasons:

The person who submitted the job wanted the job to be held until released by operations. This is specified by the TYPRUN=HOLD parameter on the JOB card.

All initiators supporting the jobclass of the job request are all busy

There is no initiator started that supports the job class requested by the job.

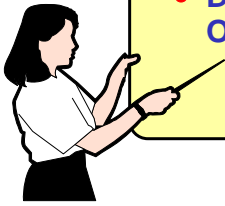
Print output for a specific job will be in the Held output queue because a held output class was requested on the job, which may be because the held output class is the default.

Print output for a specific job will be in the Output queue when it is waiting for a specific printer, or printer setup.

## Summary

---

- Described the purpose of JES2
- Described the main JES2 components and operation
- Described the various modes of Operation for JCL



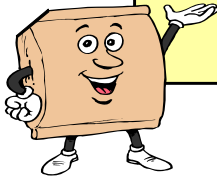
---

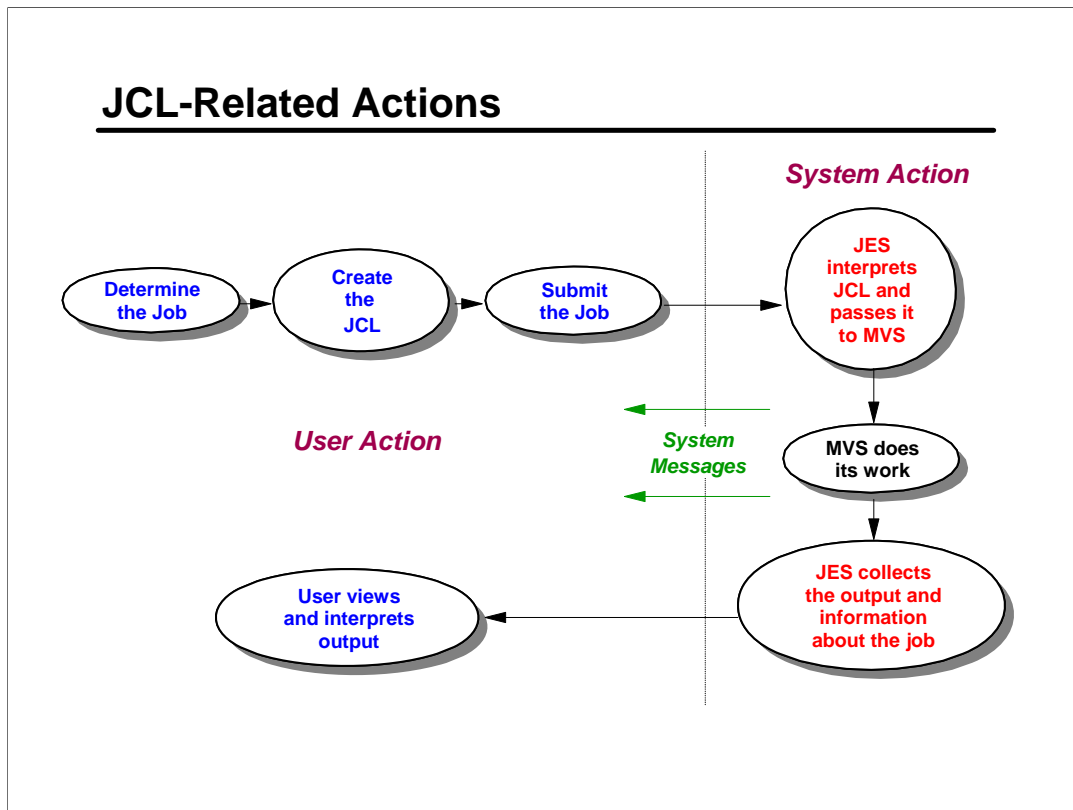
# **Introduction to JCL**

## Objectives

---

- Describe the concepts of JCL and syntax
- Describe the JOB, EXEC and DD statements
- Code DD statements using specified information



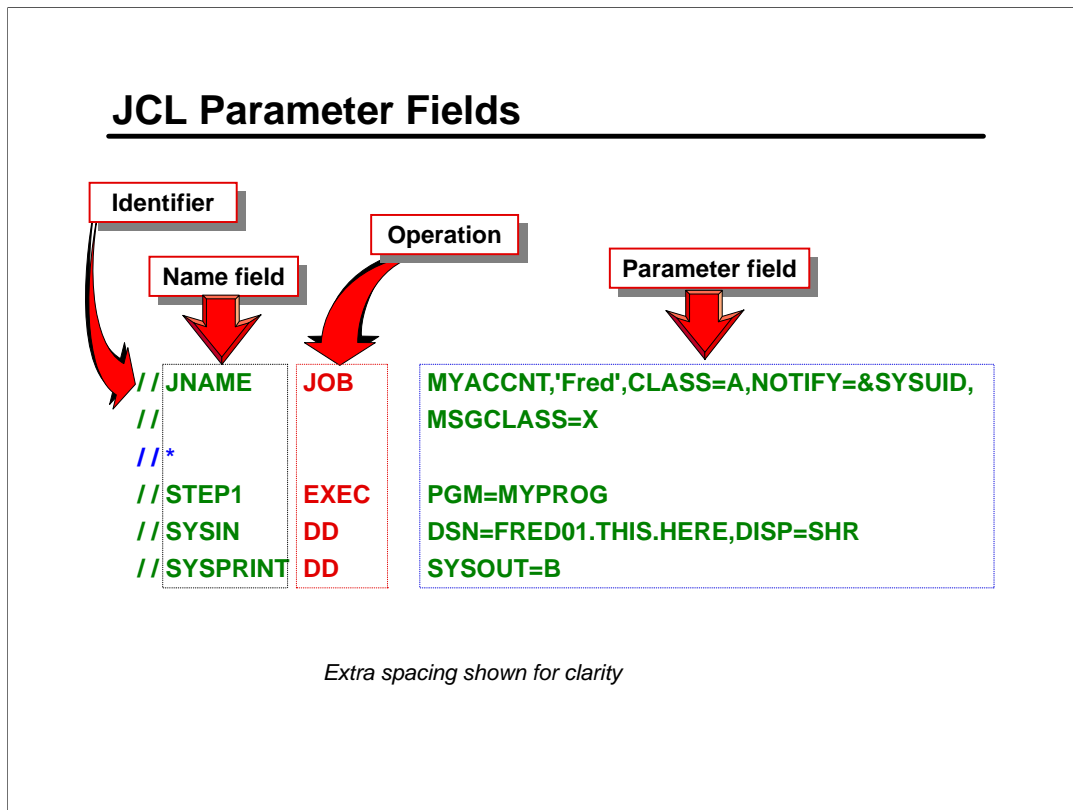


Job Control Language (JCL) is used to tell the system what program to execute, followed by a description of program inputs and outputs.

It is possible to submit JCL for batch processing or start a JCL procedure (PROC), which is considered a started task.

The details of JCL can be complicated but the general concepts are quite simple. Also, a small subset of JCL accounts for at least 90% of what is actually used.

While application programmers need some knowledge of JCL, the production control analyst responsible must be highly proficient with JCL, to create, monitor, correct and rerun the company's daily batch workload.



The identifier field tells the system that the statement is

- a JCL statement (contains // )
- a delimiter statement (contains /\*)
- a Comment statement (contains /\*\*)

The Name field identifies a control statement that other statements and the system can reference. The name field.

- must begin in column 3
- 1 to 8 alphanumeric (A-Z, 0-9) or national characters (#,@,£)
- first character must be alphabetic or national
- name field must be followed by at least one blank

The Operation field specifies the type of control statement that

- follows the name field
- must be preceded and followed by at least one blank
- can contain JOB, EXEC and DD statements as well as those listed in the next few pages.

The Parameter field contains one or more parameters separated by commas

- follows the operation field
- NO blanks between parameters
- use commas to separate parameters
- preceded and followed by at least one blank
- parameters can be coded up to position 71



## Keys SYNTAX points - JCL statement

- Consists of one or more records 80 bytes in length.
- Is coded from column 1 to column 71, in uppercase (except for USS).
- Starts with a // at the beginning of the line.
- A commentary line begins with a /\*.
- A line with only // and blanks indicates end of job.
- A comma indicates that the statement has continuation.
- A continuation of a statement from a previous line must start from column 4 to 16.
- Comments must be separated from operators / parameters by at least one blank.

## JOB, EXEC & DD statements

Every job must contain the two following types of control statements

- JOB
- EXEC

Also, jobs usually contain

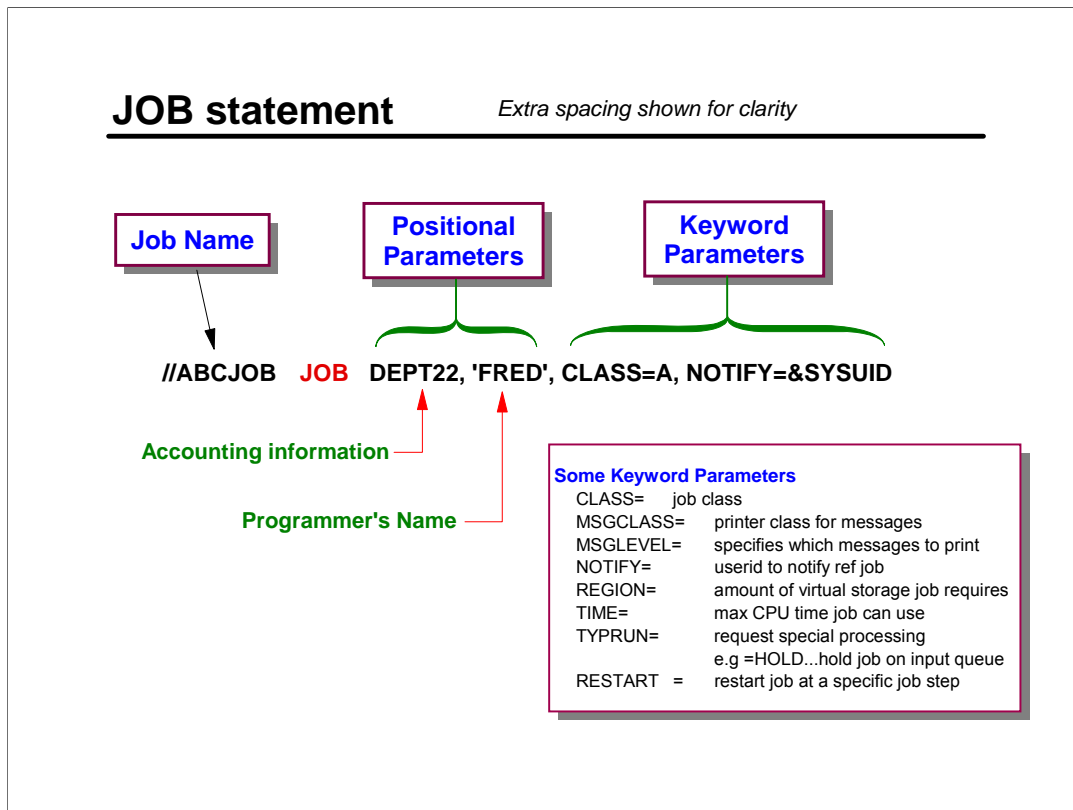
- One or more DD statements



In some installations users might not need a JOB card as this may be automatically included. However this is done, a JOB card must always be part of the JCL for a batch job.

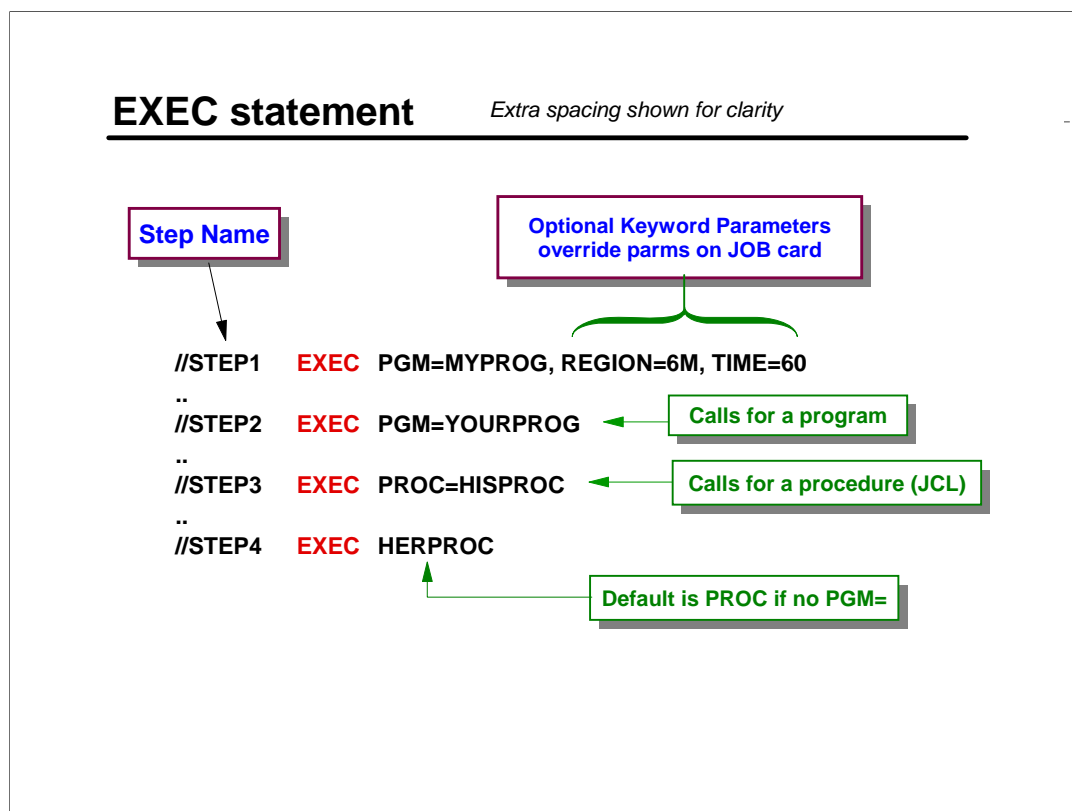
The JOB and EXEC statements must always be used, and the DD statement is used in most JCL. Consequently this topic will concentrate on the use of these statements. The full set of control statements are listed below.

```
// COMMAND   Specifies an MVS or JES command that the system issues when the JCL is converted.
// *         Contains comments. The comment statement is used primarily to
document a program and its resource requirements.
// CNTL      Marks the beginning of one or more program control statements.
// DD        Identifies and describes a data set.
// *         Indicates the end of data placed in the input stream.
// ENDCTL    Marks the end of one or more program control statements.
// EXEC      Marks the beginning of a job step; assigns a name to the step; identifies the program or
the cataloged or in-stream procedure to be executed in this step.
// IF/THEN/ELSE/ENDIF IF/THEN/ELSE/ENDIF   Specifies conditional execution of job steps within a
job.
// INCLUDE   Identifies a member of a partitioned data set (PDS) or partitioned data
set extended (PDSE) that contains JCL statements to be included in the job stream.
// JCLLIB    Identifies the libraries that the system will search for.
// JOB       Marks the beginning of a job; assigns a name to the job.
//           Marks the end of a job.
// OUTPUT    Specifies the processing options that the job entry subsystem is to use for printing a
sysout data set.
// PEND      Marks the end of an in-stream or cataloged procedure.
// PROC      Marks the beginning of an in-stream procedure and may mark the beginning of a
cataloged procedure; assigns default values to parameters defined in the procedure.
// SET       Defines and assigns initial values to symbolic parameters used when processing JCL
statements. Changes or nullifies the values assigned to symbolic parameters.
```



Many of the parameters on the JOB statement are optional. Should they be omitted, defaults are taken according to values specified when JES was initialised.

There are two types of parameters; positional and keyword. As the name suggests, positional parameters must be coded in the order in which they are expected. Keyword parameters on the other hand can be coded in any position after the positional parameters.



Each job step begins with an EXEC statement that either names the program to execute for invokes a catalogued or in-stream procedure. An EXEC statement is required for all job steps. There can be up to 255 job steps in a single job.

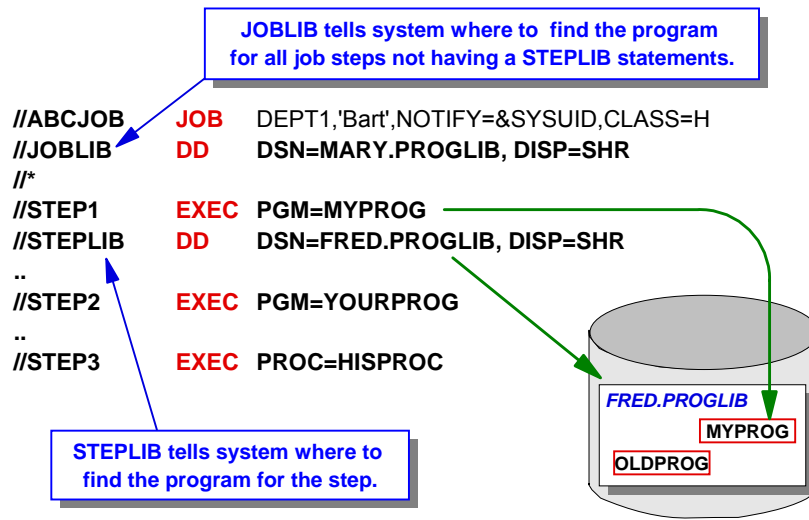
The parameter immediately following EXEC can be the keyword parameters PGM=xxxx (as in STEP1 and STEP2) or PROC=yyyy (as in STEP3) or the name of a procedure (as in STEP4). Note that if neither keyword statements are used the system assumes that the name following the EXEC word is the name of a procedure.

A procedure is essentially a file containing other JCL.

Many of the parameters used on the JOB statement can be used on the EXEC statement, as shown in STEP1. When coded on the EXEC statement they override those on the JOB statement

## STEPLIB and JOBLIB

*Extra spacing shown for clarity*



## Program Search Sequence

- STEPLIB
- JOBLIB
- LPA
- SYS1.LINKLIB
- (LINKLIST concatenation)

SYS1.PARMLIB(PROGxx)

-----  
 -----  
 -----

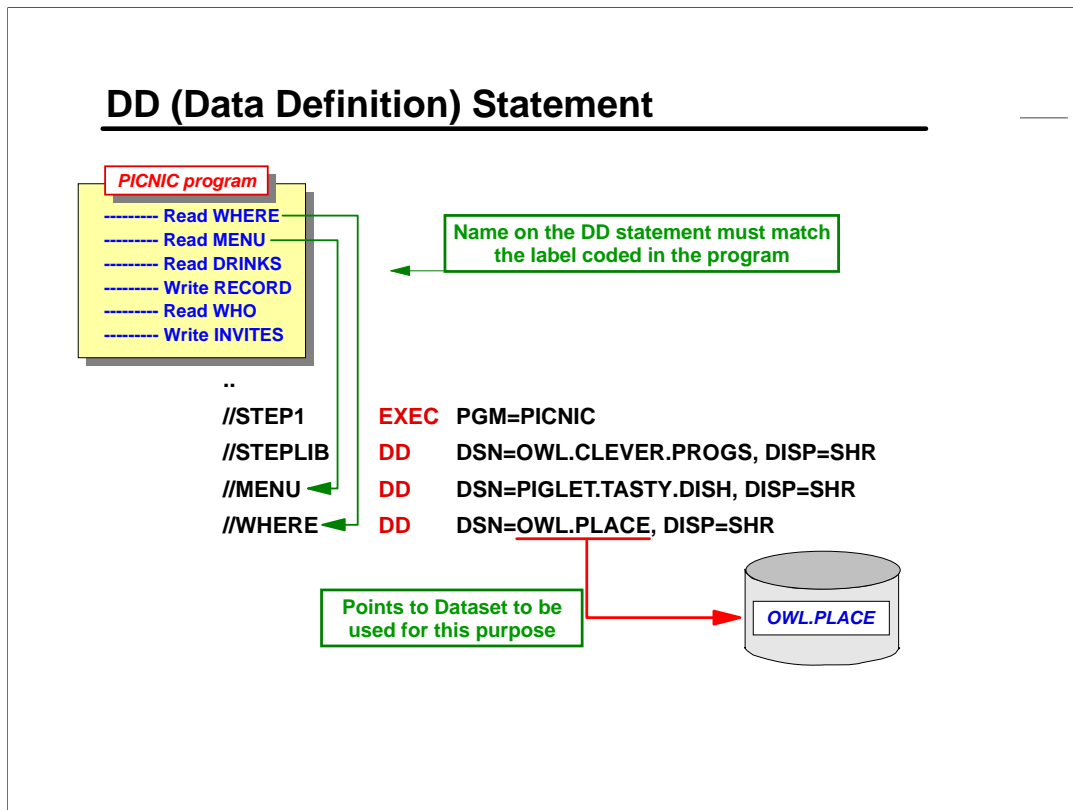
Lists of other  
program libraries



This foils shows the search order used by the system to locate a particular program. If a STEPLIB statement is coded for a particular step then the program library referenced will be searched. If there is no STEPLIB but there is a JOBLIB, then that library will be searched. If there is no STEPLIB or JOBLIB the system first looks in LPA for the program. If it is not found there then it will look in SYS1.LINKLIB and other program libraries defined in the LINKLIST.

Many programs that are executed on the system very often come from a group of system libraries call the LINKLIST. This list of libraries is available for all address spaces in the system. The LINKLIST is normally the largest source of programs for the system. Here your system programmer defines the program libraries that are accessible by all users, without having to code STEPLIB or JOBLIB statements.

The LINKLIST (LNKLST) concatenation is established at IPL time and much of the information about the location of programs in the LINKLIST can be cached to improve search time.



In a program, rather than hard code specific values for data input and output, labels are used to make the program more flexible. In our example the PICNIC program has labels, as shown and whether they are to be used for input or output. The connection between an MVS dataset and the program is achieved by coding the appropriate DD name and dataset name to be used.

To reach a dataset, programs begin with a request to MVS to open the dataset for Input and/or Output. This results in a number of preparations for input/output. Among other things some storage is reserved for buffer areas. To do this MVS needs to know what the data looks like, and it gets this information from the DCB.

The DCB (Device Control Block) is a number of bytes in storage contained in the user's program. The programmer may or may not put all information of logical record length, blocksize, etc: in the DCB at processing time. The DCB name must however contain the name of the JCL statement (DD) used to describe the dataset and its location.

## DD Statement - DISP parameter

The **DISP** parameter specifies how the dataset will be opened.  
i.e. for Read, Update, Append or Create

```
//STEP1      EXEC  PGM=PICNIC
//STEPLIB    DD   DSN=OWL.CLEVER.PROGS, DISP=SHR
//MENU       DD   DSN=PIGLET.TASTY.DISH, DISP=SHR
//WHERE      DD   DSN=OWL.PLACE, DISP=SHR
```

**DISP=NEW** (Create new dataset)  
**DISP=OLD** (Open for update)  
**DISP=MOD** (Open for append)  
**DISP=SHR** (Open for read)

If **DISP** is not specified, then **NEW** is the default

Open for  
Read

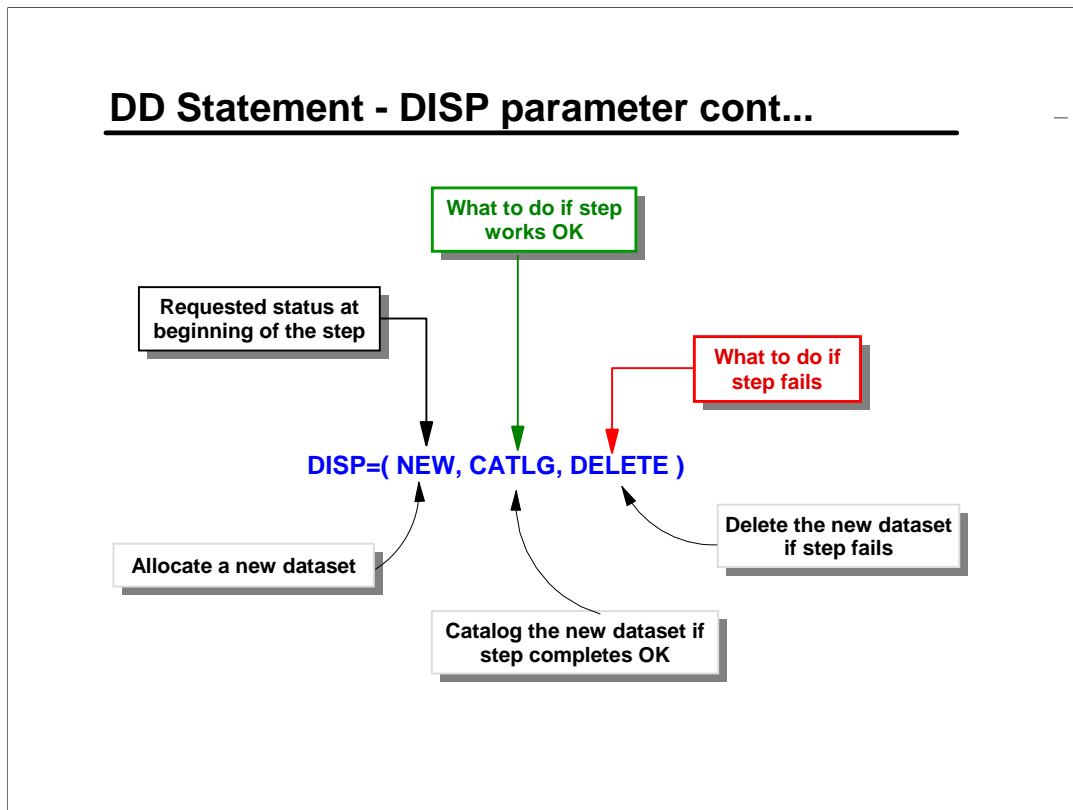


In a program, rather than hard code specific values for data input and output, labels are used to make the program more flexible. In our example the PICNIC program has labels, as shown and whether they are to be used for input or output. The connection between an MVS dataset and the program is achieved by coding the appropriate DD name and dataset name to be used.

To reach a dataset, programs begin with a request to MVS to open the dataset for Input and/or Output. This results in a number of preparations for input/output. Among other things some storage is reserved for buffer areas. To do this MVS needs to know what the data looks like, and it gets this information from the DCB.

The DCB (Device Control Block) is a number of bytes in storage contained in the user's program. The programmer may or may not put all information of logical record length, blocksize, etc: in the DCB at processing time. The DCB name must however contain the name of the JCL statement (DD) used to describe the dataset and its location.





All subparameters within the DISP parameter are positional. Where more than one subparameter is used they must be enclosed in open brackets.

The second subparameter can be:

DELETE	delete the dataset
KEEP	leave as it was at beginning of step
PASS	dataset will be used by following step, so let later step decide what to do.
CATLG	Catalog the dataset
UNCATLG	Uncatalog the dataset

The third subparameter can be the same as those above except PASS.

DISP parameter defaults are as follows:

No DISP	(NEW,DELETE,DELETE)
DISP=OLD	(OLD,KEEP,KEEP)
DISP=(,KEEP)	(NEW,KEEP,KEEP)
DISP=NEW	(NEW,DELETE,DELETE)
DISP=SHR	(SHR,KEEP,KEEP)

Note that you may see DISP=(,CATLG). This equates to (NEW,CATLG,CATLG)

## DCB, SPACE, UNIT and DCB parameters

*Extra spacing shown for clarity*

```

..
//STEP1      EXEC  PGM=PICNIC
//STEPLIB    DD    DSN=OWL.CLEVER.PROGS, DISP=SHR
//MENU       DD    DSN=PIGLET.TASTY.DISH, DISP=SHR
//WHERE      DD    DSN=OWL.PLACE, DISP=SHR
//RECORD     DD    DSN=OWL.RECORD, DISP=( NEW,CATLG ),
//            SPACE=( TRK, ( 1, 1 ) ), RECFM = FB, LRECL = 80,
//            BLKSIZE = 0, UNIT = SYSDA

```

Specifies dataset characteristics and where dataset should be placed



In a program, rather than hard code specific values for data input and output, labels are used to make the program more flexible. In our example the PICNIC program has labels, as shown and whether they are to be used for input or output. The connection between an MVS dataset and the program is achieved by coding the appropriate DD name and dataset name to be used.

To reach a dataset, programs begin with a request to MVS to open the dataset for Input and/or Output. This results in a number of preparations for input/output. Among other things some storage is reserved for buffer areas. To do this MVS needs to know what the data looks like, and it gets this information from the DCB.

The DCB (Device Control Block) is a number of bytes in storage contained in the user's program. The programmer may or may not put all information of logical record length, blocksize, etc: in the DCB at processing time. The DCB name must however contain the name of the JCL statement (DD) used to describe the dataset and its location.

When a new dataset is allocated the system must be told the type of dataset, its characteristics and where it should be placed. SYSDA is an 'esoteric' definition to which the systems programmer has defined various disk volumes for general use.

Other UNIT= parameters could be UNIT=3390, UNIT=3380, UNIT=CART, UNIT=TAPE, or other generics and esoterics defined by the installation.

In the example above, the space parameter specifies 1 track primary and 1 track secondary. If a partitioned dataset were to be allocated then the number of directory blocks would also have to be defined, e.g. SPACE=(TRK,(1,1,6)). Space can be requested in various units, such as tracks, cylinders, bytes, logical records and so on.

A blocksize of 0 indicates that the system should determine the optimum block size for the particular dataset.

There is also a DSNTYPE= parameter which specifies the type of dataset required, e.g. DSNTYPE=LIBRARY requests a PDS/E. If this is not specified then defaults will apply.

## Instream Data & DUMMY

*Extra spacing shown for clarity*

```

..
//STEP1 EXEC PGM=PICNIC
//STEPLIB DD DSN=OWL.CLEVER.PROGS, DISP=SHR
//MENU DD DSN=PIGLET.TASTY.DISH, DISP=SHR
//WHERE DD DSN=OWL.PLACE, DISP=SHR
//RECORD DD DSN=OWL.REC
// SPACE=( TRK, (
// BLKSIZE = 0, UN
//DRINKS DD DUMMY
//WHO DD *
EYEORE
RABBIT
CHRISTOPHER ROBIN
TIGGER
PIGLET
POOH
/*

```

DD DUMMY indicates to the program that this label is not required for this program run

DD \* tells the program that the information associated with the label 'WHO' is not in a dataset but listed in the JCL stream following

When a new dataset is allocated the system must be told the type of dataset, its characteristics and where it should be placed. SYSDA is an 'esoteric' definition to which the systems programmer has defined various disk volumes for general use.

Other UNIT= parameters could be UNIT=3390, UNIT=3380, UNIT=CART, UNIT=TAPE, or other generics and esoterics defined by the installation.

In the example above, the space parameter specifies 1 track primary and 1 track secondary. If a partitioned dataset were to be allocated then the number of directory blocks would also have to be defined, e.g. SPACE=(TRK,(1,1,6)). Space can be requested in various units, such as tracks, cylinders, bytes, logical records and so on.

A blocksize of 0 indicates that the system should determine the optimum block size for the particular dataset.

There is also a DSNTYPE= parameter which specifies the type of dataset required, e.g. DSNTYPE=LIBRARY requests a PDS/E. If this is not specified then defaults will apply.

## SYSOUT

*Extra spacing shown for clarity*

```

..
//STEP1 EXEC PGM=PICNIC
//STEPLIB DD DSN=OWL.CLEVER.PROGS, DISP=SHR
//MENU DD DSN=PIGLET.TASTY.DISH, DISP=SHR
//WHERE DD DSN=OWL.PLACE, DISP=SHR
//RECORD DD DSN=OWL.PLACE, DISP=( NEW,CATLG ),
//
//
//DRINK
//WHO CHR
      TIGG
      PIGLET
      POOH
/*
//INVITES DD SYSOUT=P

```

- **SYSOUT** is associated with print output.
- **SYSOUT=P** indicates that printout should go to output class P
- **SYSOUT=\*** indicates that print output should go to the same print class as Messages (defined in the MSGCLASS JOB parameter)

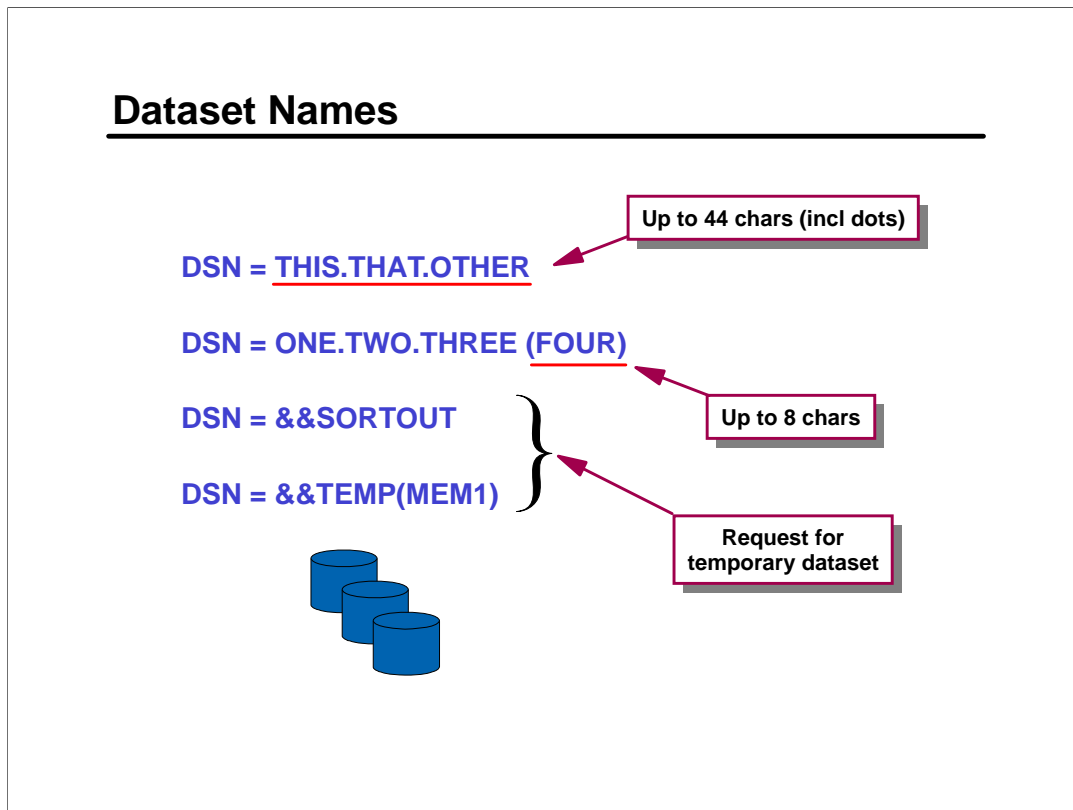
When a new dataset is allocated the system must be told the type of dataset, its characteristics and where it should be placed. SYSDA is an 'esoteric' definition to which the systems programmer has defined various disk volumes for general use.

Other UNIT= parameters could be UNIT=3390, UNIT=3380, UNIT=CART, UNIT=TAPE, or other generics and esoterics defined by the installation.

In the example above, the space parameter specifies 1 track primary and 1 track secondary. If a partitioned dataset were to be allocated then the number of directory blocks would also have to be defined, e.g. SPACE=(TRK,(1,1,6)). Space can be requested in various units, such as tracks, cylinders, bytes, logical records and so on.

A blocksize of 0 indicates that the system should determine the optimum block size for the particular dataset.

There is also a DSNTYPE= parameter which specifies the type of dataset required, e.g. DSNTYPE=LIBRARY requests a PDS/E. If this is not specified then defaults will apply.



There are two ways to specify a temporary data set name on the DD statement.

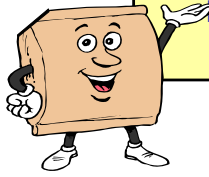
Code the DSN parameter using two ampersands (&&)

Omit the DSN parameter from the DD statement in which case the system will assign a name.

A temporary data set can be specified as a partitioned data set. The entire data set name can be 1-8 characters (excluding the member name)

## Summary

- Described the concepts of JCL and syntax
- Described the JOB, EXEC and DD statements
- Coded DD statements using specified information



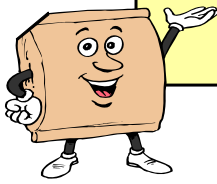
---

# Running JCL

## Objectives

---

- How to run JCL
- Examining the output of the JCL





## Submit a Job

### TSO Command Line

```
READY
submit 'aues100.tsoe.cntl(iefbr14) '
JOB AUES100A(JOB04970) SUBMITTED
READY
```

### ISPF/PDF Editor

```
File Edit Confirm Menu Utilities Compilers Test Help
Command ==> submit Scroll ==> PAGE
EDIT AUES100.TSOE.CNTL(AMS2) - 01.00
***** ***** Top of Data *****
000001 //AUES100A JOB (AUES100),
000002 // CLASS=L,
000003 // MSGLEVEL=(1,1),
000004 // NOTIFY=AUES100,
000005 // MSGCLASS=T
000006 //STEP1 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=*
000008 //SYSIN DD *
000009 DELETE AUES100.RRDS PURGE

.....
F1=HELP F2=SPLIT F3=END F4=RETURN F5=IFIND F6=BOOK
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE
```

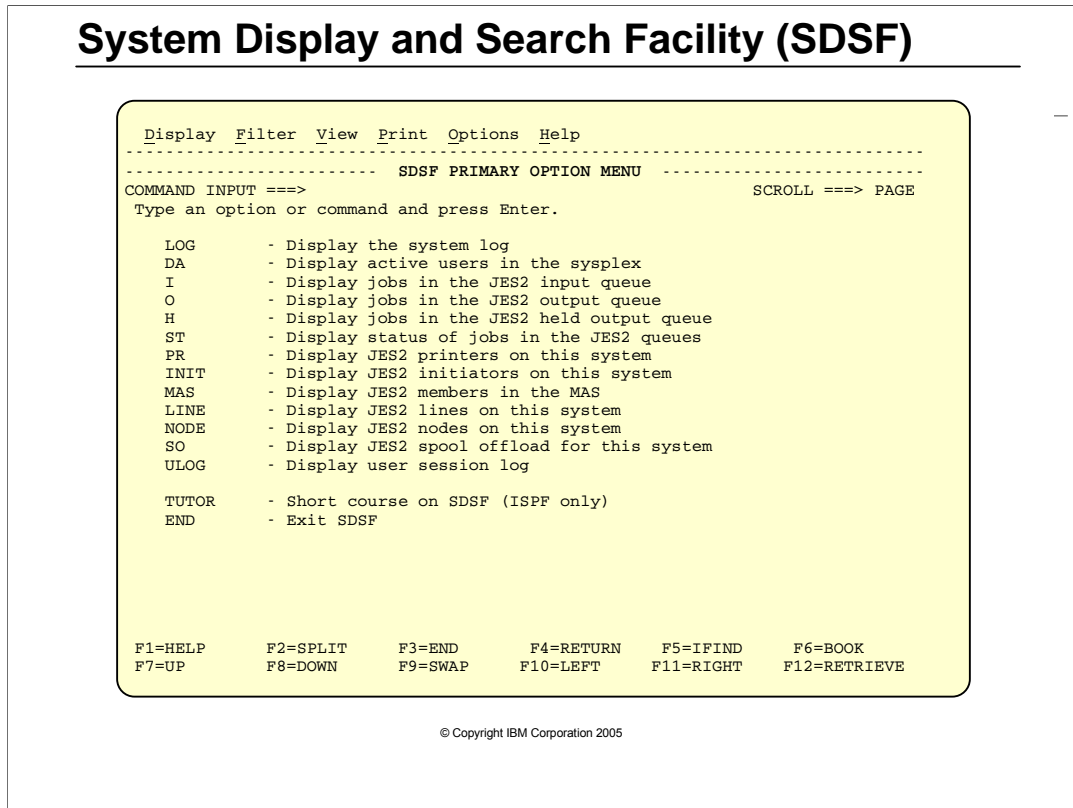
© Copyright IBM Corporation 2005

To submit a job, you can either use the TSO/E command SUBMIT, as introduced in the previous unit, or the ISPF/PDF primary command SUBMIT.

To submit a job using the TSO/E SUBMIT command, enter *submit* followed by the data set and member name on any command line or in basic TSO/E line mode. (Do not forget to prefix the command with *TSO*.)

To use the ISPF/PDF primary command SUBMIT just enter *submit* on the command line while editing the data set or member containing the JCL.

**Note:** SUBMIT may be abbreviated to SUB.



The **System Display and Search Facility (SDSF)** lets you control job processing (hold, release, cancel, purge), monitor job execution, display job output, control printers and initiators, check network lines, work with spools, issue JES2 and z/OS commands, and so forth.

To work with SDSF select option **S** from the *ISPF Primary Option Menu*. This results in the display of the *SDSF Primary Option Menu*, as shown above. To browse a job's output choose option **H**. This selection displays a list of all jobs in the JES2 **held** output queue.

The next chart shows an example of such a list.

## Viewing JCL output - held output

```

Display Filter View Print Options Help
-----
SDSF HELD OUTPUT DISPLAY ALL CLASSES LINES 78          LINE 1-1 (1)
COMMAND INPUT ==>                                     SCROLL ==> CSR
PREFIX=MV04* DEST=(ALL) OWNER=* SORT=JOBNAME/A
NP  JOBNAME  JOBID  OWNER  PRTY C ODISP DEST          TOT-REC TOT-
?   MV04MSTA JOB02857 MV04MST 144 X HOLD LOCAL          78

```

```

Display Filter View Print Options Help
-----
SDSF JOB DATA SET DISPLAY - JOB MV04MSTA (JOB02857)   LINE 1-3 (3)
COMMAND INPUT ==>                                     SCROLL ==> CSR
PREFIX=MV04* DEST=(ALL) OWNER=*
NP  DDNAME   STEPNAME PROCSTEP DSI D OWNER  C DEST          REC-CNT PAGE
    JESMSG LG JES2      2 MV04MST X LOCAL          40
    JESJCL   JES2      3 MV04MST X LOCAL           8
    JESYSMSG JES2      4 MV04MST X LOCAL          30

```

```

F1=HELP   F2=SPLIT  F3=END    F4=RETURN  F5=IFIND  F6=BOOK   F7=UP
F8=DOWN   F9=SWAP   F10=LEFT  F11=RIGHT  F12=RETP

```

The most useful command in SDSF is “SET DISPLAY ON” this will show you the PREFIX line of information as shown above.

To view the output from a batch job, select the Held output (H) option from SDSF. You can filter the job output displayed by use of the prefix command.

Notice that in the top display a prefix value of MV04\* has been used. This shows one job on the held output queue. To expand the listing enter a question mark (?) against the jobname.

The output of this is shown in the lower panel. The top entry is the JES2 Job Log, next is the job's JCL and the last entry shows JES system messages.

To select one of the above files, enter an 's' against the appropriate item.

Note that the number of individual items that appear in this display depends on the number of SYSOUT DD statements coded in the JCL.

## Viewing JCL output (JES2 Job Log)

```

1
0
      J E S 2  J O B  L O G  --  S Y S T E M  M V S L  --  N O D E
07.28.19 JOB02857 ---- MONDAY,    09 APR 2001 ----
07.28.19 JOB02857 IRR010I  USERID MVO4MST  IS ASSIGNED TO THIS JOB.
07.28.40 JOB02857 ICH70001I MVO4MST  LAST ACCESS AT 07:25:26 ON MONDAY, APRIL
07.28.40 JOB02857 EHASP373 MVO4MSTA STARTED - INIT 1A - CLASS A - SYS MVSL
07.28.40 JOB02857 IEF403I  MVO4MSTA - STARTED - TIME=07.28.40
07.28.40 JOB02857 -
07.28.40 JOB02857 --TIMINGS (MINS.)--
07.28.40 JOB02857 -JOBNAME STEPNAME PROCSTEP  RC  EXCP  CPU  SRB  CLOC
07.28.40 JOB02857 -MVO4MSTA          STEP1          00    8    .00   .00   .
07.28.40 JOB02857 IEC130I  SYSPRINT DD STATEMENT MISSING
07.28.40 JOB02857 -MVO4MSTA          STEP2          12    8    .00   .00   .
07.28.40 JOB02857 -MVO4MSTA          STEP3          00    9    .00   .00   .
07.28.40 JOB02857 CSV003I  REQUESTED MODULE IRT123  NOT FOUND
07.28.40 JOB02857 CSV028I  ABEND806-04  JOBNAME=MVO4MSTA  STEPNAME=STEP4
07.28.40 JOB02857 IEA995I  SYMPTOM DUMP OUTPUT
07.28.40 JOB02857 SYSTEM COMPLETION CODE=806  REASON CODE=00000004
07.28.40 JOB02857 TIME=07.28.40  SEQ=00210  CPU=0000  ASID=0027
07.28.40 JOB02857 PSW AT TIME OF ERROR 070C1000  810B48D2  ILC 2  INTC OD
07.28.40 JOB02857 NO ACTIVE MODULE FOUND
07.28.40 JOB02857 NAME=UNKNOWN
07.28.40 JOB02857 DATA AT PSW 010B48CC - 9FB0181C 0A0D18FB 180C181D
07.28.40 JOB02857 GPR 0-3 00001C00 84806000 00FCFA40 00000010
07.28.40 JOB02857 GPR 4-7 000000FF 007DBF30 00000004 0000000C

```

This shows an example section of the JES2 Job Log which gives details about how successful each job step was and provides other statistics.

## Viewing JCL output (JES2 Job Log) cont...

```

***** TOP OF DATA *****
G -- SYSTEM MVSL -- NODE 0S390LP1

PR 2001 ----
4MST IS ASSIGNED TO THIS JOB.
LAST ACCESS AT 07:25:26 ON MONDAY, APRIL 9, 2001
TARTED - INIT 1A - CLASS A - SYS MVSL
STARTED - TIME=07.28.40

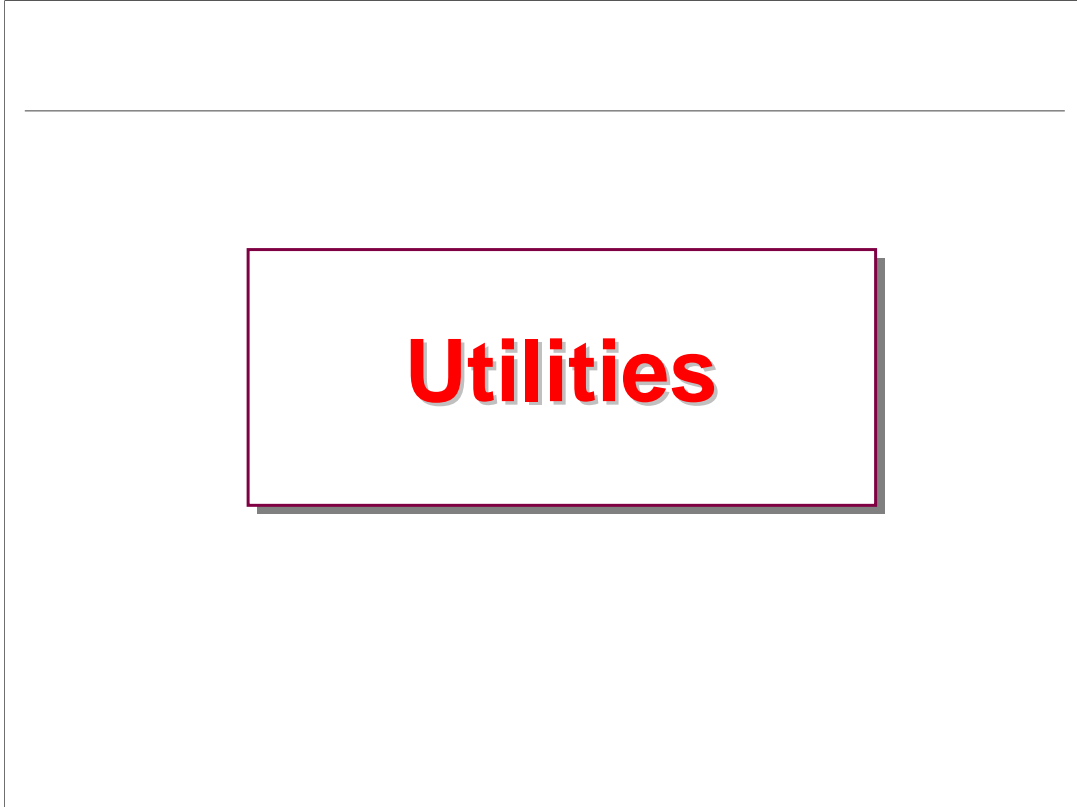
--TIMINGS (MINS.)--
PROCSTEP  RC  EXCP  CPU  SRB  CLOCK  SERV  PG  PAGE  SWAP  VIO  SWAPS
STEP1      00    8    .00  .00  .0    268  1    0    0    0    0
STATEMENT MISSING
STEP2      12    8    .00  .00  .0    427  1    0    0    0    0
STEP3      00    9    .00  .00  .0    241  1    0    0    0    0
MODULE IRT123 NOT FOUND
JOBNAME=MV04MSTA STEPNAME=STEP4
P OUTPUT
ODE=806 REASON CODE=00000004
=00210 CPU=0000 ASID=0027
OR 070C1000 810B48D2 ILC 2 INTC 0D
FOUND

B48CC - 9FB0181C 0A0D18FB 180C181D
00 84806000 00FCFA40 00000010
FF 007DBF30 00000004 0000000C

```

This shows the right hand section of the previous example display. Note the statistics CPU usage, paging etc; As this example job was trivial in nature most of the values are too small to register.

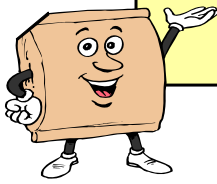




## Objectives

---

- Describe the function and use of the following utilities:
  - ▶ IEFBR14
  - ▶ IEBCOPY
  - ▶ IEBGENER
  - ▶ IDCAMS
- Describe the function and use of DFSORT





## Types of Utilities

<p><b>System Utility Programs</b></p> <ul style="list-style-type: none"> <li>• IEHINITT</li> <li>• IEHLIST</li> <li>• IEHMOVE</li> <li>• IEHPROGM</li> <li>• IFHSTATR</li> <li>• IDCAMS</li> </ul> <p><b>z/OS Feature - Licensed Program</b></p> <ul style="list-style-type: none"> <li>• DFSORT</li> </ul>	<p><b>Data Set Utility Programs</b></p> <ul style="list-style-type: none"> <li>• IEBCOMPR</li> <li>• IEBCOPY</li> <li>• IEBDG</li> <li>• IEBEDIT</li> <li>• IEBGENER</li> <li>• IEBIMAGE</li> <li>• IEBPTPCH</li> <li>• IEBUPDTE</li> </ul>
---	---

© Copyright IBM Corporation 2005

### Data Set Utility Programs

You can use data set utility programs to reorganize, change, or compare data at the data set or record level. These utilities allow you to manipulate partitioned, sequential or indexed sequential data sets, or partitioned data sets extended (PDSEs), which are provided as input to the programs. You can manipulate data ranging from fields within a logical record to entire data sets.

**IEBCOMPR** - Compare records in sequential or partitioned data sets, or PDSEs.

**IEBCOPY** - Copy, compress, or merge partitioned data sets or PDSEs; select or exclude specified members in a copy operation; rename or replace selected members of partitioned data sets or PDSEs.

**IEBDG** - Create a test data set consisting of patterned data.

**IEBEDIT** - Selectively copy job steps and their associated JOB statements.

**IEBGENER** - Copy records from a sequential data set or convert a data set from sequential to partitioned.

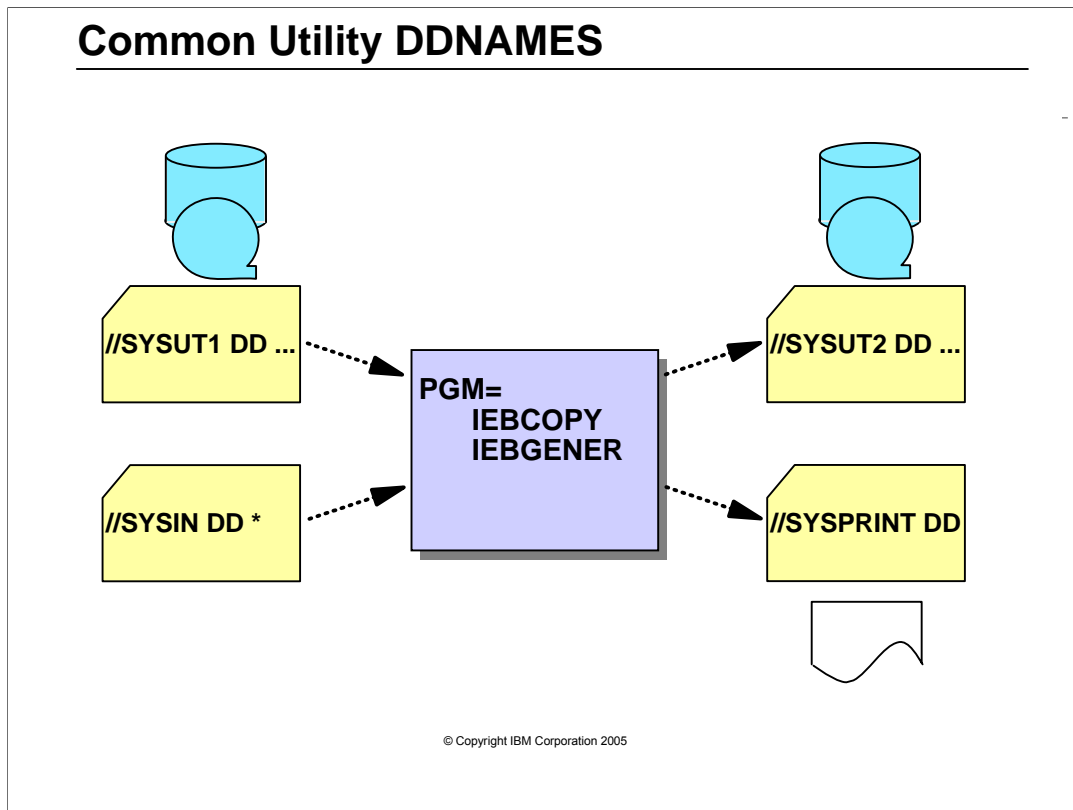
**IEBIMAGE** - Modify, print, or link modules for use with the IBM 3800 Printing Subsystem, the IBM 3262 Model 5, or the 4248 printer

**IEBISAM** - Unload, load, copy, or print an ISAM data set.

**IEBPTPCH** - Print or punch records in a sequential or partitioned data set.

**IEBUPDTE** - Incorporate changes to sequential or partitioned data sets, or PDSEs.

Data Facility Sort (DFSORT) is a member of the IBM Data Facility family of products. It is a high-performance data arranger.



Most utilities require the use of the ddnames **SYSUT1**, **SYSUT2**, **SYSIN**, and **SYSPRINT** to define their input and output and to pass control information to the executed program.

**SYSUT1** Specifies the input source for the program, which is usually a sequential data set, a PDS or PDSE, or a member of a PDS or PDSE.

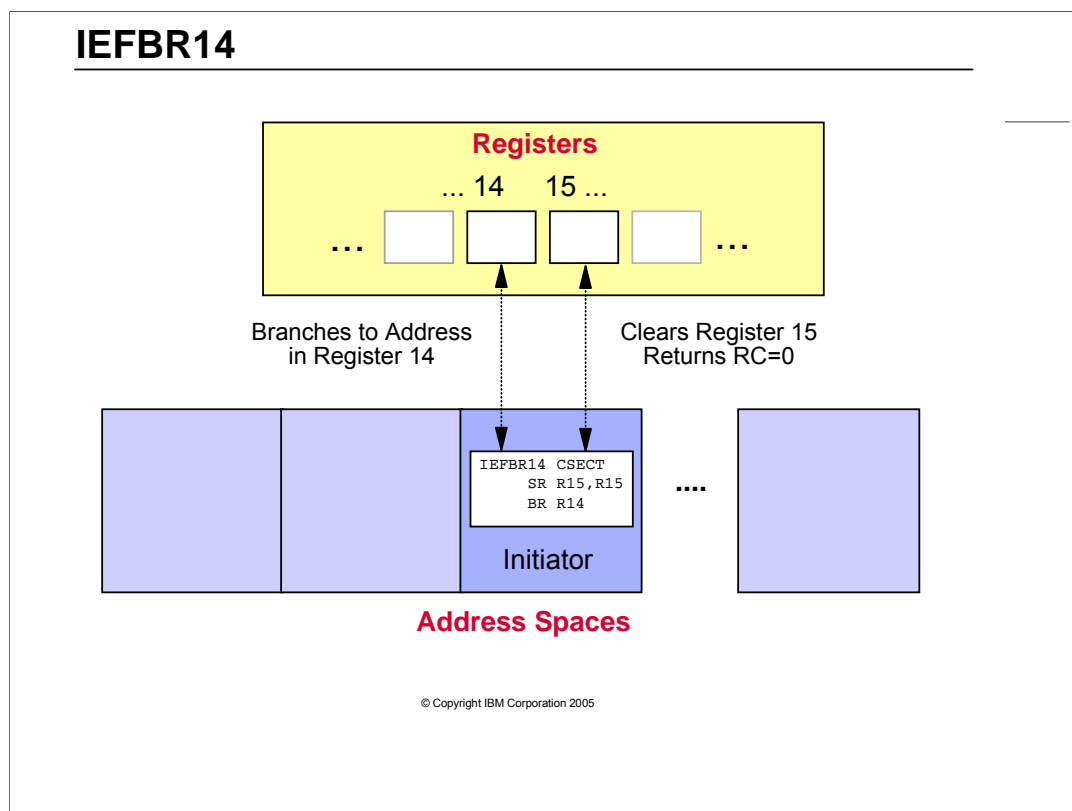
**SYSUT2** Defines where the output of the program is to be stored. **SYSUT2** usually specifies a sequential data set, a member of a PDS or PDSE, or a PDS or PDSE as the target for the performed function.

**SYSIN** Specifies the control data, which is usually supplied with the input stream or specifies DUMMY.

**SYSPRINT** Defines a sequential data set for messages. The data set can be written to a system output device, a tape volume, or a DASD volume.

Here is an example:

```
//PRINT JOB...
//STEP1 EXEC PGM=IEBGENER
//SYSUT1 DD DSNAME=AUES100.IN.DATA,DISP=SHR
//SYSUT2 DD DSNAME=AUES100.OUT.DATA,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
```



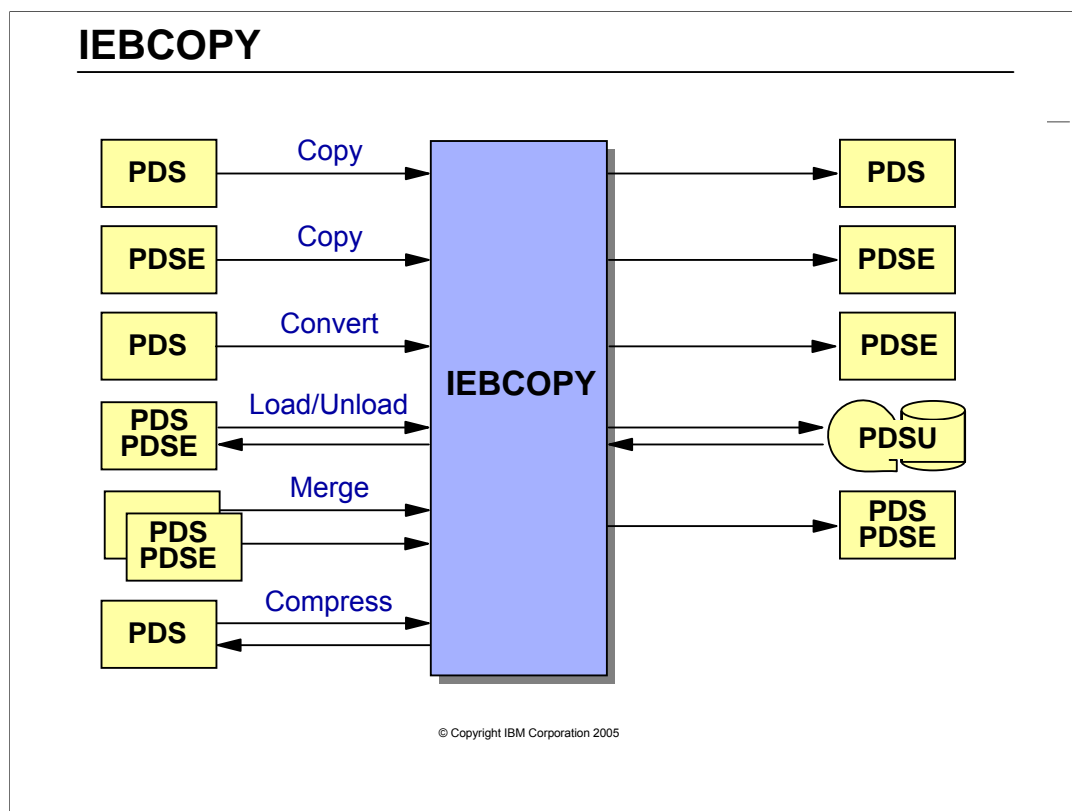
The only function of this program is to provide a zero (0) completion code. It is used as a safe vehicle to "execute JCL." The notion of executing JCL is considered incorrect terminology, but it conveys the idea very well. For example, consider the following job:

```
//OGDEN1 JOB 1,BILL,MSGCLASS=X
// EXEC PGM=IEFBR14
//A DD DSN=OGDEN.LIB.CNTL,DISP=(NEW,CATLG),VOL=SER=WORK02,
// UNIT=3390,SPACE=(CYL,(3,1,25))
//B DD DSN=OGDEN.OLD.DATA,DISP=(OLD,DELETE)
```

This is a useful job although the program that is executed (IEFBR14) does nothing. While preparing to run the job, the initiator allocates OGDEN.LIB.CNTL and keeps the data set when the job ends. It also deletes OGDEN.OLD.DATA at the end of the job. The DD names A and B have no meaning and are used because the syntax of a DD statement requires a DD name.

The same functions to create one data set and delete another could be done through ISPF, for example, but these actions might be needed as part of a larger sequence of batch jobs.

**Note:** The name IEFBR14 is interesting. One IBM group writing early OS/360 code used the prefix IEF for all their modules. In assembly language BR means Branch to the address in a Register. Branching to the address in general register 14 is the standard way to end a program. While not an especially clever name, practically all dedicated z/OS users remember IEFBR14 easily. IEFBR14 is not a utility, in the sense that it is not included in the Utilities manual. However, there is no other practical category for this useful program, so we have arbitrarily placed it in the utility category.



This utility is commonly used for several purposes:

To copy selected (or all) members from one partitioned data set to another.

To copy a partitioned data set into a unique sequential format known as an unloaded partitioned data set. As a sequential data set it can be written on tape, sent by FTP,<sup>2</sup> or manipulated as a simple sequential data set.

To read an unloaded partitioned data set (which is a sequential file) and recreate the original partitioned data set. Optionally, only selected members might be used.

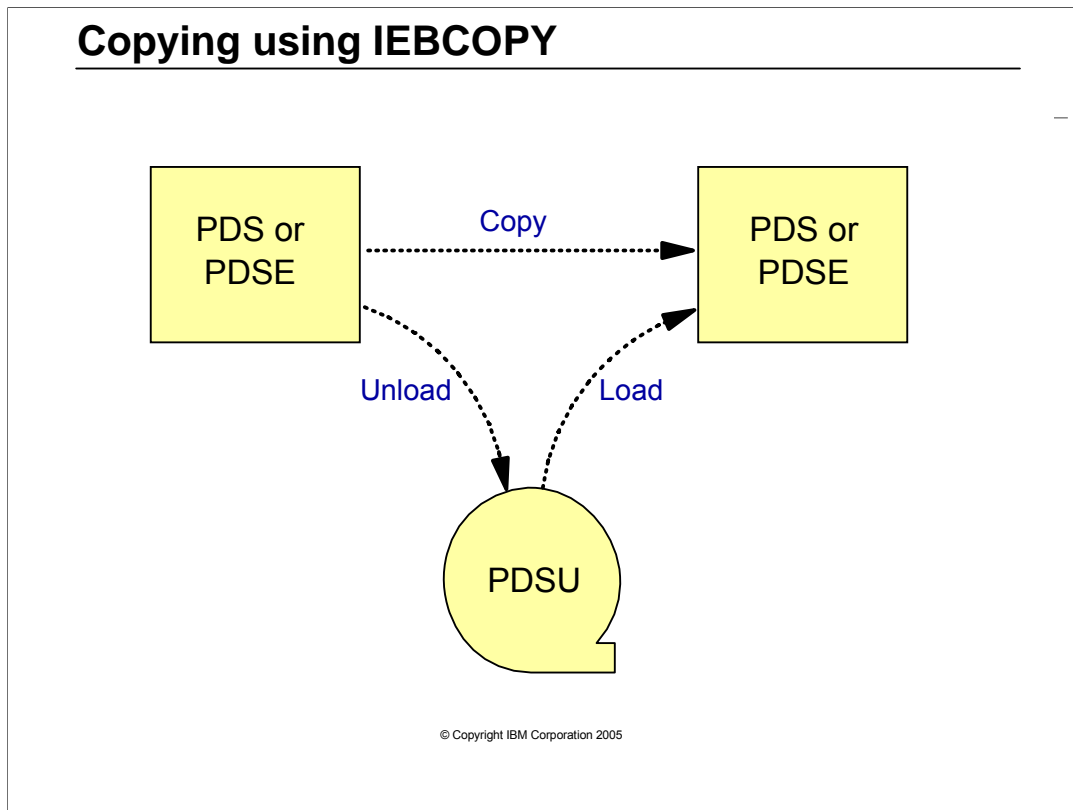
To compress partitioned data sets (in place) to recover lost space.

Most z/OS software products are distributed as unloaded partitioned data sets. The ISPF copy options (option 3.3, among others) uses IEBCOPY "under the covers." Moving a PDS or PDSE from one volume to another is easily done with IEBCOPY. If there is a need to manipulate partitioned data sets in batch jobs, IEBCOPY is probably used. Equivalent manipulation under TSO (using ISPF) uses IEBCOPY indirectly.

A simple IEBCOPY job might be the following:

```
//OGDEN5 JOB 1,BILL,MSGCLASS=X
// EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=OGDEN.LIB.SOURCE
//SYSUT2 DD DISP=(NEW,KEEP),UNIT=TAPE,DSN=OGDENS.SOURCE,
// VOL=SER=123456
```

This job will unload OGDEN.LIB.SOURCE (which we assume is a partitioned data set) and write it on tape. (The name TAPE is assumed to be an esoteric name that the local installation associates with tape drives.) By default IEBCOPY copies from SYSUT1 to SYSUT2. Notice that the data set name on tape is not the same as the data set name used as input (the same name could be used, but there is no requirement to do so).



The following job could be used to restore the PDS on another volume:

```
//JOE6 JOB 1,JOE,MSGCLASS=X
// EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=OLD,UNIT=TAPE,DSN=OGDENS.SOURCE,VOL=SER=123456
//SYSUT2 DD DISP=(NEW,CATLG),DSN=P390Z.LIB.PGMS,UNIT=3390,
// SPACE=(TRK,(10,10,20)),VOL=SER=333333
```

In this example IEBCOPY will detect that the input data set is an unloaded partitioned data set. We required external knowledge to determine that the data set would fit in about 10 tracks and should have 20 directory blocks.

Instead of using DD DUMMY for SYSIN we could this:

```
//SYSIN DD *
COPY OUTDD=SYSUT2,INDD=SYSUT1
SELECT MEMBER=(PGM1,PGM2)
/*
```

The OUTDD and INDD parameters specify the DD names to be used. In this case we simply used the default names, but this is not required. The SELECT statement specifies the member names to be processed.

Restoring a partitioned data set from an unloaded copy automatically compresses (recovers lost space) the data set.

## JCL Example – IDCAMS to setup VSAM LDS

---

```
//TSA0001C JOB CLASS=A,MSGCLASS=Q,  
//          NOTIFY=&SYSUID,REGION=0M  
//DEFINE   EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//AMSDUMP  DD SYSOUT=*  
//DASD0    DD DISP=OLD,UNIT=3390,VOL=SER=SMS005  
//SYSIN     DD *  
          DEFINE CLUSTER (NAME(OMVS.COMPAT.AGGR002) -  
          STORAGECLASS(BASE) -  
          LINEAR CYL(25 0) SHAREOPTIONS(3))
```

The above example illustrates the use of IDCAMS to create a VSAM Linear Dataset of 25 cylinders in size, with no secondary allocation.

## JCL Example – RACF commands via TSO batch

```
//TSA0001A JOB NOTIFY=&SYSUID
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDGROUP DFSGRP SUPGROUP(SYS1) OMVS(GID(2))
ADDUSER DFS OMVS(HOME(/opt/dfslocal/home/dfsctl) UID(0)) +
  DFLTGRP(DFSGRP) +
  AUTHORITY(CREATE) UACC(NONE)
RDEFINE STARTED DFS.** STDATA(USER(DFS))
RDEFINE STARTED ZFS.** STDATA(USER(DFS))
CO DFS GRO(STUDENT) AUTH(CREATE)
SETROPTS RACLIST(STARTED)
RALTER STARTED (ZFS.***) STDATA(TRUSTED(YES))
SETROPTS RACLIST(STARTED) REFRESH
```

The above example shows the typical way a RACF person might use TSO in batch to do the following:

Add a new group called DFSGRP

Add a new userid called DFS, whose default group is DFSGRP, and who has an OMVS segment

Define two started task (STCs) profiles that are associated with userid DFS

Connect the userid DFS to another group called STUDENT

Update the STC profile for ZFS.\*\* to TRUSTED

Refresh the in-storage 'started task' profiles.

When running such jobs in TSO batch, a return code of zero (0) means that the command was executed, not necessarily that it achieved what was wanted. You should always check the SYSPRINT output to check on the RACF messages and ensure there were no unexpected errors.

## JCL Example – IEBCOPY copy selected members

```
//COPYLIBS EXEC PGM=IEBCOPY
//*****
//SYSPRINT DD SYSOUT=*
//IN1X DD DSN=D80WW.CB47V1.BASE.VTAMLIB,DISP=SHR
//OU11 DD DSN=SYS2.H1.VTAMLIB,DISP=(NEW,CATLG),
// VOL=SER=MVW210,STORCLAS=NONSMS,UNIT=3390,
// LIKE=D80WW.CB47V1.BASE.VTAMLIB
//OU12 DD DSN=SYS2.H2.VTAMLIB,DISP=(NEW,CATLG),
// VOL=SER=MVW210,STORCLAS=NONSMS,UNIT=3390,
// LIKE=D80WW.CB47V1.BASE.VTAMLIB
//SYSUT3 DD UNIT=VIO,SPACE=(CYL,(5,5))
//SYSUT4 DD UNIT=VIO,SPACE=(CYL,(5,5))
//SYSIN DD *
COPY OUTDD=OU11,INDD=((IN1X,R))
S M=AMODSNA,ISTINCLM,ITPECHO,PP3MODE,SNAUSS,WP1MODE,
COPY OUTDD=OU12,INDD=((IN1X,R))
S M=AMODSNA,ISTINCLM,ITPECHO,PP3MODE,SNAUSS,WP1MODE,
```

The above example uses IEBCOPY to do the following:-

Copies the members AMODSNA, ISTINCLM, etc from the data set with a DD name of IN1X to

- A new data set with a DD name of OUT11
- A new data set with a DD name of OUT12

The SYSUT3 & 4 DDs provide additional work space for IEBCOPY, should this be required.



## Time for Lab Exercise

---

### **Exercise 8** JCL Exercises



© Copyright IBM Corporation 2005

### **Summary of Labs**

JCL basic exercises

## Time for Lab Exercise

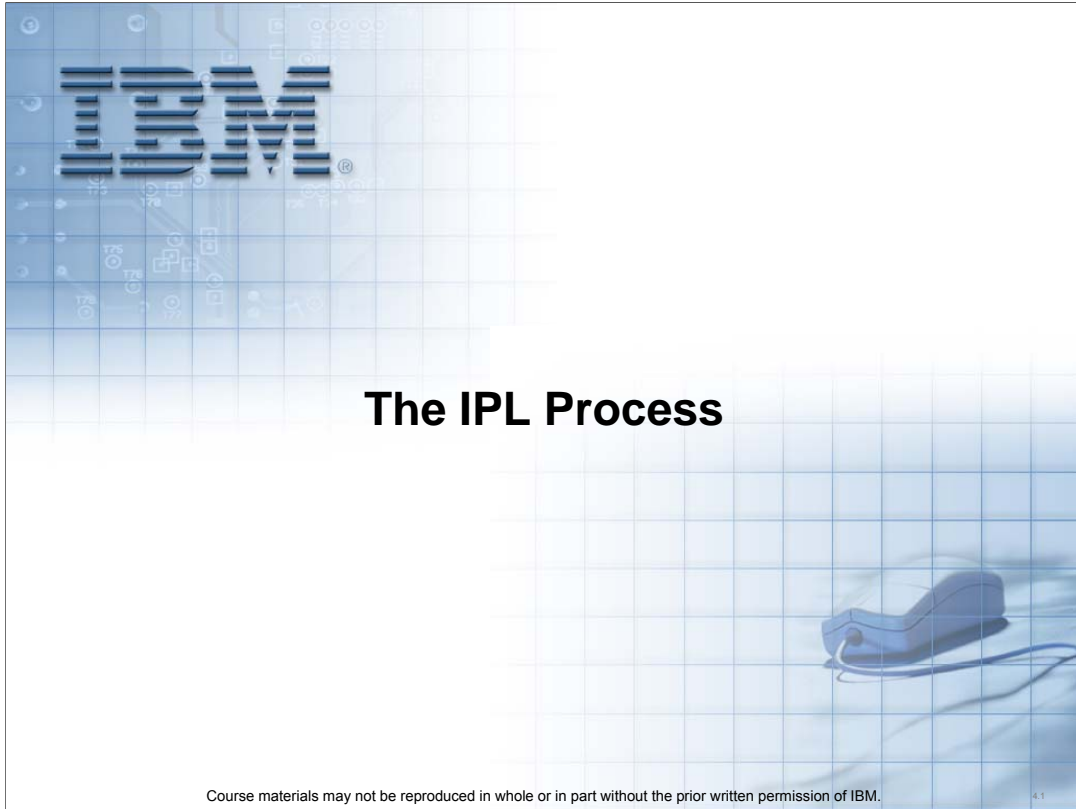
---

### **Exercise 9** JCL Procedures



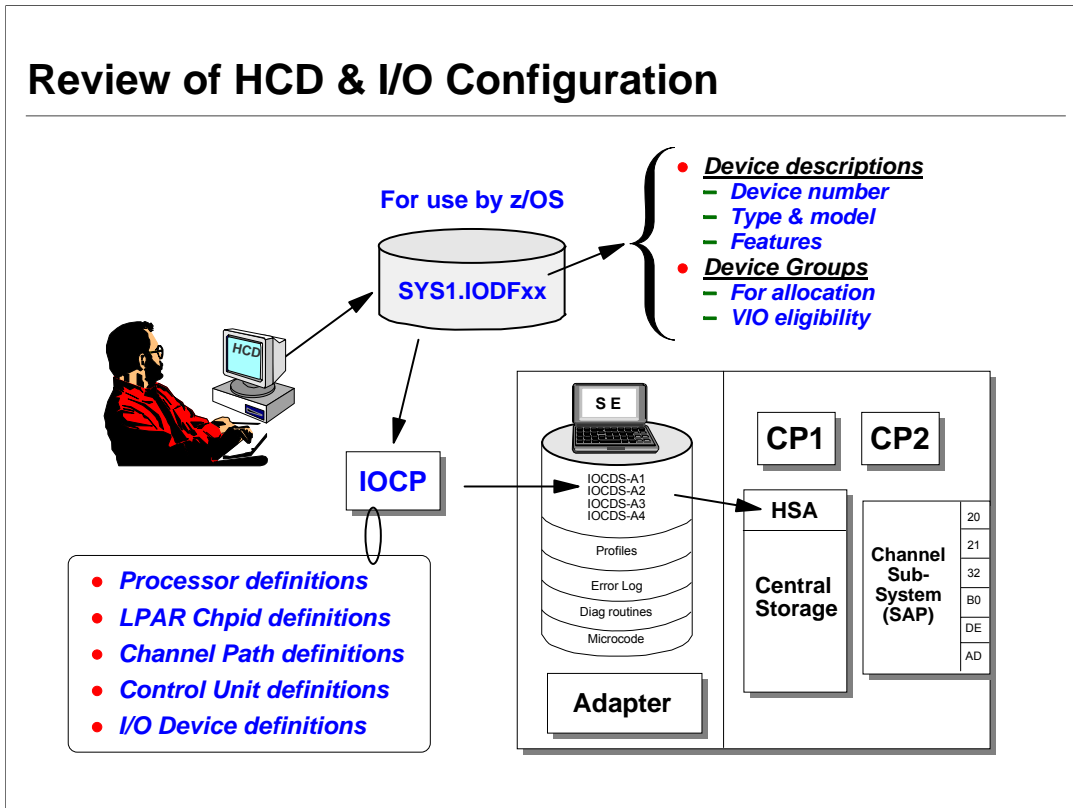
© Copyright IBM Corporation 2005

JCL Procedures



## Objectives

- **Describe z/OS initialization process**
  - LOAD
  - IPL (+IRIMs)
  - NIP (+RIMs)
- **Describe Load Parameter specified for IPL**
- **Discuss LOADxx member of PARMLIB**
  - IODF statement
  - SYSCAT statement
  - SYSPARM statement
- **Describe the startup of system and other address spaces**
- **Describe the shutdown process**

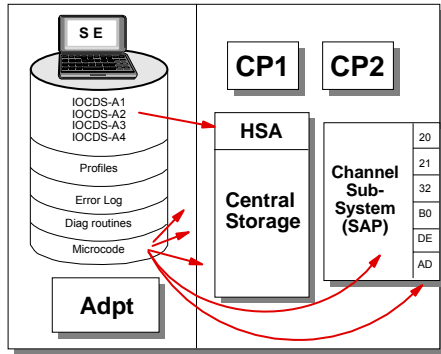


This foils reviews the way in which the hardware configuration has been specified, both to the z/OS operating system and the Channel Subsystem.

HCD generates an Input/Output Definition File (IODF) from details configured by the HCD user, typically a systems programmer. z/OS accesses the IODF during the IPL process in order to build its own I/O configuration (control blocks). An additional part of the HCD process uses data from the IODF to create the I/O Configuration Data Set (IOCDS) for use by the Channel Subsystem. (HCD invokes the I/O Configuration Program - IOCP, to perform this function).

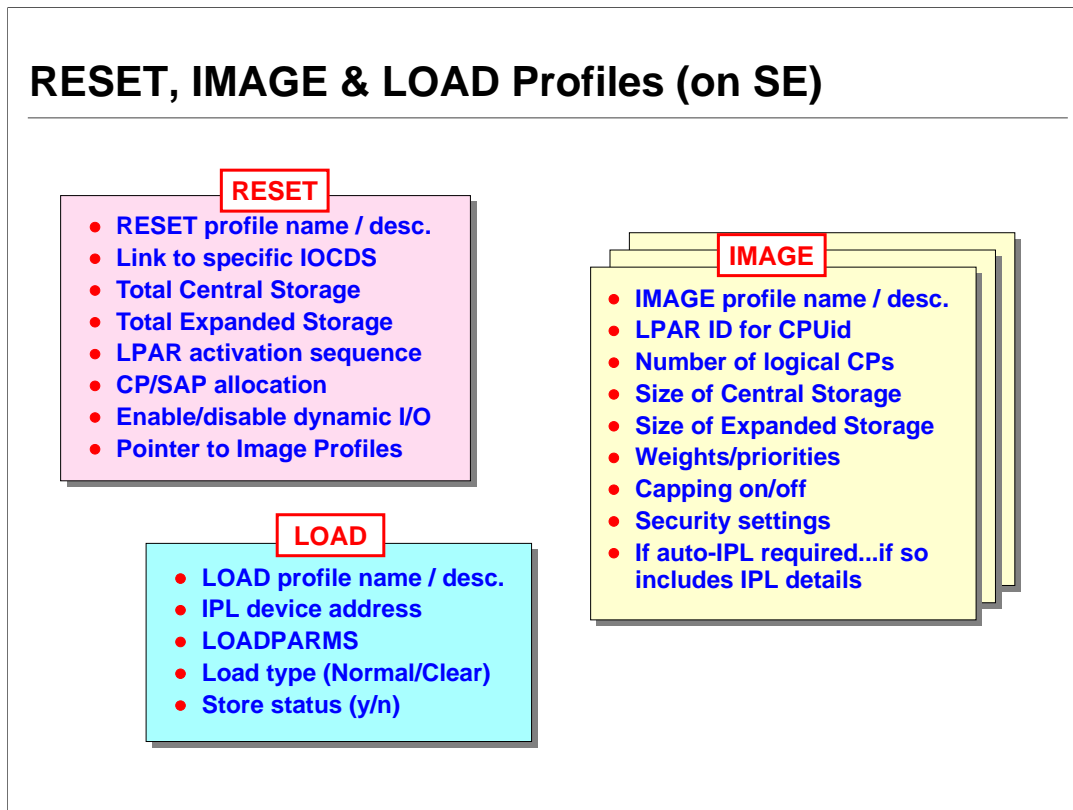
At Power-on-reset (POR) the appropriate IOCDS is loaded into the Hardware Systems Area (HSA) where it is used by the channel subsystem. The HSA is a portion of central storage and varies in size depending on the number of configured devices, chpids, and so on.

## Power on Reset (POR) complete state



- RESET and IMAGE profiles processed
- Microcode loaded into all components
- IOCDS processed & loaded into HSA
- LPARs in an activated state, ready for IPL



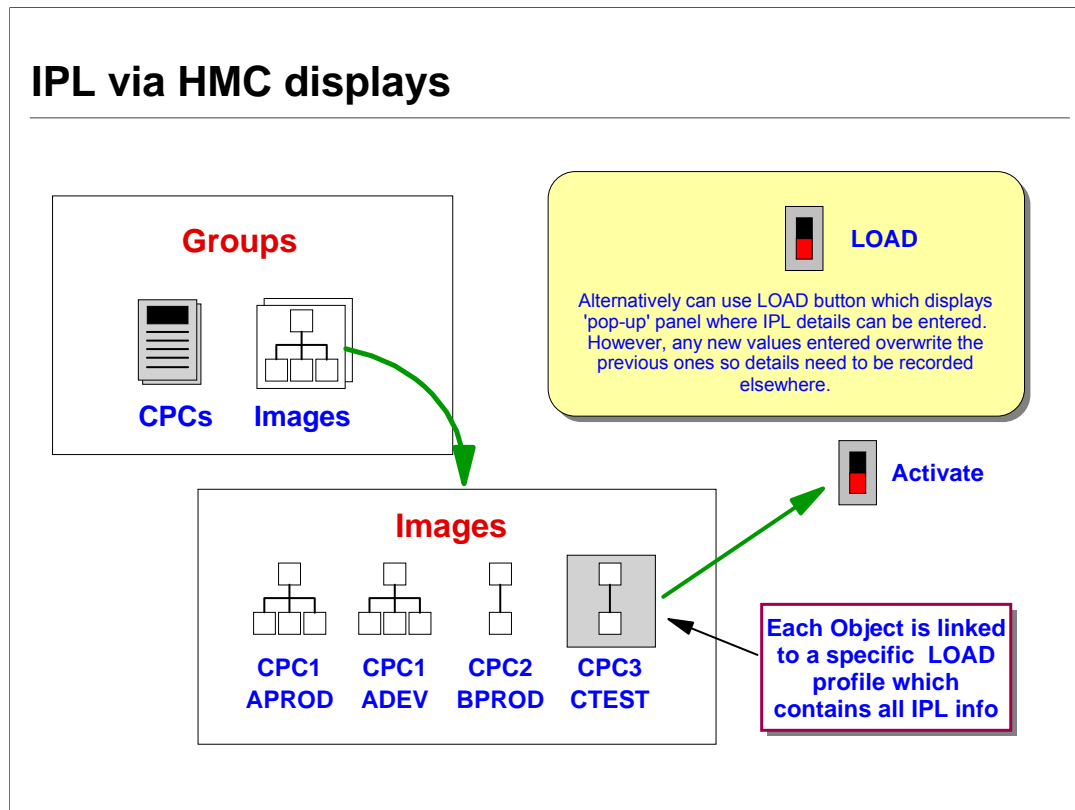


As shown in the foil above a RESET profile contains configuration and setup information for the CPC as a whole. An IMAGE profile, on the other hand, applies to individual LPARs. The RESET profile contains links to specified IMAGE profiles. Normally the IMAGE profiles would be processed directly after the RESET profile. However when an LPAR is manually deactivated and then reactivated, the IMAGE profile will be processed as part of the activation.

IMAGE profiles can be configured so that a z/OS system is automatically IPL'd once the LPAR is active. In this case, IPL details are specified within the IMAGE profile.

LOAD profiles contain IPL information and, when linked to an IMAGE object, i.e. a specific LPAR, they can be used to IPL that particular LPAR (or basic image).

All the above profiles are defined from the HMC application, and then stored on the Service Element (SE) hard drive.



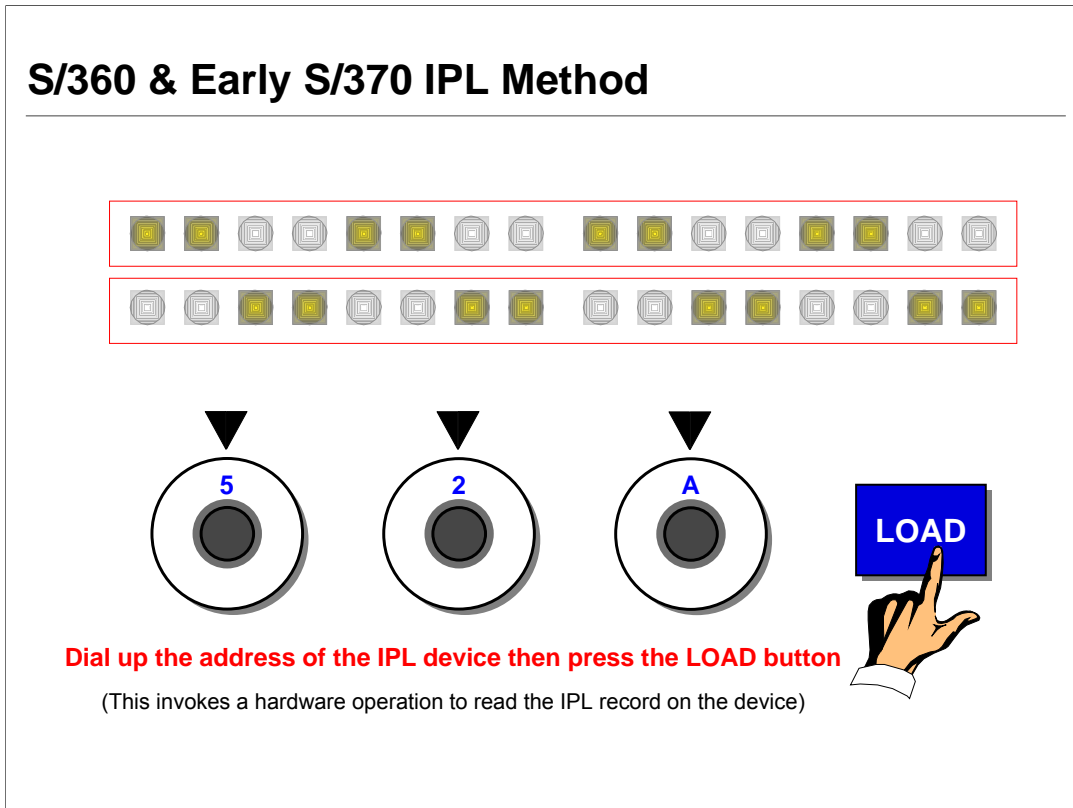
IPL on today's CMOS boxes is done via the HMC graphical user interface. The first display shown is the GROUPS display. This shows CPCs and Images. When the Images icon is selected, the screen section is replaced by icons of all the images, which includes standard LPARs, base mode images and coupling facility images.

In the above example, image CPC3 is to be IPLed. The process to do this is as follows:

- Select the CPC3 icon (by single clicking on the icon). When selected, the icon background turns grey.
- Double click the ACTIVATE icon. (There are variations on this approach such as drag-and-drop).
- A confirmation pop-up panel appears. Select Yes or No depending on details presented.
- Another pop-up panel appears which gives the status of the IPL process (i.e. In progress.. then ... Completed).

Another approach is not to use the ACTIVATE button but to use the LOAD button. If this is used a pop-up panel appears in which all the IPL information can be entered. This information is saved across IPLs and PORs but any information entered overwrites any previously entered. Consequently if an IPL test system, or standalone dump system is IPLed operations must have a record of the standard IPL information and re-enter this.

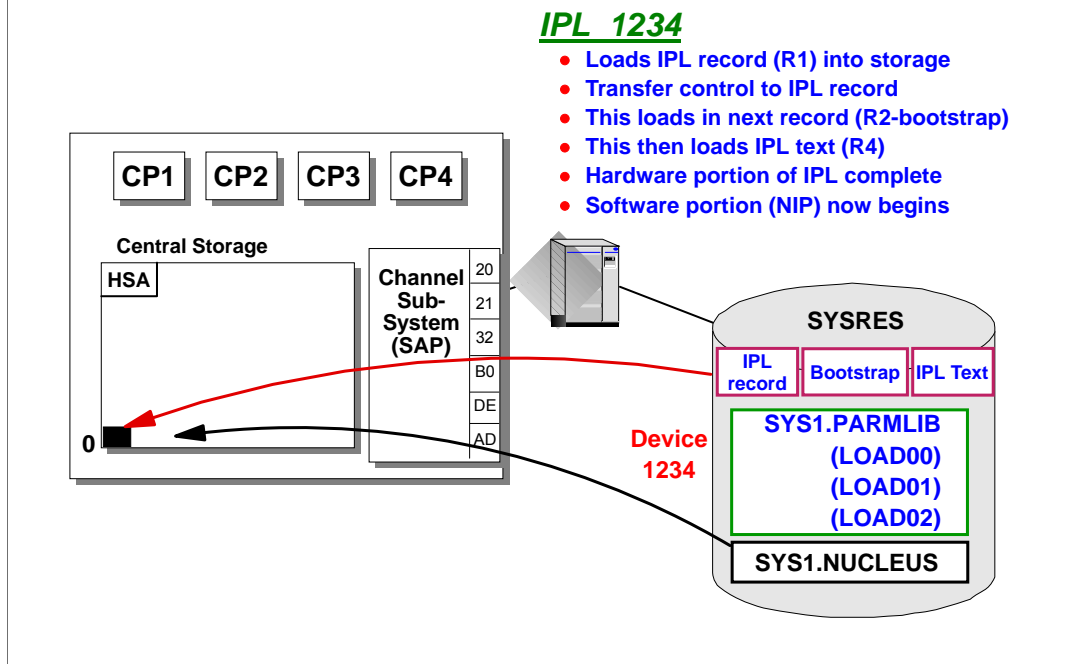




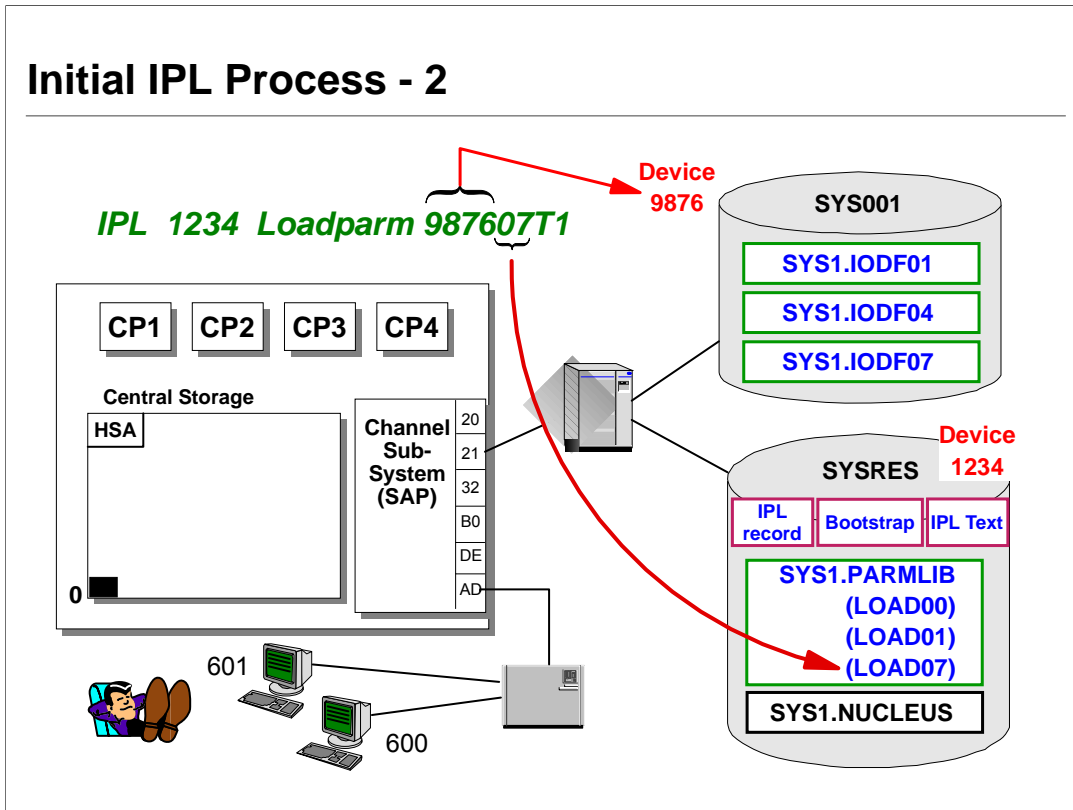
On the early range of IBM processors (S/360 and S/370), to IPL from a particular I/O device the operator dialed up the three digit device address and then pressed the LOAD button. These dials and buttons were on the front of the processor. The pressing of the LOAD button invoked a hardware process which went out to the device and read the IPL record. This hardware initiated process still takes place today but the interface is somewhat different, i.e. via the GUI interface just discussed.

Today we have 4-digit device numbers and need to pass more information to the system than just the IPL device address.

## Initial IPL Process - 1



When the Load or Activate function is invoked from the HMC for a particular Image the hardware converts the device number entered in the Load address to the corresponding subchannel number and enables that subchannel. The Channel Subsystem (CSS) sends an IPL Channel Command Word (CCW) to the IPL device. This will cause it to seek to cylinder 0, head 0 and read record 01. Record 01 will provide CCWs to read Record 02. Record 02 will provide CCWs to read Record 04. After record 04 has been read the Hardware portion of the IPL is complete. The software portion of the IPL now begins, that is the Nucleus Initialisation Program (NIP).



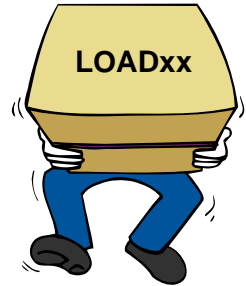
The LOADPARM contents tell the system which device address holds the IODF data sets (first four digits, e.g. 9876). It also indicates which LOADxx member is to be used. In our example this points to the LOAD07 member of SYS1.PARMLIB. Note that the LOADxx member can be located in a SYSn.IPLPARM dataset, instead of in SYS1.PARMLIB.

The remaining digits of the LOADPARM refer to the level of message suppression and the Nucleus ID to be used.

## LOADxx Usage

---

- **LOADxx specifies:**
  - IODF dataset information
  - Optional information
    - Nucleus module identifier
    - NUCLST member suffix
    - Master Catalog information
    - System Parameters information
    - Name of the Sysplex
    - System Symbolics information
- **Required for IPL**
- **Referenced in Load Parameter**



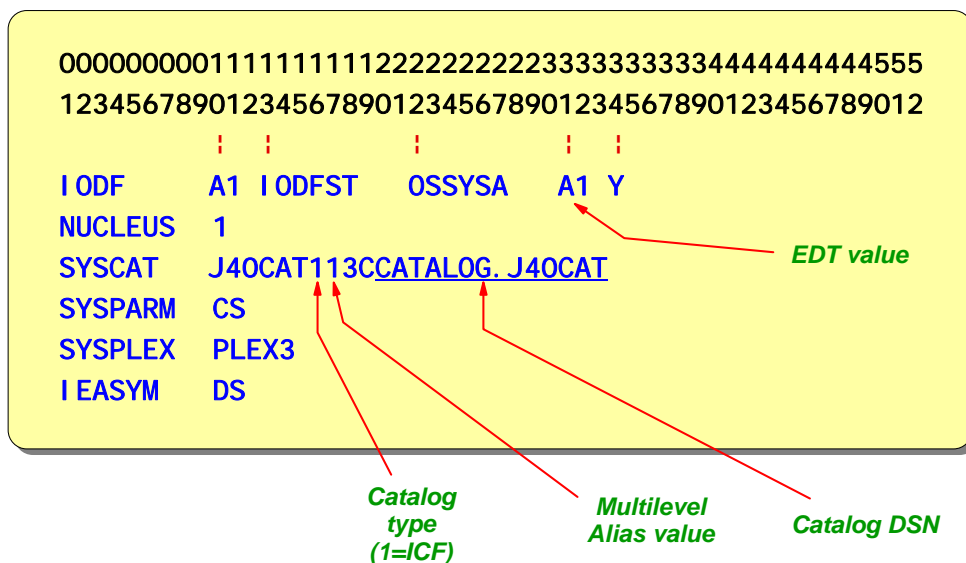
LOADxx is a PARMLIB member

It can reside in either SYS1.PARMLIB or SYSn.IPLPARM

LOADxx can specify sysplex name and point to the system symbolics member.

z/OS systems must be IPLed using an IODF

## LOADxx



The resulting IODF name in the above example will be IODFST.IODFA1.

SYSPARM value is appended to IEASYS to select the appropriate IEASYSxx member of SYS1.PARMLIB (i.e. IEASYSCS).

IEASYM identifies one or more IEASYMxx members of SYS1.PARMLIB that the system is to use.

SYSPLEX specifies the name of the sysplex in which the system participates. Because the system processes LOADxx early during initialisation, you can use the defined substitution text for the &SYSPLEX system symbol in other parmlib members.

Column 22 gives the Operating System Record Name as OS390A

Column 31 gives the Eligible Device Table to be used, i.e.. EDT 02

Column 34 (Y) tells the system to preload all device support code so that the correct drivers are there for use by dynamic reconfiguration.

## Load Parameters

<b>IODF</b>	<b>LOADxx</b>	<b>IMSI</b>	<b>NUCx</b>
<b>DDDD</b>	<b>XX</b>	<b>S</b>	<b>N</b>

*Dots are used  
as placeholders*

<b>DDDDXXSN</b>	<b>Load parameter values</b>
<b>DDDD . . . .</b>	<b>Device number of the volume containing the IODF dataset (the default is SYSRES)</b>
<b>. . . . XX . .</b>	<b>Suffix of the LOADxx member to be used (the default is 00)</b>
<b>. . . . . S .</b>	<b>Initial Message Suppression Indicator (IMSI) The default suppresses most informational messages and does not prompt for system parameters, but uses the LOADxx values, if specified.</b>
<b>. . . . . N</b>	<b>Nucleus ID to be used (the default is 1 IEANUC01)</b>

IODF Device Number specifies the device number of the disk that contains the IODF data sets. Use leading zeros to create 4-digit number if needed. Default, indicated by dots (periods) is device number of IPL Volume specified in the L1 field.

LOADxx suffix specifies the 2 hexadecimal digits that point to a particular LOADxx member in either SYSn.IPLPARM (n=0-9) or SYS1.PARMLIB. The default, indicated by dots, is 00.

Prompt feature specifies a prompt option:

. = do not prompt operator and do not display informational messages

A = prompt operator and display info messages

C = prompt operator for master catalog but don't display info messages.

D = prompt operator for System Parameters but don't display info messages

M = display info messages but don't prompt operator (use LOADxx details as input)

P = prompt operator but don't display info messages

S = same as C

T = same as D

If P for prompt is selected the SYSCATLG pointer must be available. The pointer is not created by HCD.

Alternate Nucleus points to an alternate IEANUC0x member of SYS1.NUCLEUS. A dot defaults to the IEANUC0x member specified in the LOADxx member. For more information see the z/OS System Commands manual.

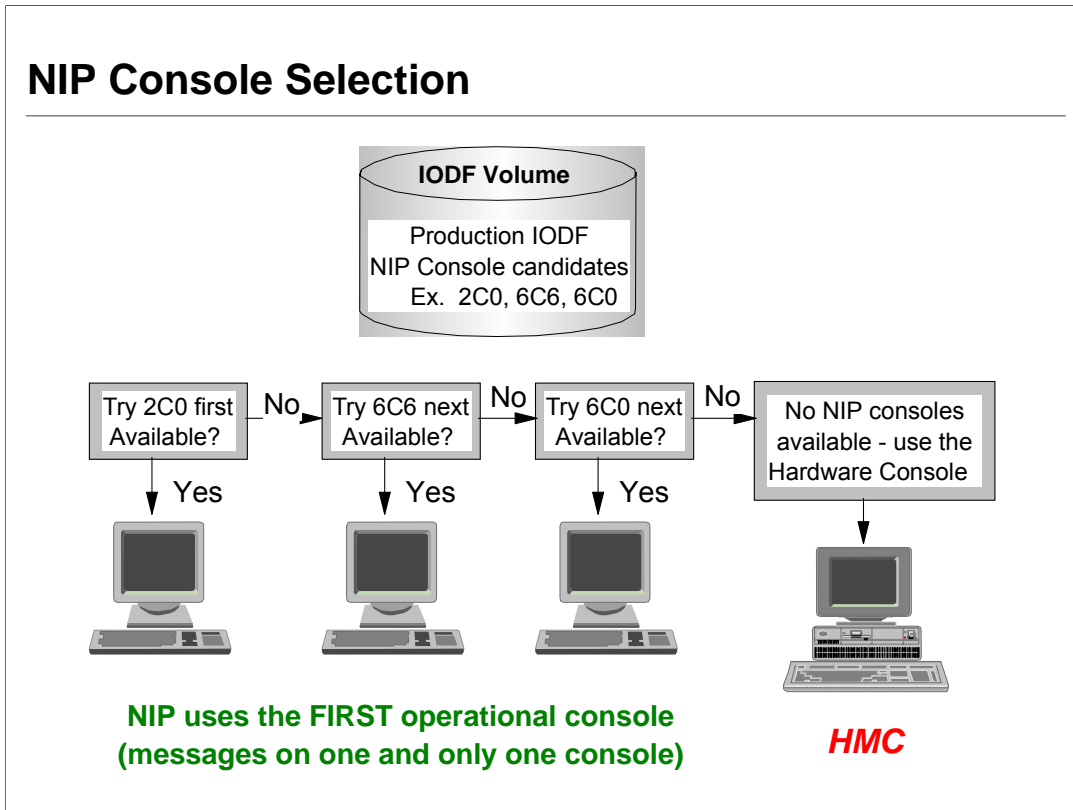
## IMSI Character

<i>IMSI Character</i>	<i>Display Informational messages</i>	<i>Prompt for Master Catalog Response</i>	<i>Prompt for System Parameter Response</i>
Period (.) or blank	NO	NO	NO
A	YES	YES	YES
C	NO	YES	NO
D	YES	YES	NO
M	YES	NO	NO
P	NO	YES	YES
S	NO	NO	YES
T	YES	NO	YES

The IMSI (Initial Message Suppression Indicator) field is the seventh character in the load parameter field.

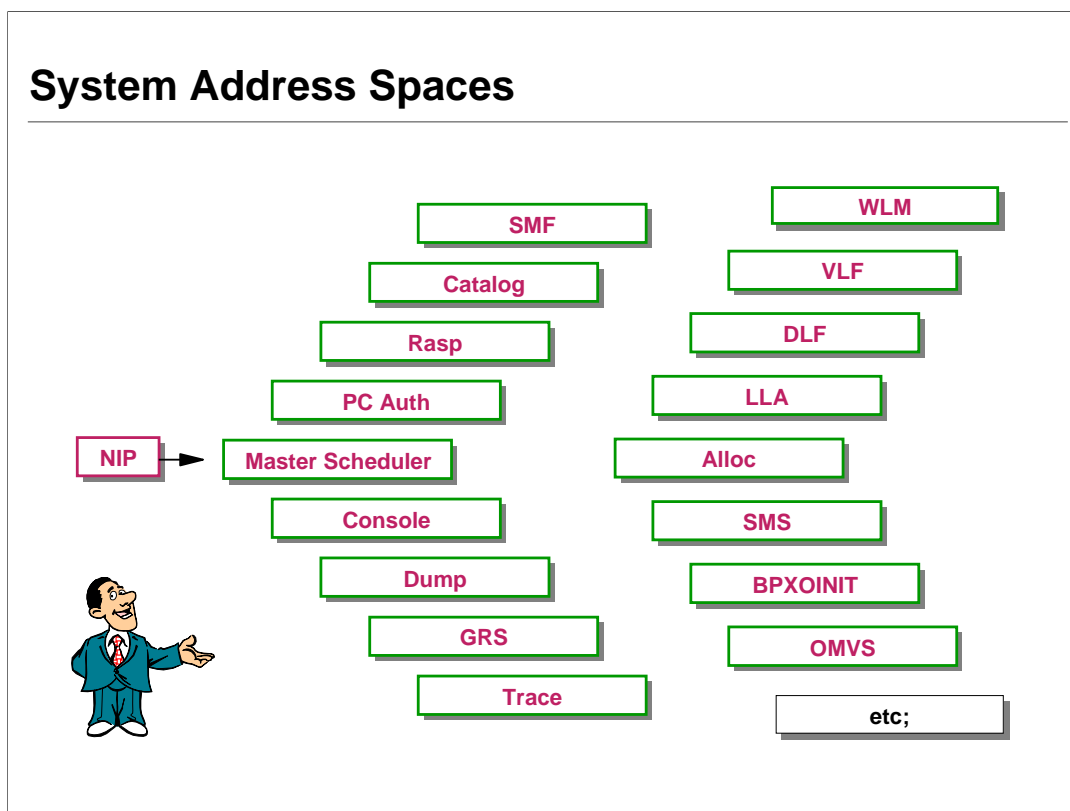
An IMSI value of 'M' is advised in a sysplex environment.

Note also that, in the event of a console failure, the console display will only switch to the HMC if messages are allowed.



NIP only uses the first operational NIP console that has been defined as a NIP. Potential NIP consoles are defined in HCD against a specific operating system ID. NIP will attempt to use the first defined NIP console, then the second and so on. If all the NIP channel-attached consoles are not operational then the 'Operating System Messages' task on the HMC can be used.





As part of the initialisation process, z/OS address spaces have to be created. First the Master Scheduler address space (`*MASTER*`) is created and initialised. `*MASTER*` supports the creation of other system address spaces. Some of the system address space functions are as follows:

`*MASTER*` is the MVS manager

PCAUTH (Program Call Authorisation) regulates authorities for 'inter address communication' (cross memory)

TRACE provides tracing functions for each processor

GRS regulates the access to common resources

DUMP allows recording of debugging information

CONSOL provides support for console communications

ALLOCAS records and regulates the allocation of I/O devices to jobs

LLA (Library Lookaside) improves performance of programs search for PDS libraries.

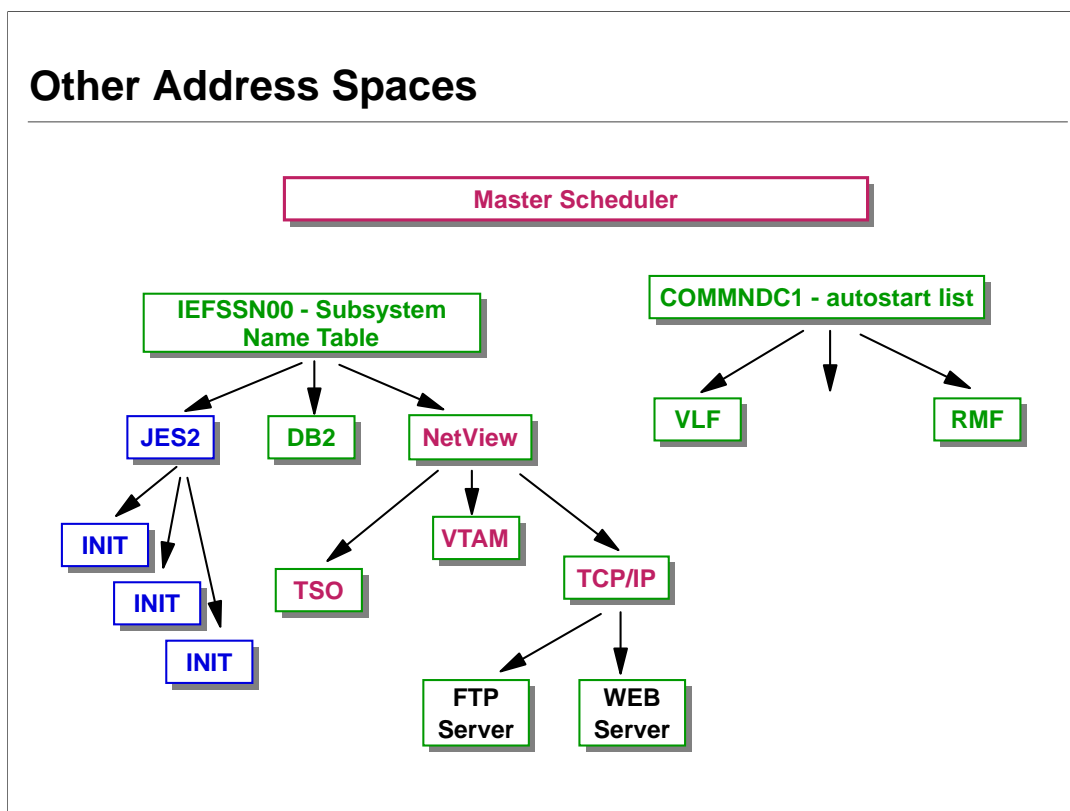
DLF provides support for Hiperspaces

VLF manages dataspace on behalf of other address spaces, e.g. LLA, CATALOG, etc;

CATALOG executes all requests to access catalogs

RASP is used by the Real Storage Manager. It also contains the tables describing data spaces

SMF records statistical data



z/OS has different mechanisms to start subsystems or procedures. As mentioned earlier, the Subsystem Names Table (IEFSSNxx member of SYS1.PARMLIB) can be used to start subsystems. This would typically be used for JES2. When JES2 initialises it, in turn, starts up the Initiator address spaces which it needs to run batch jobs. The example also shows NetView being started in this way (i.e. via the Subsystems Interface - SSI). It, in turn starts VTAM, then when VTAM is active NetView starts TCP/IP and TSO.

VTAM in turn activates its resources (major nodes) according to definitions in VTAMLST.

TCP/IP can be setup to auto-start other address spaces when it is ready. This example shows the FTP server and Web server address spaces being started in such a way.

The COMMNDxx member of SYS1.PARMLIB is another facility to automatically start procedures or subsystems.

## z/OS System Initialisation Summary

---

- **LOAD**
  - is a hardware function
- **IPL program or IPL text**
  - contains some IRIMs  
*(IPL Resource Initialization Modules)*
  - loads **NUCLEUS** and **NIP**
- **NIP (Nucleus Initialization Program)**
  - contains lots of RIMs  
*(Resource Initialization Modules)*
  - creates some system address spaces
- **Master Scheduler Initialization**
  - initializes address space number 1 (ASID=0001)
  - creates more system address spaces
- **Subsystem Initialization**
  - initializes subsystems e.g. JES2 or JES3
- **Process COMMNDxx member**



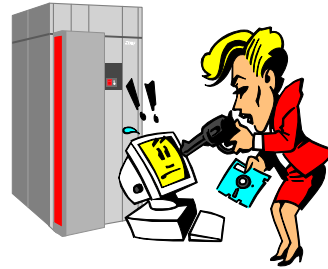
## Stopping the System

---

*Bring down system in reverse sequence to startup  
(manually or via automation)*

### Typical STOP sequence

- Various applications
- Transaction systems
- TCP/IP.....P TCPIP
- TSO.....P TSO
- VTAM.....Z NET,QUICK
- JES2.....£P JES2
- (OS/390).....Z EOD



## Unit Summary

---

- **Described z/OS initialization process**
  - LOAD
  - IPL (+IRIMs)
  - NIP (+RIMs)
- **Described Load Parameter specified for IPL**
- **Discussed LOADxx member of PARMLIB**
  - IODF statement
  - SYSCAT statement
  - SYSPARM statement
- **Described the startup of system and other address spaces**
- **Described the shutdown process**



## Summary

---

### Day 1

- Mainframe Hardware Overview
- z/OS High Level Overview
- TSO & ISPF for z/OS
- UNIX System Services
- Mainframe Access

### Day 2

- Hardware Management & Definition
- z/OS Operations
- Data Sets
- Using ISPF

### Day 3

- Virtual Storage and Address Spaces
- JES2 Introduction
- Introduction to JCL

### Day 4

- More on JCL
- The IPL Process