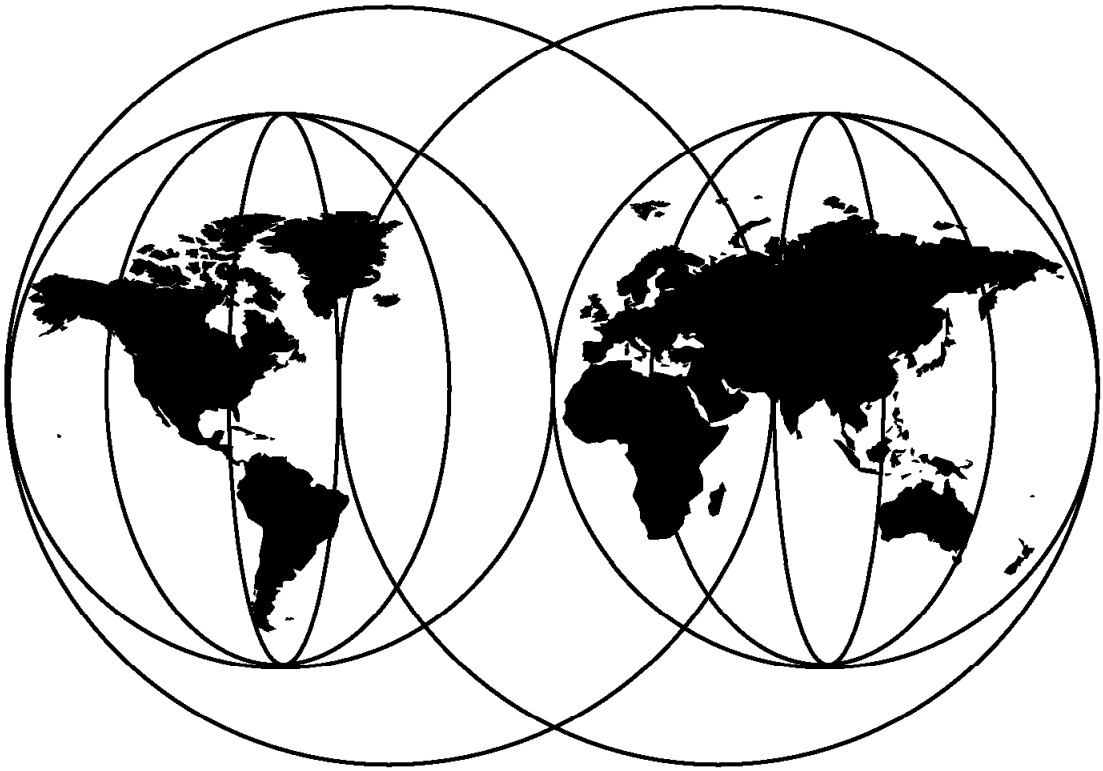
ABCs of OS/390 System Programming Volume 4

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



International Technical Support Organization

www.redbooks.ibm.com

SG24-5654-00

International Technical Support Organization

SG24-5654-00

ABCs of OS/390 System Programming Volume 4

April 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 349.

First Edition (April 2000)

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

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

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook is Volume 4 of a five-volume set that is designed to introduce the structure of an OS/390 and S/390 operating environment. The set will help you install, tailor, and configure an OS/390 operating system, and is intended for system programmers who are new to an OS/390 environment.

In this Volume, Chapter 1 provides an introduction to the basics of mainframe networking concepts, including hardware connectivity, OSA/SF, SecureWay Communications Server, SNA (VTAM), and IP (TCP/IP).

Chapter 2 describes OS/390 security with RACF.

Chapter 3 provides on overview of OS/390 UNIX System Services.

Chapter 4 describes the Language Environment which provides a common run-time environment for IBM versions of certain high-level languages (HLLs), namely, C, C++, COBOL, Fortran, and PL/I.

Chapter 5 describes the Infoprint Server which is an optional feature of OS/390 Version 2 Release 8 that uses OS/390 UNIX System Services. This feature is the basis for a total print serving solution for the OS/390 environment. It lets you consolidate your print workload from many servers onto a central OS/390 print server.

The team that wrote this redbook

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

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

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

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

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

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

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

Joao Natalino Oliveira

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

Fabio Chaves Pita

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

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

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

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

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 371 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Network Management

This chapter is an introduction to the basics of mainframe networking concepts, including hardware connectivity, OSA/SF and eNetwork Communications Server, SNA (VTAM), and IP (TCP/IP).

Included in this chapter are the following topics that will help you to:

- Understand the basics of mainframe connectivity
- Describe the various hardware and software options
- Distinguish between VTAM and TCP/IP
- Start and stop VTAM
- Identify and describe different VTAMLST members
- Understand the basics of SNA network definition
- Identify and describe different TCP/IP control files
- Understand the basics of TCP/IP
- Understand the basics of the OSA
- Set up OSA/SF
- Use OSA/SF to configure an OSA

Transmission Control Protocol/Internet Protocol (TCP/IP) is a set of industry-standard protocols and applications that enable you to share data and computing resources with other computers, both IBM and non-IBM. By using TCP/IP commands at your workstation, you can perform tasks and communicate easily with a variety of other systems and workstations. SecureWay Communications Server for OS/390 (CS for OS/390) enables the user to interactively run TCP/IP applications (TCP/IP commands) from both the Time Sharing Option (TSO) and the OS/390 shell.

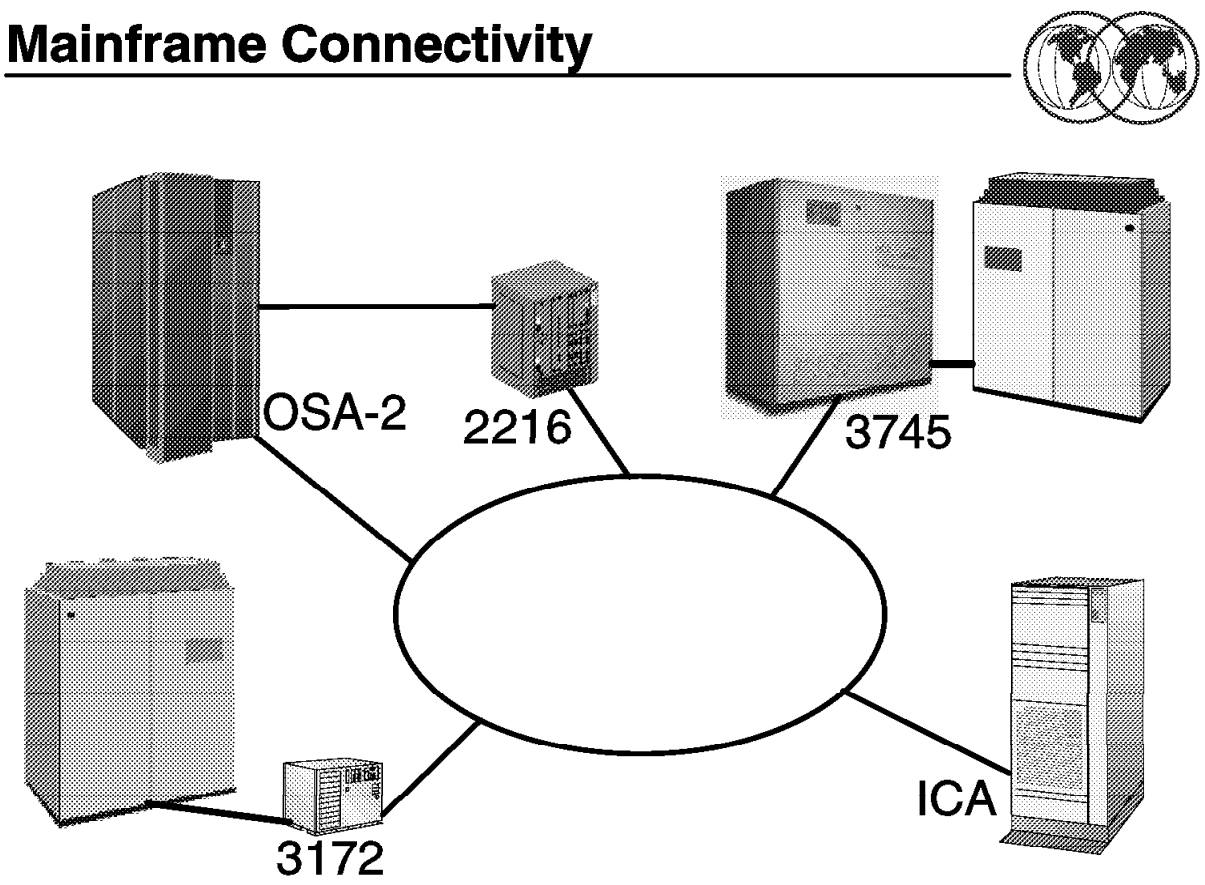


Figure 1. Mainframe connectivity overview

1.1 Mainframe connectivity overview

In order to communicate on any network, the mainframe must be attached to that network. A physical network consists of electrical wiring and components, such as modems, bridges, controllers, access units, telephone lines, fiber optic cables, and co-axial cables. These are used to connect the computer nodes together. The physical network can connect two nodes in a single room or thousands of nodes communicating across large geographic areas. The most common networks in use today are Local Area Networks (LANs) and Wide Area Networks (WANs). LANs cover a limited distance, generally one or two floors or buildings, while WANs, using telecommunication facilities, are used for longer distances.

Network protocols are the rules that define how information is delivered between nodes. They describe the sequence and contents of the data exchanged between nodes on the network. Network protocols determine how a computer node functions during communication with another node, how data is enclosed to reach its destination safely, and what path it should follow. Protocols coordinate the flow of messages and can specify which node a message is destined for in the network. A variety of protocols are used to take advantage of the characteristics of each of the physical network types. The most common protocols are Ethernet, 802.3, Token-Ring X.25, IP, and System Network Architecture (SNA).

S/390 supports the following types of network devices:

- 3172 LAN Channel Station (LCS)

- Channel-to-channel (CTC)
- Common link access to workstation (CLAW)
- ATM
- HYPERchannel A220
- MPCPTP
- OSA-Express (MPCIPA)
- SNA LU0 Links
- SNA LU 6.2 Links
- X.25 NPSI Connections
- Virtual Devices (VIPA)
- 3745/46 Channel DLC

The most common way to attach a S/390 processor to a network is via the following communication controllers:

- IBM 3172 Interconnect Controller for Token-Ring, Ethernet, fiber distributed data interface (FDDI) and asynchronous transfer mode (ATM).
- IBM 3174 Establishment Controller for Token-ring, Ethernet, Frame Relay, X.21/X.25 Switched Autocall/Autodisc, X.35/X.21, and up to 64 3270 ports.
- IBM 3745/3746 Multiprotocol Controller for Token-Ring, Ethernet, ATM, Fast Ethernet, FDDI, lines up to 2Mbps, HSSI(T3/E#) and WW primary ISDN,
- IBM 2216 Multiaccess Connector model 400 for WAN connection speeds from 9.6 Kbps to 52 Mbps (HSSI), interface attachments (V.35/V.36, X.21, X.24, V.25bis), and data link controls, including X.25, SDLC, ISDN Primary, Frame Relay, PPP,. FDDI, 10/100 Mbps Ethernet, and ATM both MM and SM.
- OSA-2 or OSA-Express for Token-Ring, Ethernet, Gigabit Ethernet, FDDI, Fast Ethernet, ATM 155 Mb Multi Mode, ATM 155 Mb Single Mode.
- IBM 3274 for local SNA for coax or LAN-attached SDLC.

Old CPU types, such as the IBM 9221 or the IBM 9370, have an Integrated ′ Channel Adapter for SNA connectivity.

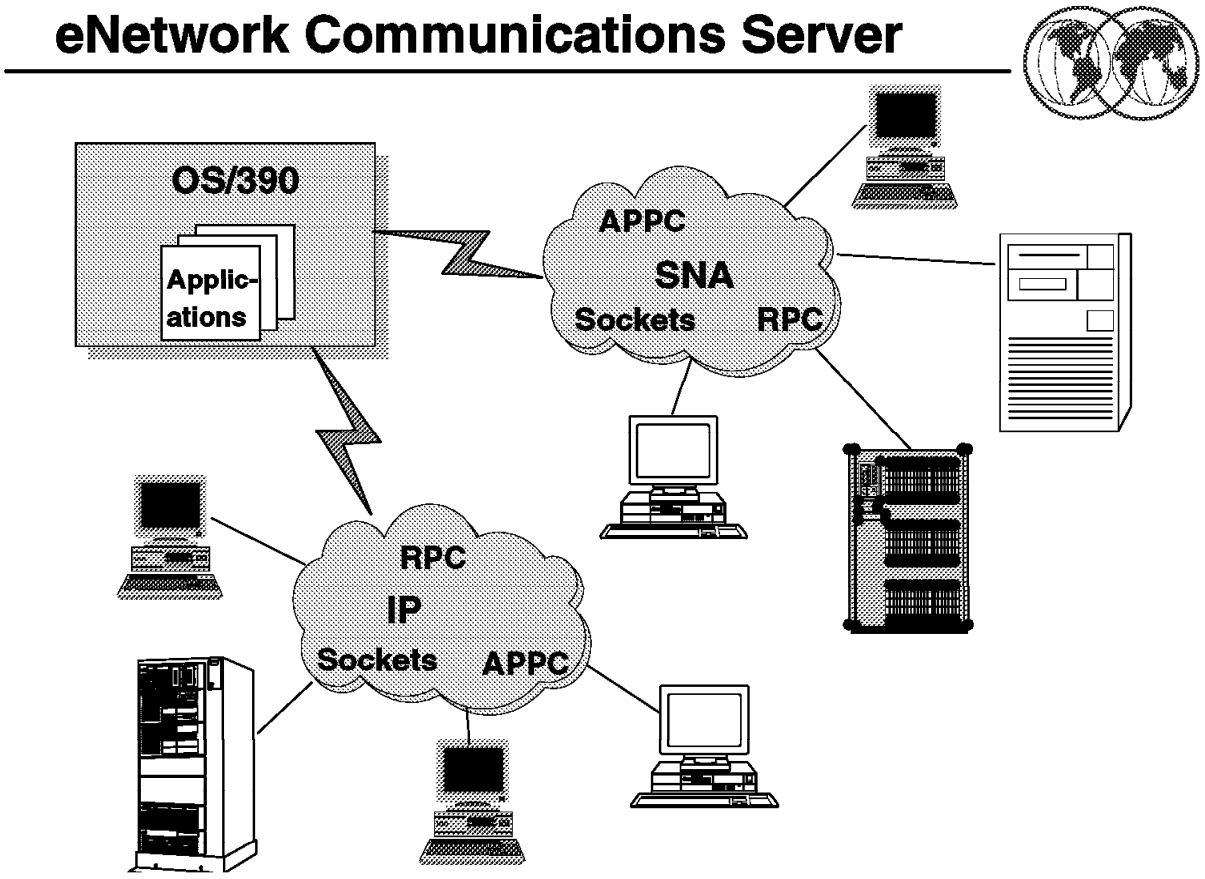


Figure 2. OS/390 eNetwork Communications Server

1.2 eNetwork Communications Server

The eNetwork Communications Server for OS/390 provides the networking foundation for S/390 e-business. Users can access S/390 application data over SNA, TCP/IP or mixed networks, WANs and LANs, and a wide variety of connection types such as frame relay, gigabit Ethernet, and ATM to connect their employees, suppliers, customers or business partners worldwide. In other words, it provides end-to-end universal connectivity.

It includes a wide variety of programming interfaces, such as sockets, remote procedure call (RPC), and APPC using wide area network protocols provided by IP and SNA. The eNetwork Communications Server SNA element also includes Advanced Peer-to-Peer Networking (APPN), which is an extension to SNA and offers enhanced functions suitable for doing client/server and cooperative processing in mixed LAN/WAN networks.

The eNetwork Communications Server IP stack services have been completely rewritten in OS/390 R5 to provide significantly improved performance.

The support for Multiprotocol Transport Networking (MPTN) by the OS/390 network services provides the ability to have UNIX applications communicating over a SNA network, or have APPC (SNA) applications communicating over a TCP/IP network. This enables application program types to communicate, without change, over different transport networks and across interconnected networks.

OS/390 R5 provides a facility called High Speed Web Access (HSWA), which is intended for users with high-demand Web-serving requirements.

The integrated Communications Server make it possible for a S/390 server to manage and share information and transactions across different system platforms and multivendor networks.

The OS/390 operating system includes communication services. These services are essential for a server operating system, and enables support for open, distributed computing services.

Traditionally, MVS applications have communicated with other applications on MVS, VM, OS/400, or OS/2 platforms using an SNA network. The application interface has been the Advanced Program to Program Communication (APPC). APPC builds on the Common Programming Interface for Communication (CPI-C).

UNIX applications can communicate with other UNIX applications using the RPC or socket interfaces over a TCP/IP network. TCP/IP has been supported on MVS for quite a few years. However, the UNIX support on MVS (UNIX Services) makes TCP/IP more important than before. Both SNA and TCP/IP support is included in OS/390.

The MPTN architecture provides the capability to use any program interface over any network protocol. The AnyNet feature of VTAM allows sockets and RPC calls to be transferred over a SNA network, and TCP/IP allows APPC calls over a TCP/IP network. The only limitation is that a program interface must communicate with the same interface (APPC to APPC, socket to socket, and RPC to RPC).

The terminal input output controller (TIOC) is the interface between TSO and VTAM. It allows TSO to communicate with the terminal hardware.

The VTAM element in OS/390 includes Advanced Peer-to-Peer Networking (APPN) support which provides a dynamic way of connecting nodes in a network with a minimal amount of system definition. The nodes must be of type 2.1 which means a programmable node, for example, a PC workstation, an AS/400, a RISC/6000, or a VTAM node. High Performance Routing (HPR) is an addition to APPN, often written APPN/HPR. HPR enhances data routing performance and reliability. It increases the performance of an APPN network by reducing the processing required in intermediate nodes. The intermediate node passes data to the next node without examining any session identifier or performing pacing. (*Pacing* is a technique by which a receiving node controls the rate of transmission of a sending node to prevent overrun.)

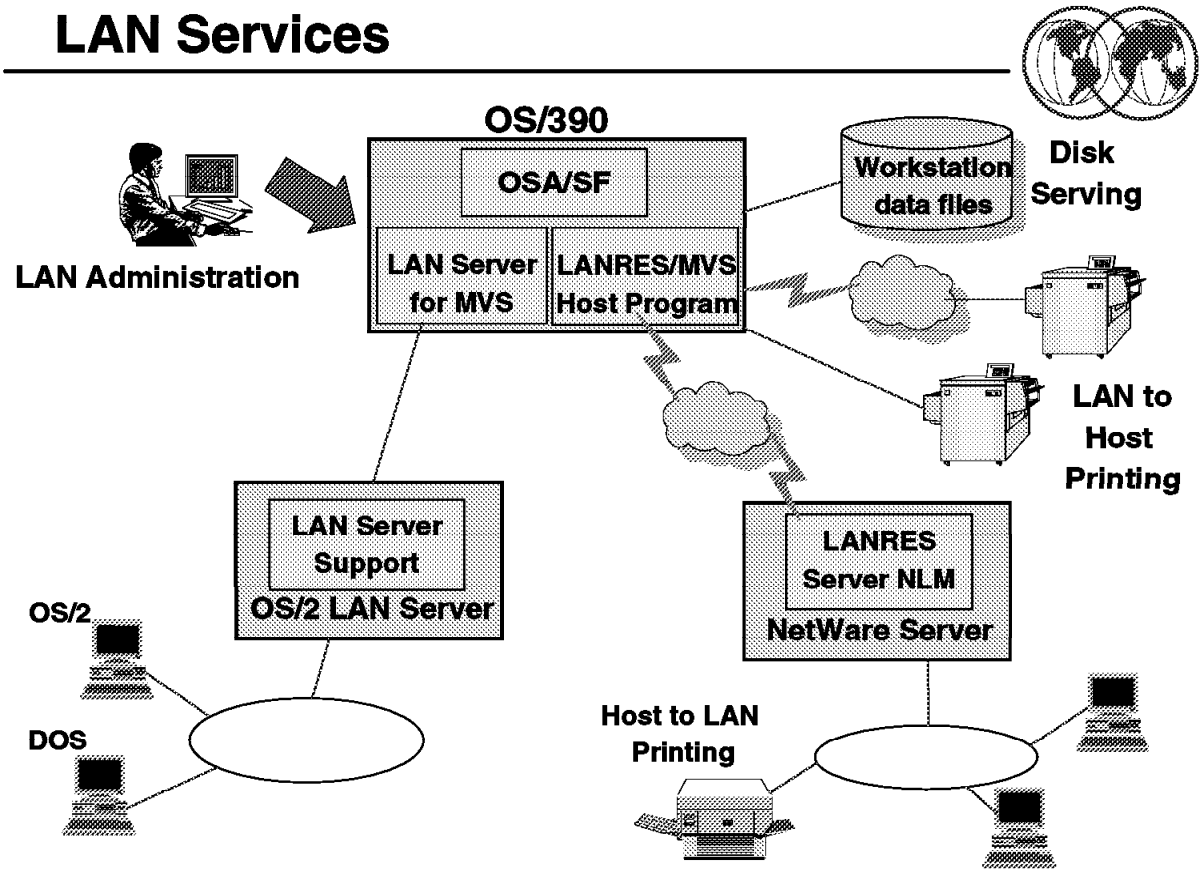


Figure 3. LAN Services

1.3 eNetwork Communications Server

OS/390 includes local area network (LAN) services which allow LAN users to store and access data on an S/390 system, and to use the print services of OS/390. These services provide central management, high capacity, and high performance for a distributed computing solution.

The LAN services are provided by:

- LANRES/MVS provides disk serving, bidirectional print serving, data distribution, and central administration for Novell NetWare LANs. It also supports the IBM AFP Printer Driver for Windows.
- LAN Server provides an OS/2 high-performance file serving system to OS/2 WARP and NFS LAN clients connected to the S/390 Servers. In conjunction with the OS/2 Ultimedia product, LAN Server supports end-to-end Quality-of-Service (QoS) multimedia delivery to OS/2 LAN Server and WARP clients in a timely, useable and quality manner. It provides the capability for OS/2 LAN Server and NFS clients to share a common data repository with full update capability. The LAN Server for OS/390 R3 is enhanced in the areas of availability, scalability, performance, inter-operability, security, and administration to support the needs of large user workstation configuration growth. One of these enhancements is to the managed access function which provides for a server to dynamically control access to OS/2 or NFS files stored by LAN Server by using a token generated by the application server. The workstation client, using the token, can then access the file, overriding OS file attributes or NFS permissions. Also, the LAN Server for OS/390 provides an open interoperability for UNIX application s to access and utilize LAN user data in an integrated client server heterogeneous environment.

The OS/390 LAN services are based on LANRES/MVS and LAN Server for MVS. These products are included in the OS/390 system and are part of the base elements.

- LANRES/MVS provides server capabilities for Novell NetWare LANs. The services include data serving, print serving, and central management of the NetWare LANs. With LANRES/MVS a single NetWare server can communicate with a VM, MVS, and OS/400 operating system at the same time.
- LAN Server for MVS provides disk server capabilities and central management for Token Ring LANs and Ethernet LANs. It interacts with two other LAN server products: OS/2 LAN Server which serves OS/2 and DOS clients on Token Rings, Network File System (NFS) clients on Ethernet LANs. LAN Server allows sharing of data between OS/2 LAN Servers and NFS clients. LAN Server does not provide any print services.

The OSA feature is available for 9121 511-based and 9021 711-based ES/9000 processors and for the 9672 S/390 Parallel Server (CMOS). The OSA feature allows connection to a local LAN without using a communication controller (37x5 or 3172). This makes a LAN connection less expensive and easier to implement. For large LAN environments with high performance requirements, and for any remote LAN connection, a communication controller is needed.

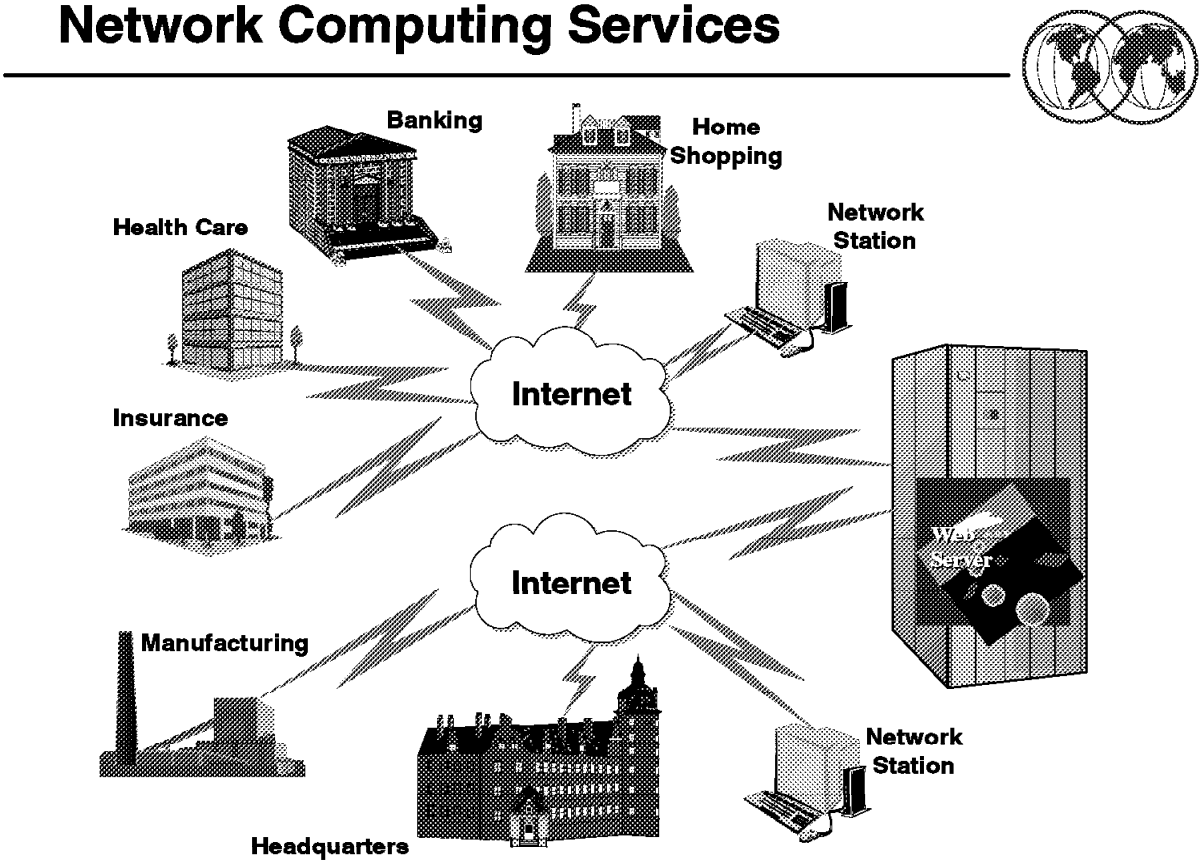


Figure 4. OS/390 Network Computing

1.4 Network Computing Services

OS/390 provides Lotus Domino Go Webserver for OS/390 which includes the Internet Connection Security Server (ICSS). This allows the OS/390 system to become a World Wide Web server with a data repository for text, images, sound, and video clips stored in the hierarchical file system (HFS).

The Domino Go Webserver for OS/390 also includes NetQuestion which is a powerful, full-text indexing and search server.

The Internet Connection Security Server (ICSS) for MVS is based on UNIX Services services. The Internet Connection Security Server is delivered as a no-cost feature with Lotus Domino Go Webserver for OS/390.

The OS/390 Internet Bonus Pak II is a set of sample HTML pages, programs, and redbooks that demonstrate how to:

- Retrieve data from DB2, IMS, and CICS databases and present this data to a client browser
- Write simple Hypertext Markup Language (HTML) pages
- Write simple Common Gateway Interface (CGI) programs
- Write simple Java programs
- Write simple DB2 WWW macros
- Use secure sockets
- Use Server Side Includes

Physically, the sample HTML pages and executables reside in directories in the UNIX Services MVS Hierarchical File System (HFS).

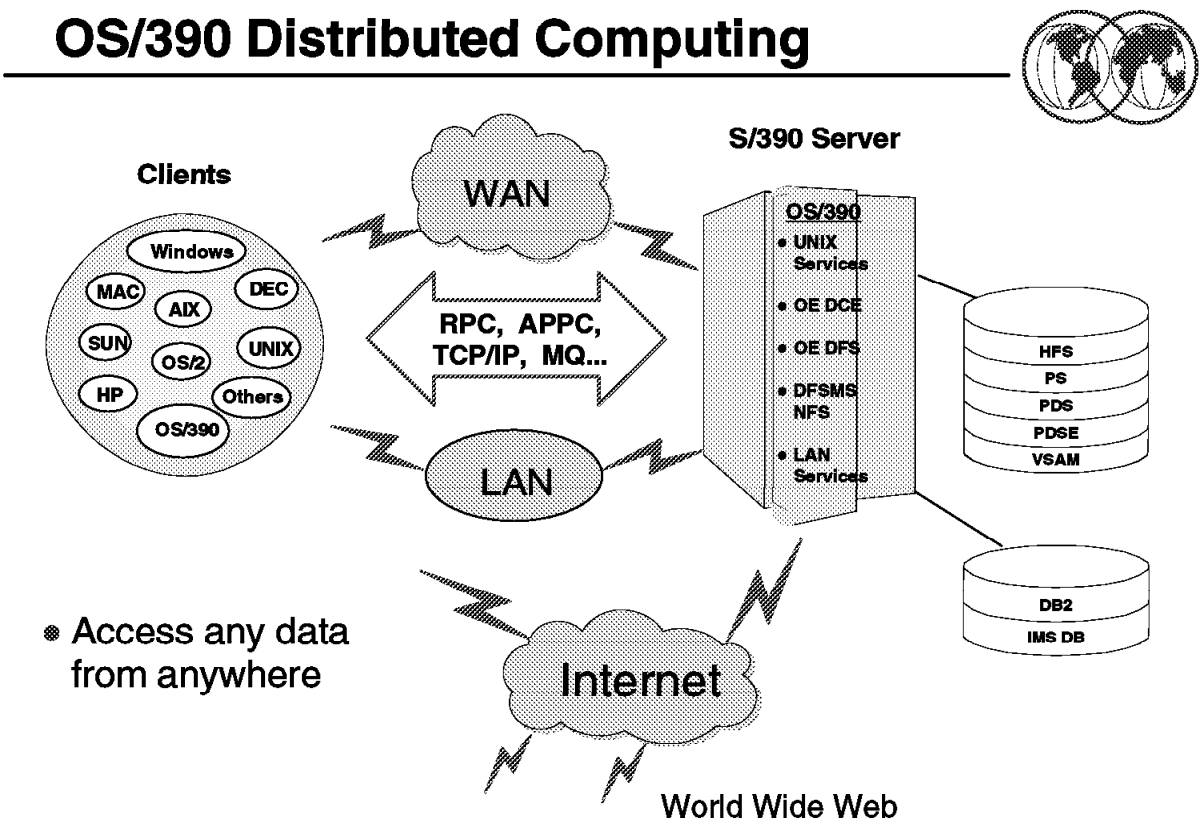


Figure 5. OS/390 Distibuted Computing

1.5 OS/390 Distributed Computing

OpenEdition Distributed Computing Environment (DCE) provides support for the open client/server technology from Open Software Foundation (OSF) called DCE. DCE is a set of services and tools that support the creation, use, and maintenance of distributed applications in a heterogeneous environment.

Support for distributed file systems are provided by:

- The Distributed File System (DFS) is part of the DCE solution and provides transparent access to remote files from anywhere in the DCE network. It allows users to easily share data in a distributed environment.
- The Network File System (NFS) is a solution which allows users to share and access data in a heterogeneous environment. It is widely used in a UNIX environment. NFS provides workstation access to data sets on an OS/390 system.

OS/390 provides the Internet Connection Security Server (ICSS) as a no-cost Feature of OS/390 Distributed Computing Services and it is automatically shipped with OS/390. This allows the OS/390 system to become a World Wide Web server with a data repository for text, images, sound, and video clips stored in the hierarchical file system (HFS). The OS/390 Internet BonusPak II is also shipped automatically, but separately, and it must be installed after the ICSS.

The OS/390 system includes OpenEdition DCE which is a solution for distributed computing. DCE is supported by the Open Software Foundation (OSF), which looked at multiple open distributed solutions from multiple vendors and selected the best functions to be integrated into one solution called DCE.

The OpenEdition DCE for OS/390 consists of:

- OpenEdition DCE Base Services (base element in OS/390)
- OpenEdition DCE Security Server (optional feature)
- OpenEdition DCE Distributed File Server (base element)
- OpenEdition DCE Application Support (optional feature)

Distributed applications use a client/server model based on Remote Procedure Call (RPC) to provide connectivity from client application through the network to the server application. A quick overview of the DCE client/server model: A client application uses RPC to call a service (function) from a server, RPC uses the services of the DCE Distributed Directory to locate the server, Distributed Security is invoked to authenticate both the client and the server, Distributed Time Services will synchronize the client and server processing, and threads allows parallelism in execution.

DFS is a distributed client/server application which uses the DCE services. DFS provides access to the OpenEdition HFS or to local DFS file systems. Some of the advantages of DFS over similar distributed file system solutions is that DFS provides caching of data which reduces network load and improves performance. Data can be replicated (shadow copies) across multiple servers, resulting in improved reliability and availability.

NFS is a solution well known in the UNIX environment. The DFSMS/MVS NFS feature now provides both server and client support. This enables one OS/390 system to exchange data files with another using NFS client/server technology. Each OS/390 system with the DFSMS/MVS 1.3 NFS feature can read from or write to any other OS/390 system which has the DFSMS/MVS NFS feature installed. DFSMS/MVS NFS enables OS/390 UNIX applications on one OS/390 system to read or write MVS conventional data sets and/or OS/390 UNIX Hierarchical File System (HFS) files on another OS/390 system.

DFS and NFS provide file server capabilities for OS/390 in a distributed heterogeneous environment. The LAN services previously introduced are limited to LANs, while NFS and DFS are available for use on both LANs and WANs (TCP/IP, VTAM AnyNet).

The Internet Connection Security Server (ICSS) for MVS is based on OS/390 UNIX services. The Internet Connection Security Server is delivered as a no-cost feature with OS/390.

The OS/390 Internet Bonus Pak II is a set of sample HTML pages, programs, and redbooks that demonstrate how to:

- Retrieve data from DB2, IMS, and CICS databases and present this data to a client browser
- Write simple Hypertext Markup Language (HTML) pages
- Write simple Common Gateway Interface (CGI) programs
- Write simple Java programs
- Write simple DB2 WWW macros
- Use secure sockets
- Use Server Side Includes

Physically, the sample HTML pages and executables reside in directories in the OS/390 UNIX MVS Hierarchical File System (HFS).

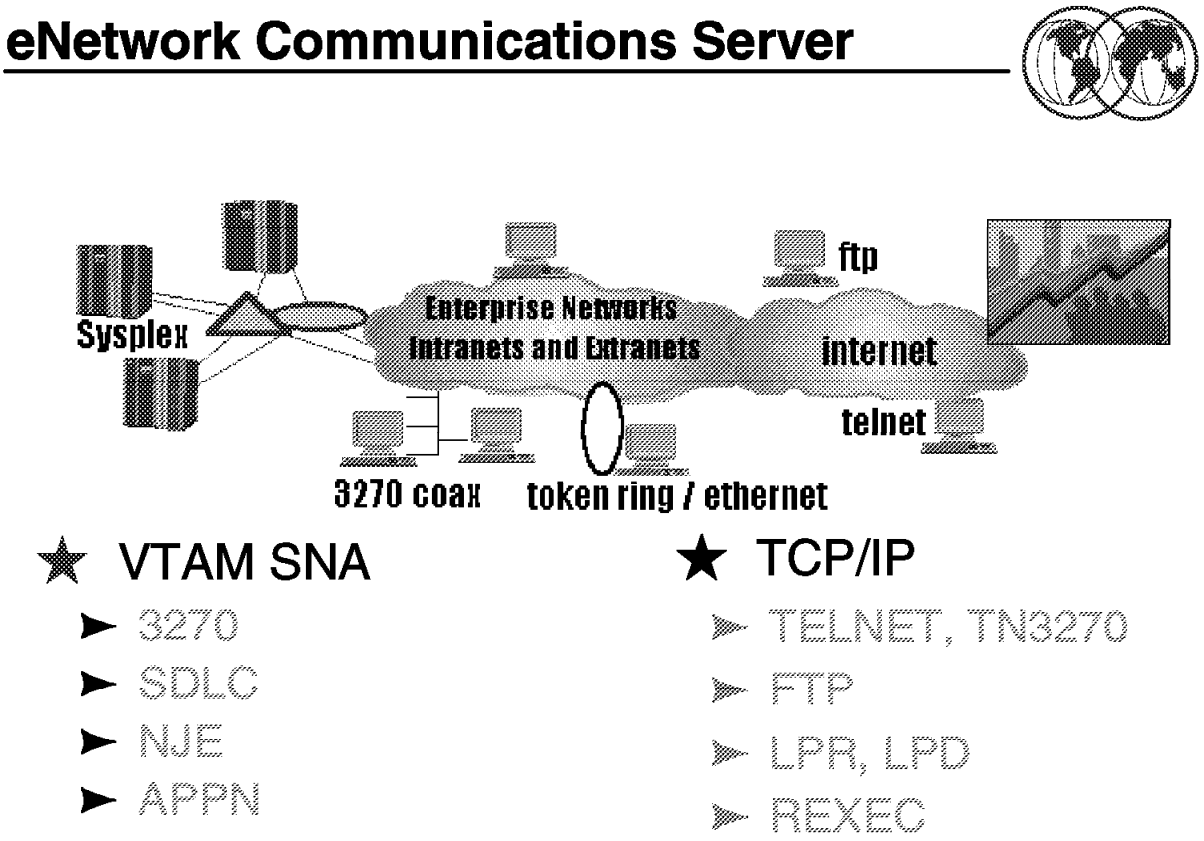


Figure 6. eNetwork

1.6 Networking Products

Until OS/390 V2R5, IBM marketed two separate networking products:

- Virtual Telecommunications Method (VTAM)
- Transmission Control Protocol / Internet Protocol (TCP/IP)

VTAM has been around for some time and forms the backbone of IBM′s Systems Network Architecture (SNA) networking protocol including basic connectivity such as 3270, SDLC and NJE.

TCP/IP is a relative newcomer and is the basis for OS/390′s presence on the WWW. It enables FTP and Telnet (both client and server), as well as webserving, LPR, and LPD, to name a few.

As of OS/390 Version 2 Release 6, both products were merged under the banner of eNetwork Communications Server. However, they still remain functionally separate components as eNetwork Communications Server SNA and eNetwork Communications Server IP.

1.6.1 VTAM

VTAM provides a method by which application programs can communicate with telecommunication devices and their users. VTAM was the first IBM program to allow programmers to deal with devices as “logical units” without having to understand the details of line protocols and device operation. Prior to VTAM, programmers used IBM′s Basic Telecommunications Access Method (BTAM) to communicate with devices that used the binary synchronous (BSC) and start-stop line protocols.

1.6.2 SNA

Systems Network Architecture (SNA) is IBM′s proprietary network architecture and set of implementing products for network computing within an enterprise. It existed prior to and became part of IBM′s Systems Application Architecture (SAA) and it is currently part of IBM′s Open Blueprint. With the advent of multi-enterprise network computing, the Internet, and the industry standard open network architecture of TCP/IP, IBM is blurring the edges between VTAM/SNA and TCP/IP application support via the enterprise-wide product eNetwork Communications Server.

SNA itself contains several functional layers and includes a communications protocol for the exchange of control information and data, and a data link layer, Synchronous Data Link Control (SDLC). SNA includes the concepts of nodes that can contain both physical units that provide certain setup functions and logical units, each associated with a particular network transaction.

1.6.2.1 SDLC

Synchronous Data Link Control (SDLC) is a transmission protocol developed by IBM in the 1970s as a replacement for its binary synchronous (BSC) protocol.

SDLC became part of IBM′s Systems Network Architecture (SNA) and the more comprehensive Systems Application Architecture (SAA) and its more recent Open Blueprint. SDLC is still commonly encountered and probably the prevalent data link protocol in today′s mainframe environment.

It utilizes the concept of a Primary and a Secondary partner in any communications link, referred to as the Primary Logical Unit (PLU) and the Secondary Logical Unit (SLU). Typically in IBM mainframe networks, the host mainframe is the PLU and workstations, for example a 3270 device, and other devices are SLUs. Each SLU has its own address in the network, thus enabling it to be identified individually. Typically, multiple devices or SLUs are attached to a common line in what is known as a multipoint or multidrop configuration. SDLC is primarily intended for remote communication on corporate wide-area networks (WANs).

1.6.2.2 APPN

Advanced Peer to Peer Networking (APPN) protocol was developed by IBM to support computer-to-computer (not necessarily 3270) communications, often referred to as LU6.2. It is a non-hierarchical protocol, that is, either end can initiate a session based around the concept of:

- End Nodes - The application programs, either clients or servers
- Network Nodes - Used to route information around the network between end nodes.

1.6.2.3 3270

3270 is the session protocol used to establish a screen connection to a System/390 mainframe.

It started life as the traditional 24x80 *green screen* and developed into the more usable, but still essentially *dumb* screens or terminal emulators we use today:

- The 3270 model 2 24x80
- The 3270 model 3 32x80
- The 3270 model 4 43x80
- The 3270 model 5 27x132
- Extended Attribute Support (colors, blink, etc.)

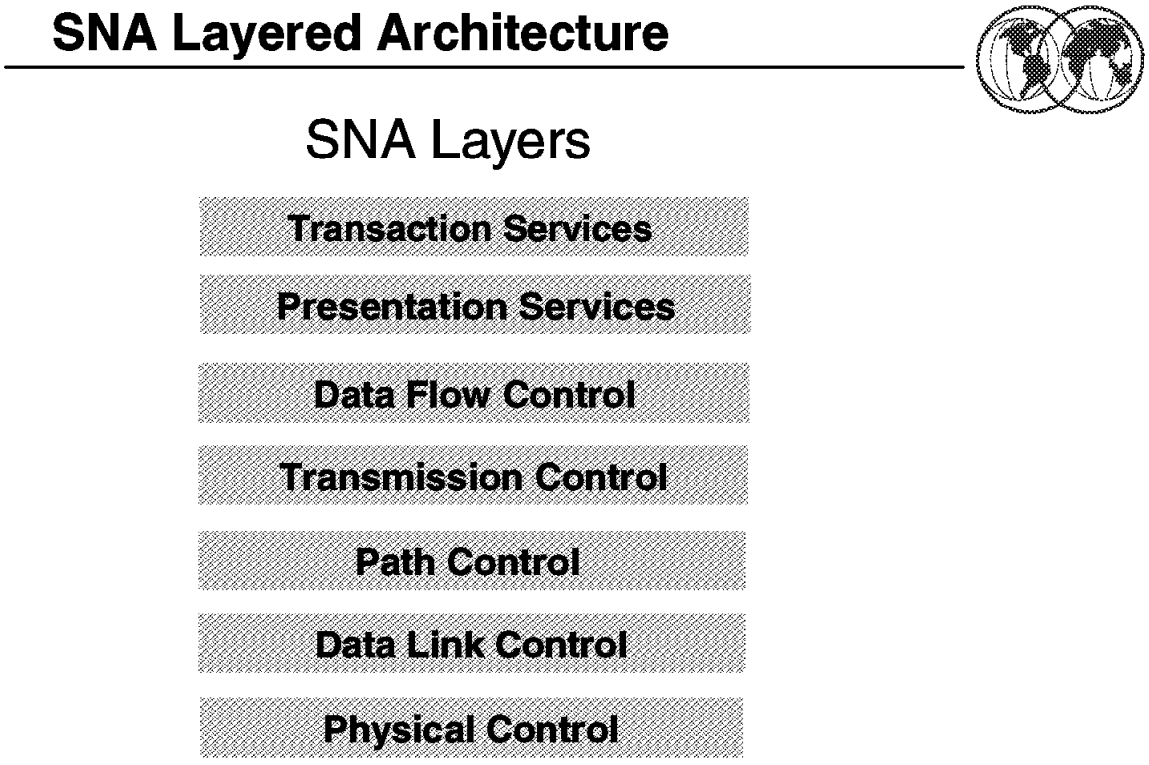


Figure 7. SNA Layered Architecture

1.7 SNA Layered Architecture

This section provides an overview of SNA, including its architectural objectives, network components, data transport services, transaction services, and network management services.

A *data communication network* is a collection of hardware and software components that enable end users to exchange data. To send or receive data through a network, end users interact with communication devices such as telephones, terminals, or computers. The term *end user* is used to identify both (1) individuals who interact with the network through a workstation and (2) application programs. The architecture views end users as the ultimate sources and destinations of information that flows through a network.

Data communication requires that network components agree on both the layouts of the messages they exchange and the actions they take based on the kinds of data they receive. The layouts are referred to as *formats*, and the actions taken are referred to as *protocols*. Formats and protocols together constitute an *architecture*.

Systems Network Architecture (SNA) is a data communication architecture established by IBM to specify common conventions for communication among the wide array of IBM hardware and software data communication products. The manner in which products internally implement these common conventions can differ from one product to another. But because the external interface of each implementation is compatible, different products can communicate without the need to distinguish among the many possible product implementations.

SNA functions are divided into a hierarchical structure that consists of seven well-defined layers. Each layer in the architecture performs a specific set of functions. In the graphic you can see the SNA's seven layers and their major functions.

SNA defines formats and protocols between layers that permit equivalent layers (layers at the same level within the hierarchy) to communicate with one another. Each layer performs services for the next higher layer, requests services from the next lower layer, and communicates with equivalent layers. To illustrate this concept, consider end-user data that requires encryption. The two transmission control layers encipher and decipher data independently of the functions of any other layer. The transmission control layer in the originating node enciphers the data it receives from the data flow control layer. It then requests that the path control layer route the enciphered data to the destination node. The transmission control layer in the destination node deciphers the data that the path control layer delivered. It then requests that the data flow control layer give the deciphered data to the destination end user.

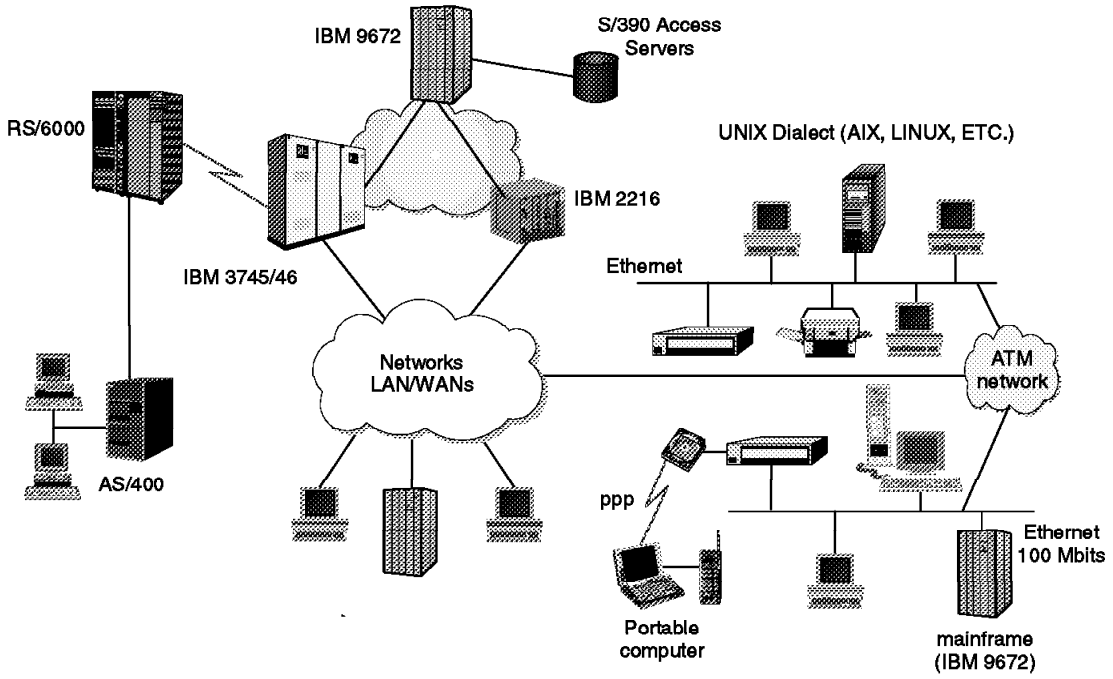


Figure 8. Hardware and software components

1.8 Hardware and software components

Hardware and software components implement the functions of the seven architectural layers. Hardware components include:

- Processors such as the ES/9000(*) family
- Distributed processors such as the Application System/400(*)
- Communication controllers such as the 372X and 374X series
- Cluster controllers
- Workstations
- Printers

The software components that implement SNA functions include:

- Operating systems such as Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA(*)), and Operating System/400(*) (OS/400(*))
- Telecommunication access methods such as the Virtual Telecommunications Access Method (VTAM(*)) and Communications Manager/2 (CM/2(*))
- Application subsystems such as Customer Information Control System (CICS)
- Network control programs such as the Advanced Communication Function for Network Control Program (NCP)

The graphic illustrates one possible network configuration of these hardware and software components.

The Network Blueprint

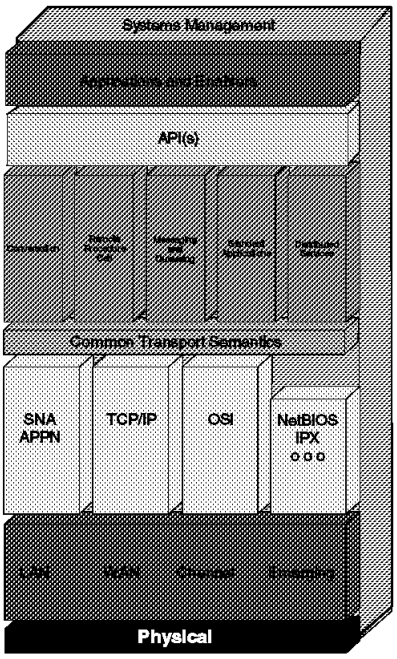


Figure 9. The network blueprint

1.9 The Network Blueprint

IBM′s networking design strategy employs a “Networking Blueprint” that provides an open, highly modular framework for structuring networks using industry-wide standards. The blueprint is designed to facilitate the creation of an evolutionary and flexible plan that an organization can specifically tailor to meet its needs; this is much needed considering today′s fast-moving technologies and the diversity of networks. The graphic shows the modules used to describe the Networking Blueprint.

The Networking Blueprint supports the implementation of multiple protocols, as well as integrating these protocols into a cohesive, modular structure. The Networking Blueprint defines layers of functions, plus a systems management backplane. SNA Advanced Peer-to-Peer Networking(*) SNA Advanced Peer-to-Peer Networking (APPN), is part of the transport layer, and it is one of the protocols that can be used in this layer.

APPN′s any-to-any connectivity makes it possible for large and small networks alike to communicate over local- and wide-area networks, across slow and fast links.

APPN provides two basic functions: keeping track of the location of resources in the network, and selecting the best path to route data between resources. APPN nodes dynamically exchange information about each other; therefore, customers may never have to deal with complicated system and path definitions. APPN nodes limit the information they exchange, enabling more efficient use of network resources.

High-performance routing (HPR) is a small but powerful evolutionary extension to APPN. It enhances data routing performance via decreasing intermediate node processing. HPR also increases session reliability via “nondisruptive path switch.” The two main components of HPR are rapid-transport protocol (RTP) and automatic network routing (ANR).

Systems Network Architecture was designed with certain objectives in mind. These objectives address common concerns of data communication network users. One such concern is the reliability of the network. Recoverable data communication errors must be handled transparently to the user. At the data link layer of the architecture, error-checking protocols ensure that error detection and message retries are performed automatically. When errors are not recoverable, communicating partners must be able to achieve a mutual understanding as to the results of attempted message transfers. Protocols at the presentation services layer provide positive-negative responses and establish synchronization points, which enable communicating partners to ensure the consistency of their resources.

For some applications, consistent, fast performance is needed to handle updates immediately; for other applications, it is important to send data as inexpensively as possible. APPN includes several features that can handle this type of traffic mix efficiently. APPN′s priority services ensure that important data moves through the network quickly. Similarly, by using intelligent class-of-service routing, APPN nodes consider factors such as security, cost, delay, and throughput to select the best route for different types of data. Unlike other protocols that react to network bottlenecks by dropping and resending packets, APPN avoids network congestion by using adaptive pacing, which ensures a higher, more consistent traffic volume.

High-performance routing (HPR) improves on APPN′s reliability by providing greater session reliability in case of link and node failure via nondisruptive path switch. It is accomplished transparently to both the sessions and end users.

An SNA network is dependable because SNA products recognize and recover from loss of data during transmission, use flow control procedures to prevent data overrun and avoid network congestion, identify failures quickly, and recover from many errors with minimal involvement of network users. SNA products also increase network availability through options such as the extended recovery facility, backup host, alternate routing capability, and maintenance and recovery procedures integrated into workstations, modems, and controllers.

Another concern is the efficiency with which data is transferred. Components within SNA act to promote efficiency both by selecting options that maximize data transmission throughput and by taking action to reduce network congestion when it is detected. Route selection services, for example, can select optimal routes for data transmission.

When too much data is introduced into the network, congestion can occur. Severe congestion blocks the flow of messages and can result in the loss of data. Protocols at the transmission control and path control layers prevent overload and deadlocks by controlling the pace at which messages flow through the network.

APPN features help eliminate unnecessary network control traffic, thus providing more bandwidth for moving data through the network. APPN nodes never broadcast changes to all machines in the network. Instead, only network nodes exchange topology information, and they exchange this information only when changes occur in the backbone of the network. Other protocols broadcast routing information at frequent regular intervals, even if nothing changes in the network. Other APPN features, like directory caching and central directory servers, limit searches for other resources in the network, and, as a result, improve network performance. In summary, APPN permits more bandwidth for real work for applications, without the need to invest in new equipment.

An ongoing objective of SNA is ease of use, both for end users and network personnel. Because SNA products have compatible interfaces, they can connect and communicate with one another. Application program interfaces at the presentation services layer, for example, shield end users from

concern with the underlying details of communication protocols. When hardware and software upgrades are required, the functional independence of the SNA layers enables any one layer to be enhanced or modified without disrupting the functions of any other layer. To link independent networks, network interconnection protocols are provided that enable the networks to communicate without redefining network identifiers. An example of this is Common Programming Interface for Communications (CPI-C), which can protect an organization′s investment in application programming. The services provided by APPN include network topology updates and automatic route selection, which make it unnecessary to predefine node locations and routes between nodes. Thus, network operators can add new components to the network without affecting the network availability for existing users.

With peer-to-peer communication protocols, end users benefit through more timely access to their applications because they are less dependent on the system programming staff to code network definitions. As workstations and applications first become available in the network, or are moved, APPN directory functions locate them dynamically without requiring coordinated definitions to be coded by the system programmer. This improves network availability and end-user productivity.

Resource sharing addresses the concern for fair and effective resource utilization. The sharing of access to storage devices, output devices, and data communication lines is paramount for containing network costs. Resource sharing is addressed at many levels within SNA, from the multiplexing of data links to the sharing of sessions by individual end-user transactions.

At the same time other mechanisms control the equitability with which services are provided to network users. Transmission priorities and classes of service enable equal service to be given to sessions of equal priority, or preference given to sessions of higher priority.

The security of data is an increasing need for today′s networks, which have become the vehicles for such sensitive data as banking transactions. Protocols for data encryption at the transmission control layer, and password verification at the transaction services layer and the presentation services layer, serve the data security objective.

Tools for resource management provide the ability to identify errors, help in problem determination, and maintain accounting data on network resource usage. Capabilities exist within all layers of SNA for monitoring and reporting errors relating to the functions for which they are responsible. Usage statistics facilitate fair charging of network users, performance tuning, capacity planning, and capital budgeting.

IBM offers communication products that conform to SNA specifications. Because of IBM′s ongoing development of products compatible with this architecture, organizations often can substitute one type of SNA product for another as their needs change. In an SNA network, newer devices with improved capabilities can coexist with older ones. As an organization′s SNA network evolves with the addition of new workstations, processors, communication facilities, and applications, it can continue using the applications already in place.

- Subarea Networks

  Networks that use peer-to-peer communication protocols can be integrated with subarea networks, allowing application sessions between peer-to-peer nodes and hierarchical nodes.

- Dependent LU Support

  Users of APPN can carry their investment in dependent LUs into the future of dynamic Advanced Peer-To-Peer (APPN) networking. Dependent LUs can continue to access VTAM applications in the same VTAM domain, or in different domains, using subarea protocols. In addition, dependent LUs in one domain can access VTAM applications in a different domain utilizing APPN protocols between domains. This allows the user to begin implementing APPN instead of subarea protocols between VTAM domains, while continuing to provide access to VTAM applications from dependent LUs in different domains.

- Applications

  Existing applications continue to be supported for dependent LUs and independent LUs. There is no change to APIs.

- Ease of Migration

  Migrating to a client/server environment does not have e to be complex, time-consuming, and expensive. Advanced Peer-to-Peer Networking (APPN) provides a better solution for migrating both networks and applications. With APPN, a network can be migrated to the client/server environment at any pace. The new version of VTAM, for example, allows traditional SNA networks to be integrated with APPN networks. APPN also enables running both existing mainframe applications and new client/server applications at the same time.

SNA helps find and resolve problems that can occur in a network. SNA management services functions in each network component to assist in problem determination. Also, the NetView(*) family of products, which is network management software that runs on a host processor, has components that monitor, collect, and store network data. Other system management functions built into the NetView program allow rapid problem determination, increase network availability, and minimize the number of the personnel needed to operate and maintain the SNA network.

An SNA network accommodates such facilities and technologies as:

- Digital networks
- Digitized voice
- Distributed systems
- Fiber optics
- Graphics
- Public packet-switching data networks
- Satellites
- Token-ring networks
- Videotex
- Client/server
- Ethernet
- Frame relay
- Security.

The SNA Network Interconnection (SNI) facility helps exchange information with another independent organization or merge separately-administered subarea networks into one.

The SNA Network Interconnection facility allows users in one SNA network to access information and application programs in other SNA networks. Each interconnected network can maintain its existing management procedures and controls. A “gateway” between two or more SNA networks connects them operationally while isolating their administrative characteristics from one another. Network users are not aware of network boundaries.

Two APPN subnets may communicate with each other via a *border node*. The border node allows sessions between users in different APPN subnets while isolating the subnets′ topology information from one another.

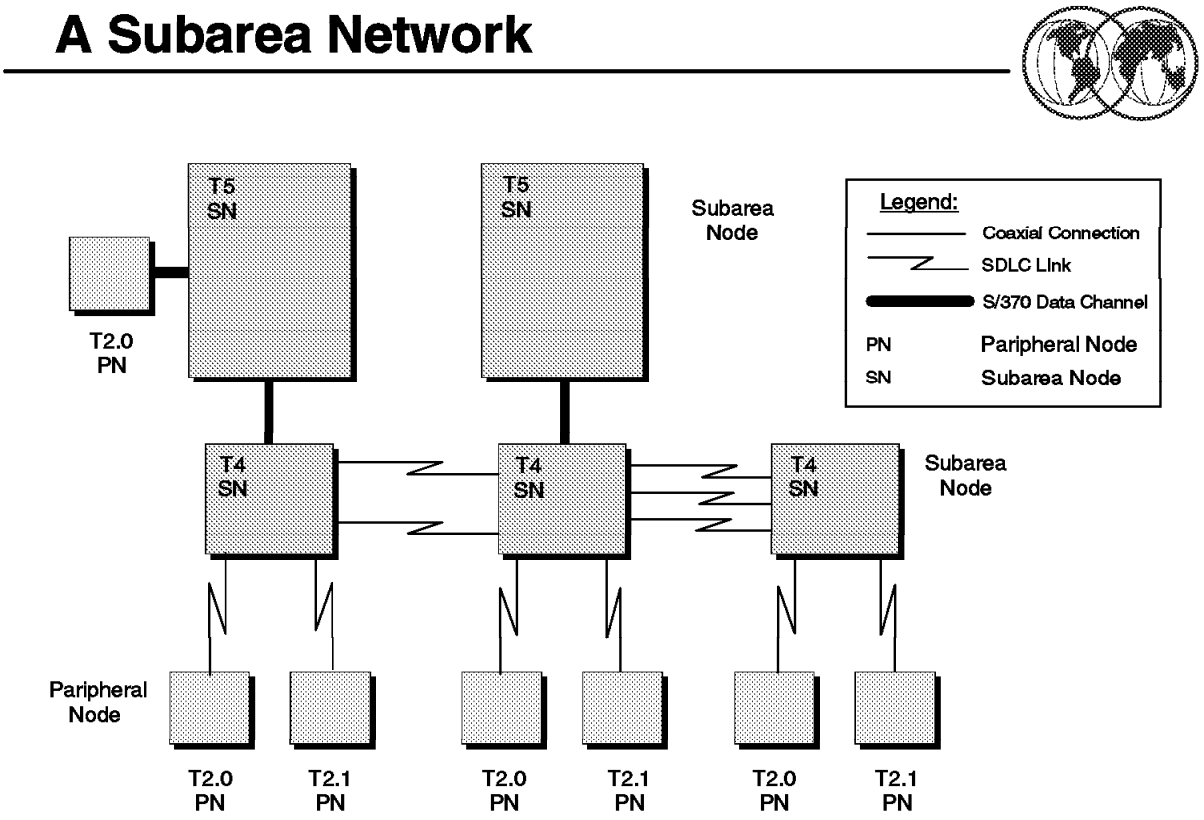


Figure 10. A subarea network

1.10 A subarea network

A data communication network can be described as a configuration of nodes and links. Nodes are the network components that send data over, and receive data from, the network. Node implementations include processors, controllers, and workstations. Links are the network components that connect adjacent nodes. Nodes and links work together in transferring data through a network.

A node is a set of hardware and associated software components that implement the functions of the seven architectural layers. Although all seven layers are implemented within a given node, nodes can differ based on their architectural components and the sets of functional capabilities they implement. Nodes with different architectural components represent different node types. Four types of nodes exist: type 5 (T5), type 4 (T4), type 2.0 (T2.0), and type 2.1 (T2.1).

Nodes that perform different network functions are said to act in different network roles. It is possible for a given node type to act in multiple network roles. A T4 node, for example, can perform an interconnection role between nodes at different levels of the subarea network hierarchy, or between nodes in different subarea networks. The functions performed in these two roles are referred to as *boundary function* and *gateway function*, respectively. T2.1 and T5 nodes can also act in several different network roles. Node roles fall into two broad categories: hierarchical roles and peer-oriented roles.

Hierarchical roles are those in which certain nodes have a controlling or mediating function with respect to the actions of other nodes. Hierarchical networks are characterized by nodes of all four types acting in hierarchical roles. Within such networks, nodes are categorized as either subarea

nodes (SNs) or peripheral nodes (PNs). Subarea nodes provide services for and control over peripheral nodes. Networks consisting of subarea and peripheral nodes are referred to as *subarea networks*.

- Subarea nodes

  Type 5 (T5) and type 4 (T4) nodes can act as subarea nodes. T5 subarea nodes provide the SNA functions that control network resources, support transaction programs, support network operators, and provide end-user services. Because these functions are provided by host processors, T5 nodes are also referred to as *host nodes*.

  T4 subarea nodes provide the SNA functions that route and control the flow of data in a subarea network. Because these functions are provided by communication controllers, T4 nodes are also referred to as *communication controller nodes*.

- Peripheral nodes

  Type 2.0 (T2.0) and type 2.1 (T2.1) nodes can act as peripheral nodes attached to either T4 or T5 subarea nodes. Peripheral nodes are typically devices such as distributed processors, cluster controllers, or workstations. A T2.1 node differs from a T2.0 node by the T2.1 node′s ability to support peer-oriented protocols as well as the hierarchical protocols of a simple T2.0 node. A T2.0 node requires the mediation of a T5 node in order to communicate with any other node. Subarea nodes to which peripheral nodes are attached perform a boundary function and act as subarea boundary nodes.

An APPN Network

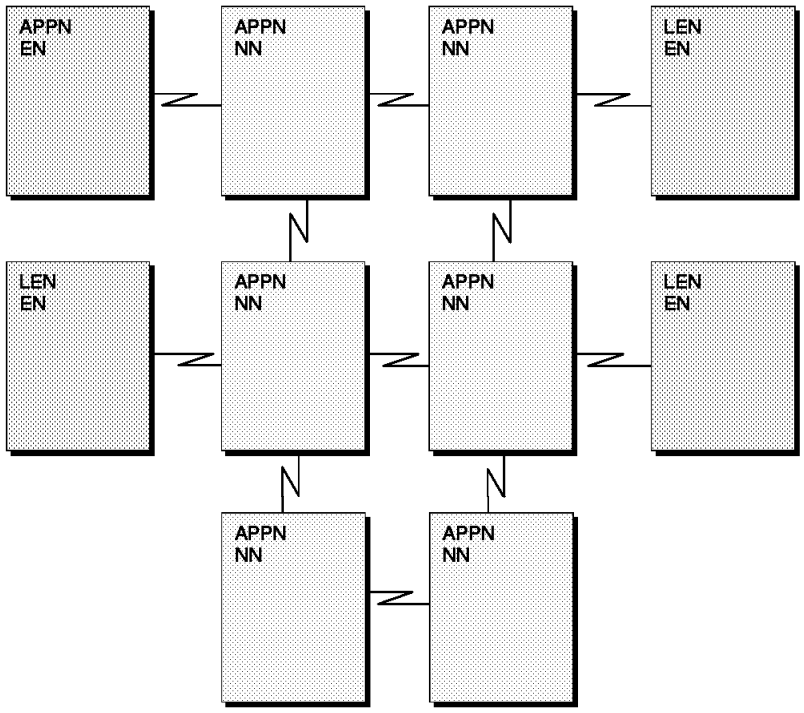


Figure 11. An APPN network

1.11 An APPN network

APPN extensions allow greater distribution of network control by enhancing the dynamic capabilities of the node. Nodes with these extensions are referred to as APPN nodes, and a network of APPN nodes makes up an APPN network. A low-entry networking (LEN) node can also attach to an APPN network. An APPN node can dynamically find the location of a partner node, place the location information in directories, compute potential routes to the partner, and select the best route from among those computed. These dynamic capabilities relieve network personnel from having to predefine those locations, directory entries, and routes. APPN nodes can include processors of varying sizes such as the Application System/400 (AS/400), the Enterprise System/9370(*) (ES/9370(*)) running under Distributed Processing Program Executive/370 (DPPX/370), the Personal System/2(*) (PS/2(*)) running under Operating System/2(*) (OS/2(*)), and VTAM running under Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA).

Peer-oriented protocols enable nodes to communicate without requiring mediation by a T5 node, giving them increased connection flexibility. APPN defines two possible roles for a node in an APPN network, that of an end node and that of a network node. (Network nodes provide additional options that can further distinguish them.)

T2.1 nodes can act as either APPN or LEN nodes. T5 nodes can also act as APPN or LEN nodes, but have additional capability to interconnect subarea and APPN networks by interchanging protocols between them. (In this capacity, they are called *interchange nodes*). Together with its subordinate T4 nodes, a T5 node can also form a composite LEN node or a composite network node. As composite

nodes, they appear as single LEN or network nodes to other LEN or APPN nodes to which they are interconnected.

If two network nodes do not support the border node option and are located in two separate net-ID subnetworks, CP-CP sessions cannot be established between them. For the two network nodes to communicate, a LEN connection may be established between the two. This allows the two net-ID subnetworks to communicate, but does not support any APPN function. If NN1 in subnet A is to establish a session with NN2 in subnet B, all LUs in subnet A must be predefined to NN2 in subnet B. The network nodes in both subnets are APPN nodes, but since they communicate across net-ID subnetwork boundaries, they are defined to each other via LEN links.

- End nodes

  These are located on the periphery of an APPN network. An end node obtains full access to the APPN network through one of the network nodes to which it is directly attached--its network node server. The two kinds of end nodes are APPN end nodes and LEN end nodes. An APPN end node supports APPN protocols through explicit interactions with a network node server. Such protocols support dynamic searching for resources and provide resource information for the calculation of routes by network nodes. A LEN end node is a LEN node attached to a network node. Although LEN nodes lack the APPN extensions, they are able to be supported in APPN networks using the services provided them by network nodes. In an APPN network, when a LEN node is connected to another LEN node, or to an APPN end node, it is referred to simply as a LEN node. When connected to an APPN network node, however, it is referred to as a LEN end node.

- Network nodes

  Together with the links interconnecting them, network nodes form the intermediate routing network of an APPN network. Network nodes connect end nodes to the network and provide resource location and route selection services for them. Routes used to interconnect network users are selected based on network topology information that can change dynamically. The graphic represents one possible APPN network configuration and contains LEN end nodes as well as APPN nodes.

- Network node server

  This is a network node that provides resource location and route selection services to the LUs it serves. These LUs can be in the network node, itself, or in the client end nodes. A network node server uses CP-CP sessions to provide network information for session setup in order to support the LUs on served APPN end nodes. In addition, LEN end nodes can also take advantage of the services of the network node server. A LEN end node, unlike an APPN end node, must be predefined by the network operator as a client end node for which the network node acts as server. Any network node can be a network node server for end nodes that are attached to it. The served end nodes are defined as being in that network node server's domain.

- Central directory server

  This is a network node that builds and maintains a directory of resources from the network. The purpose of a CDS is to reduce the number of network broadcast searches to a maximum of one per resource. Network nodes and APPN end nodes can register their resources with a CDS, which acts as a focal point for resource location information.

  A CDS can be involved in APPN end node resource registration. An APPN end node registers its its resources to improve network search performance. When an APPN end node registers its resources with its network node server, it can request that its network node server also register them with a CDS. Entries in a directory database can be registered, defined, or dynamic.

  When a network node receives a search request for a resource that has no location information, the network node first sends a directed search request to a CDS if there is one. The CDS searches in its directory for information about the location of the resource. If it does not find the location of the resource, the CDS searches end nodes in its domain, other CDSs, and, if necessary, the entire network (via a "broadcast search"). If the resource is still not found, the CDS notifies the network

node that originally requested the search that the search is unsuccessful. A central directory client is a network node that forwards directory searches to a CDS.

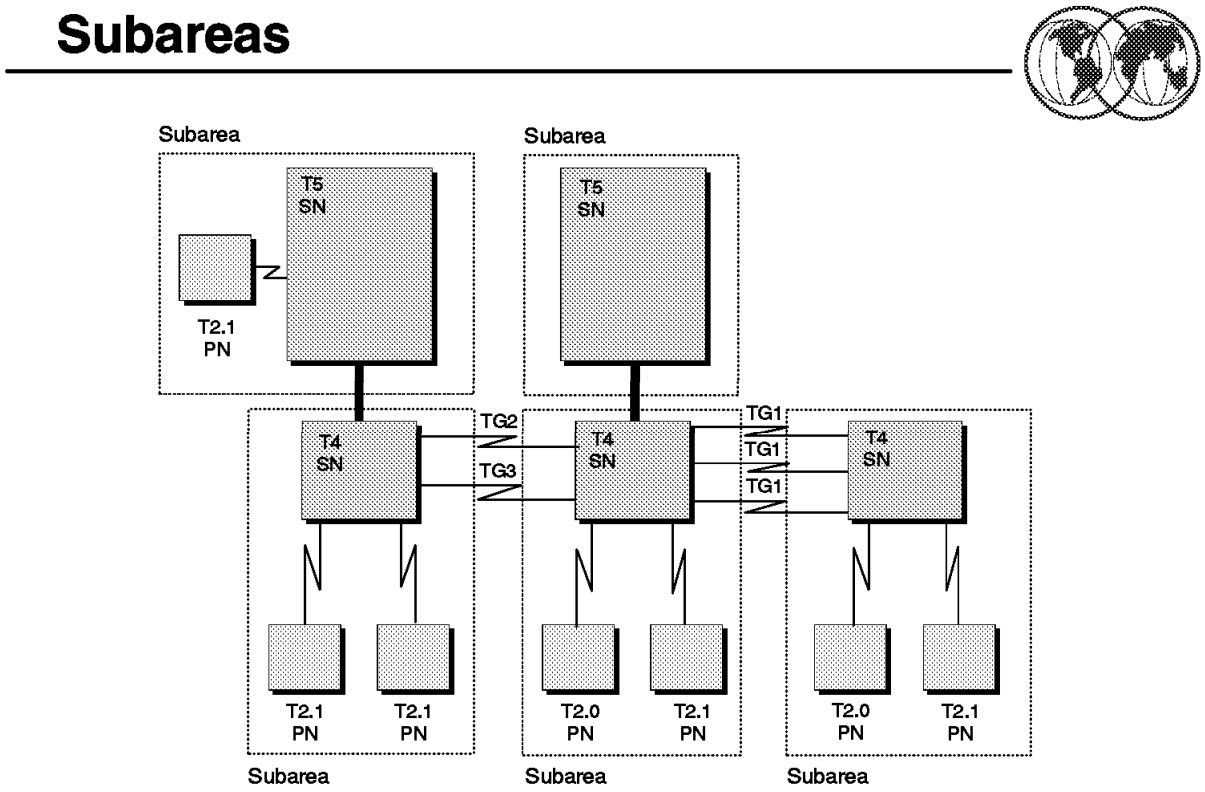


Figure 12. Subareas

1.11.1 Subareas

SNA defines the following network configurations:

- A hierarchical network consisting of subarea nodes and peripheral nodes
- A peer-oriented network consisting of APPN and LEN nodes
- A mixed network that combines one or more hierarchical subnets with one or more peer-oriented subnets.

The organization of a hierarchical network structure is determined by the way control of network services is maintained. Host nodes containing SSCPs are responsible for overall control of communication in the hierarchical network.

A hierarchical network might include LEN or APPN nodes that are attached as peripheral nodes. These nodes can communicate with each other through a subarea network if the boundary nodes to which they are attached support the basic SSCP-independent LU-LU protocols needed for such peer interactions.

A subarea consists of one subarea node and the peripheral nodes that are attached to that subarea node. The concept of a subarea applies only to subarea networks and composite networks. The network configuration in the visual contains five subarea nodes and seven peripheral nodes. Because each subarea node and its attached peripheral nodes constitute a subarea,

Domains in a Subarea Network

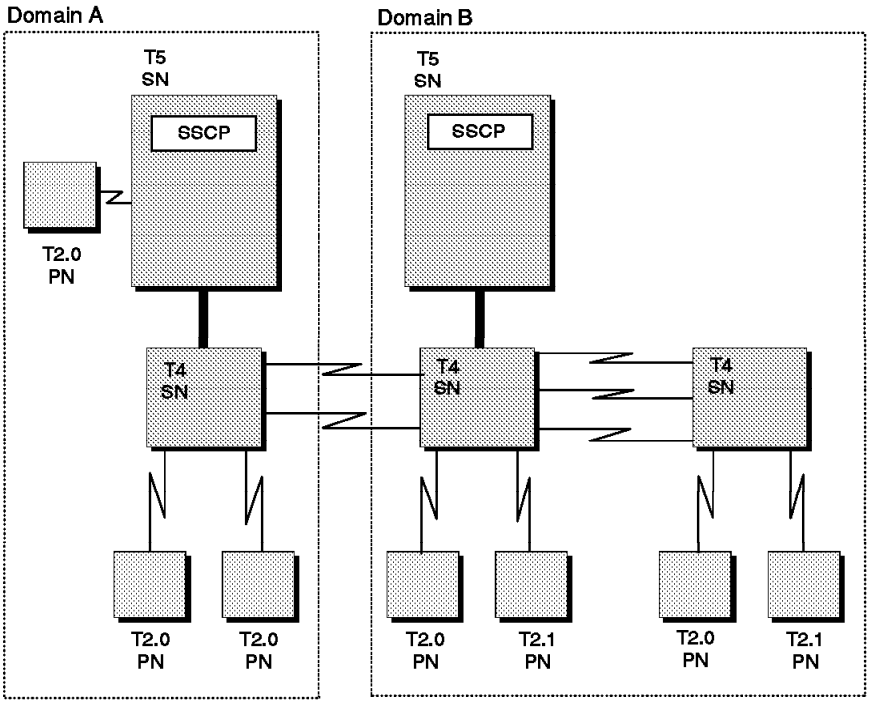


Figure 13. Domains in a subarea network

1.11.2 Domains in a subarea network

A *domain* is an area of control. The concept of a domain within a subarea network differs from that within an APPN network. Within a subarea network, a domain is that portion of the network managed by the control point in a T5 subarea node. The control point in a T5 subarea node is called a system services control point (SSCP).

When a subarea network has only one T5 node, that node must manage all of the network resources. A subarea network that contains only one T5 node is a single-domain subarea network. When there are multiple T5 nodes in the network, each T5 node may control a portion of the network resources. A subarea network that contains more than one T5 node is a multiple-domain subarea network. In a multiple-domain subarea network, the control of some resources can be shared between SSCPs. Some resources can be shared serially and some concurrently. For more information on resource sharing in subarea networks, see “Defining Shared Control of Resources in Subarea Networks” in Chapter 4, “Defining Network Resources.” The visual illustrates two domains, A and B, joined by direct-attached T4 nodes to form a multiple-domain subarea network.

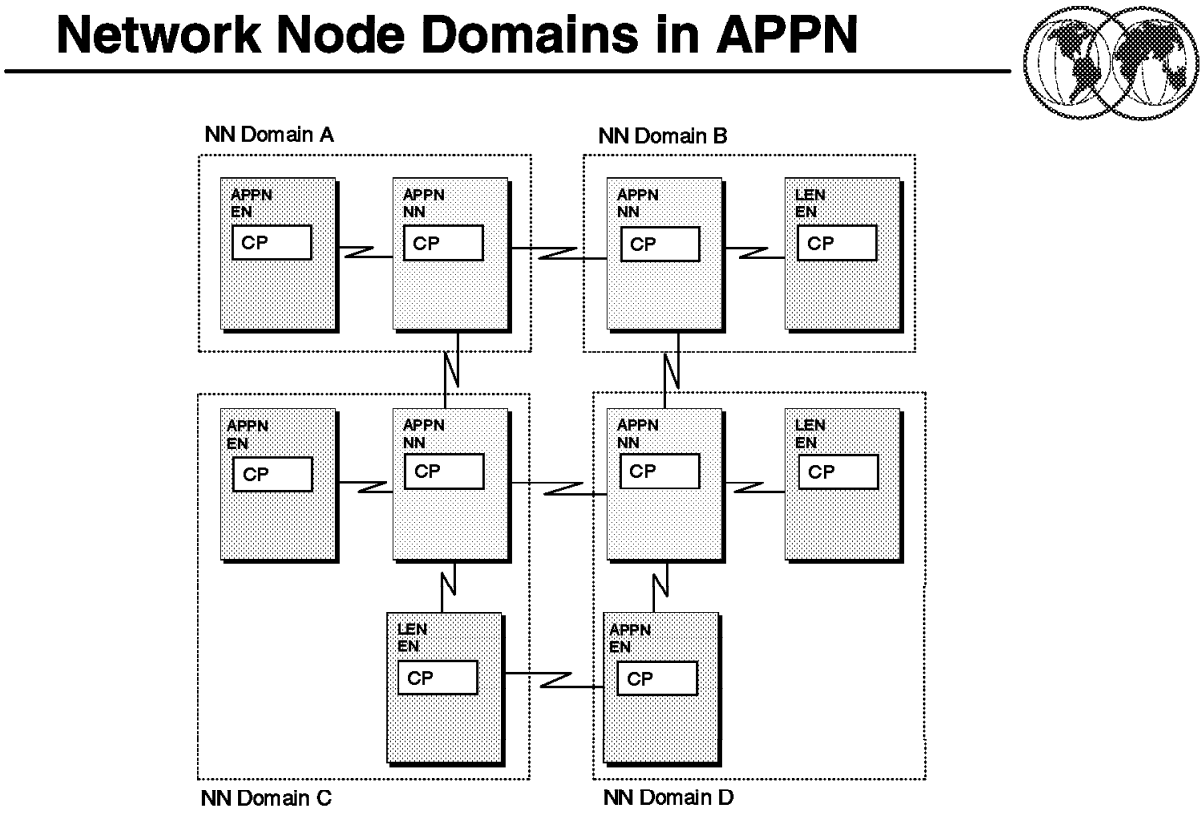


Figure 14. Network node domains in APPN

1.11.3 Network node domains in APPN

An APPN network constitutes a peer-oriented SNA network. All APPN nodes are considered to be peers and do not rely on other nodes to control communication in the network the way a subarea node controls communication between peripheral nodes. There is, however, a measure of hierarchical control because a network node server provides certain network services to its attached end nodes. The difference is that, in APPN networks, hierarchical control is not determined by product or processor type as it is in subarea networks, where only large host processors contain SSCPs and node types generally reflect product types.

The domain of a node in an APPN network is that portion of the network served by the control point in the node. The control point in an APPN node is called simply a control point (CP). An end node control point's domain consists solely of its local resources. It is included within the domain of its network node server. A network node control point's domain includes the resources in the network node and in any client end nodes (nodes for which the network node is acting as the network node server) attached directly to the network node.

APPN networks are by definition multiple-domain networks. The graphic illustrates an APPN network containing four network node domains, A, B, C, and D. The domains of the end nodes are included within the domains of their respective network node servers.

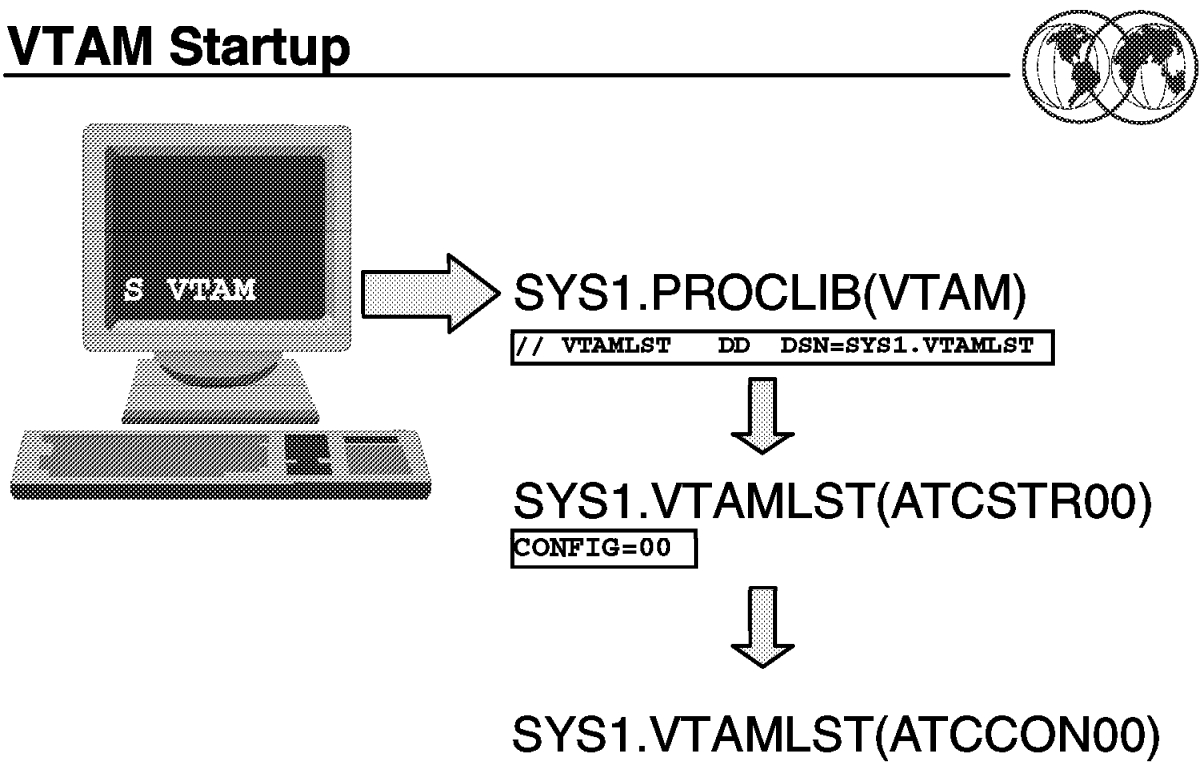


Figure 15. VTAM startup

1.12 Starting VTAM

VTAM can be started manually by entering the command:

```
S VTAM,,,(LIST=xx)
```

This will cause the VTAM start procedure to be executed.

In the following example of a basic VTAM proc, note the **VTAMLST** DD statement which specifies the location of the network definition members and the **VTAMLIB** DD statement which specifies the location of VTAM's executable code.

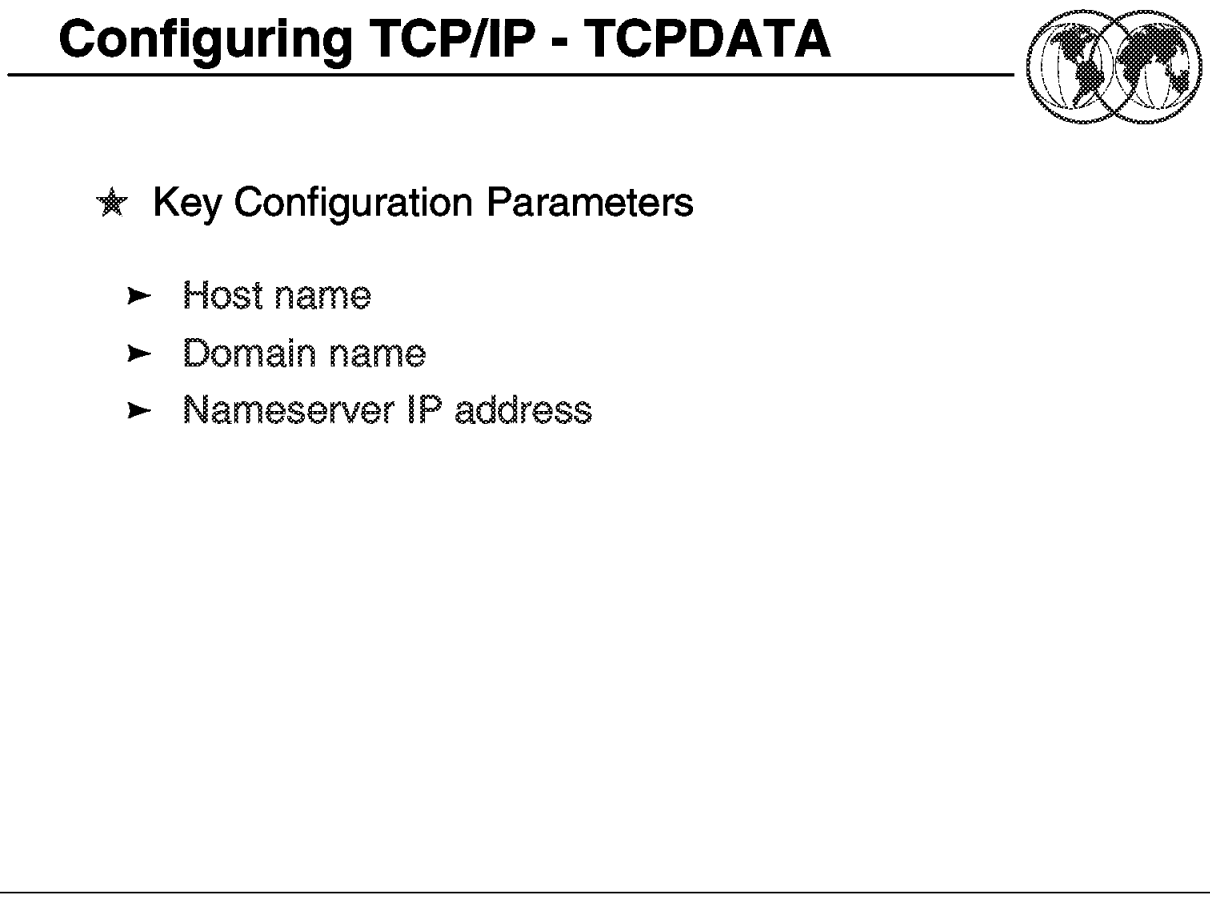


Figure 16. VTAM procedure

1.12.1 VTAM procedure

The first thing VTAM looks for is an ATCSTRxx member in its VTAMLST data set, where xx is the LIST=xx value and can be any 2-character alphanumeric value.

The ATCSTRxx member contains VTAM start parameters that define VTAM's identity on the network as well as tuning options and a pointer to the major node startup list, ATCCONxx.

If no LIST= parameter was specified on the `S VTAM` command, the default value of **00** is used.

For example:

The command `S VTAM,,,(LIST=Z9)` will cause VTAM to read startup parameters from ATCSTRZ9

The command `S VTAM,,,(LIST=EX)` will cause VTAM to read startup parameters from ATCSTREX

The command `S VTAM,,,(LIST=99)` will cause VTAM to read startup parameters from ATCSTR99

The command `S VTAM` will cause VTAM to read startup parameters from the default ATCSTR00

A typical ATCSTRxx member might contain:

```
SSCPID=0061,
CONFIG=00,
HOSTSA=130,
NETID=GBIBMPGX,
SSCPNAME=PGXYA,
IOBUF=(100,256,19,,12,30)
```

For a complete description of the parameters and their values refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*, SC31-8565.

Following is a brief description:

SSCPID A decimal integer in the range of 0-65535 that uniquely identifies this VTAM to any other connected VTAMs that will communicate with it. It is used in the setup of cross-domain sessions.

SSCPNAME This is a 1 to 8-character name that uniquely identifies this VTAM to any other connected VTAMs that will communicate with it. It is also used in the setup of cross-domain sessions.

HOSTSA This defines this VTAM′s subarea number.

NETID This specifies the name of the network containing this VTAM.

CONFIG This is a 2-character suffix of the ATCCONxx member that contains the list of major nodes to be automatically activated at startup.

IOBUF This is one of the many VTAM tuning parameters. The values inside the brackets determine how VTAM allocates and manages storage associated with a particular buffer pool.

1.12.1.1 ATCCONxx

The ATCCONxx member, where xx is the value specified on the CONFIG=.. parameter on ATCSTRxx, contains a list of the members or *major nodes* that VTAM will automatically activate on startup.

Typically, these will include at least:

- The TSO application major node
- One or more CICS application major nodes
- A local 3270 terminal′s major node

Examples of other major nodes, depending on the hardware and software installed, can include:

- An NCP major node
- An XCA major node
- An adjacent SSCP table major node
- A PATH table major node
- A Cross-Domain Resource Manager major node
- A Cross-Domain Resources major node
- A channel-to-channel Adapter major node
- Additional application major nodes, for example, NetView

The above list is not intended to be exhaustive, merely to give examples of the major nodes that might be included in a typical production startup list.

Major nodes are discussed further in 1.12.4, “VTAM major nodes” on page 38.

For reference, see *OS/390 eNetwork Communications Server SNA Resource Definition Reference*, SC31-8565 and *OS/390 eNetwork Communications Server SNA Resource Definition Samples*, SC31-8566.

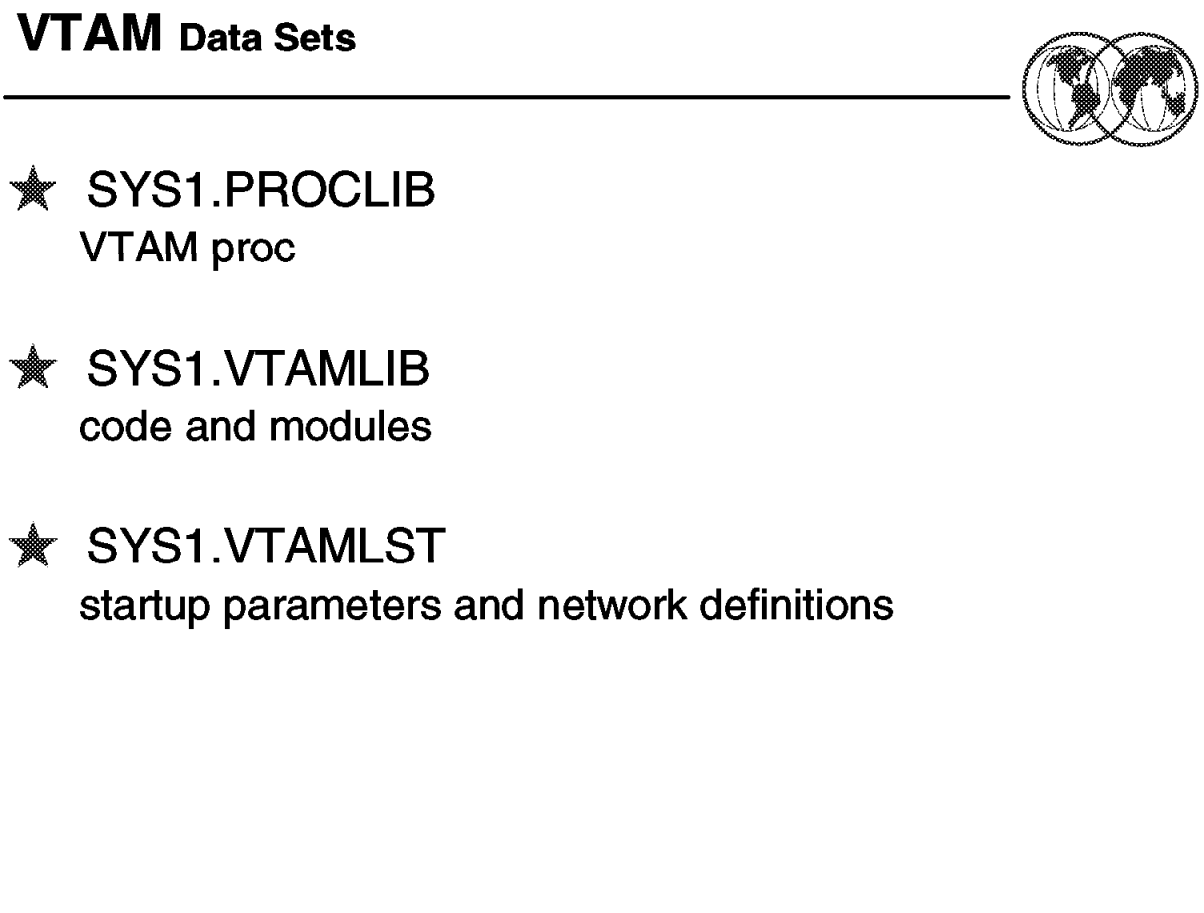


Figure 17. VTAM data sets

1.12.2 VTAM data sets

VTAM configuration depends on the customization of various control or parameter files. Advanced VTAM customization can involve writing complex user exits and changes to such items as the Unformatted System Services (USS) table, the LOGMODE table, and the Class Of Service (COS) table, to name just a few. These members are located in the VTAMLIB data set, usually SYS1.VTAMLIB.

To start with, however, basic VTAM configuration can be achieved by tailoring the following:

- The VTAM start procedure SYS1.PROCLIB(VTAM).
- The VTAM startup member SYS1.VTAMLST(ATCSTRxx).
- The VTAM major node activation list SYS1.VTAMLST(ATCCONxx).
- Major node members, usually located in SYS1.VTAMLST. These members are used to define the SNA network, and are discussed in 1.12.4, “VTAM major nodes” on page 38.

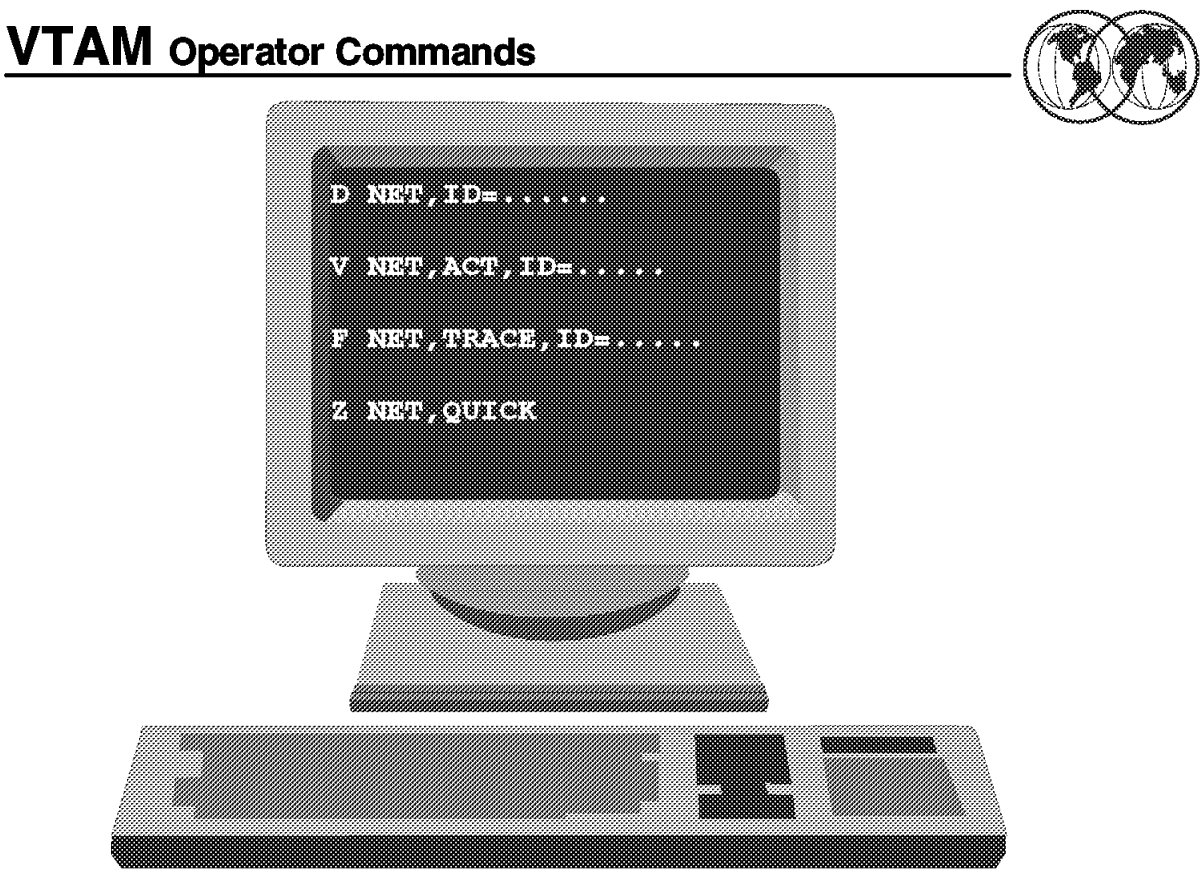


Figure 18. VTAM commands

1.12.3 VTAM commands

VTAM operator commands fall into four categories:

1. The Display commands
   D NET,...
2. The VARY commands
   V NET,...
3. The Modify commands
   F NET,...
4. The Halt commands
   Z NET,...

The following command descriptions and examples give an overview of the VTAM Operator commands available. For a complete list of VTAM Operator commands and their syntax and use refer to *OS/390 eNetwork Communications Server: SNA Operation*, SC31-8567.

1.12.3.1 Display commands

The basic VTAM Display commands can be used to interrogate VTAM for information about the status of communications links, devices, applications, and even the state of VTAM itself.

The experienced VTAM operator or systems programmer can often successfully diagnose many VTAM related problems simply by analyzing the output of the various Display commands.

Some of the more useful include:

Table 1. VTAM Display commands

Command	Description
D NET,VTAMOPTS	Display VTAM options, including SUBAREA Number, NETID, SSCPNAME, SSCPID
D NET,BFRUSE	Display information about VTAM′s virtual storage and buffer pool usage
D NET,APPLS	Display the applications (APPLIDS) that VTAM knows about
D NET,PATHTAB	Display the status of communications PATHs between subareas in the VTAM network
D NET,MAJNODES	Display details about major nodes
D NET,ID=.....	Display information about the specified VTAM resource
D NET,ID=.....,E	Display more information about the specified VTAM resource

1.12.3.2 Vary commands

These commands can be used to deactivate a resource, thereby making it unavailable on the network, or to reactivate a failed resource, for example.

Table 2. VTAM Vary commands

Command	Description
V NET,ACT,ID=.....	Activate the specified resource
V NET,INACT,ID=.....	Deactivate the specified resource
V NET,INACT,ID=.....,I	Immediately deactivate the specified resource
V NET,INACT,ID=.....,F	Force-deactivate the specified resource
V NET,ACQ,ID=.....	Acquire the specified resource (NCP or non-switched PU) without activating it - used during network recovery whereby this VTAM can take ownership of another VTAM′s resource

1.12.3.3 Modify commands

The VTAM Modify commands are used for the more advanced VTAM functions such as replacing an active LOGMODE table, for example, without having to restart VTAM, or to start and stop VTAM tracing.

Table 3. VTAM Modify commands

Command	Description
F NET,TNSTAT,CNSL=YES	Activate the VTAM tuning statistics display and output to the console
F NET,NOTNSTAT	Deactivate the VTAM tuning statistics display
F NET,TRACE,.....	Initiate or Modify VTAM tracing
F NET,NOTRACE,.....	Terminate VTAM tracing

1.12.3.4 Halt commands

The VTAM Halt commands are used to stop VTAM processing.

Table 4. VTAM Halt commands

Command	Description
Z NET	Terminate VTAM processing normally
Z NET,QUICK	Terminate VTAM processing immediately without waiting for normal termination of existing sessions - should only be used when normal termination does not complete successfully
Z NET,CANCEL	Abends VTAM - should only be used when both Z NET and Z NET,QUICK are unsuccessful

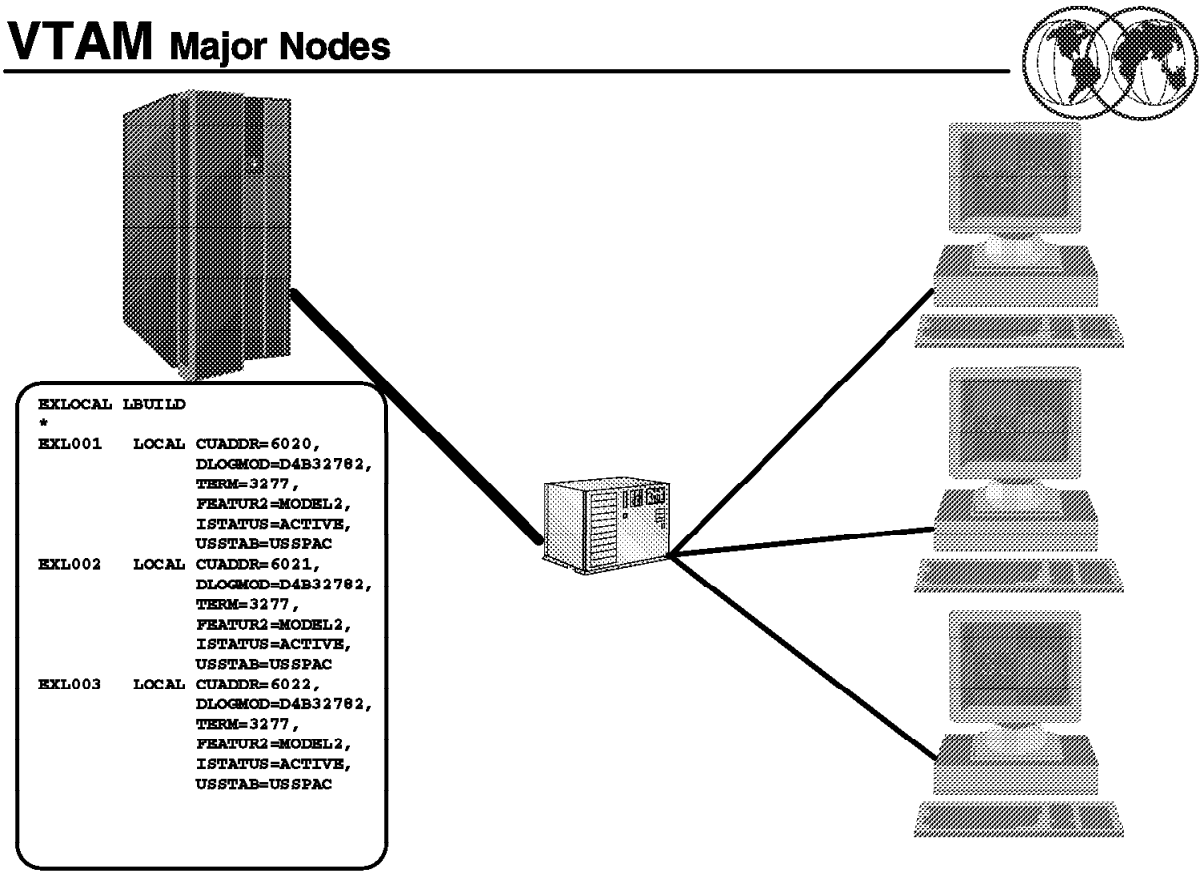


Figure 19. VTAM major nodes

1.12.4 VTAM major nodes

To define the attachment of any device, or set of devices, to VTAM you need to code a major node member in the VTAMLST data set.

A *major node* is a list of parameters that define a specific type of network device or group of related devices that VTAM can address. The different types of SNA major node are:

- Adjacent control point (ADJCP)
- Application program (APPL)
- Channel attachment (CA)
- Cross-domain resource (CDRSC)
- Cross-domain resource manager (CDRM)
- External communication adapter (XCA)
- Local non-SNA (LBUILD)
- Local SNA (LOCAL)
- LU group (LUGROUP)
- Model (MODEL)
- Network Control Program (NCP)
- Switched (SWNET)

- Transport resource list (TRL)

For a complete description of each of these, including syntax and coding rules, refer to *eNetwork Communications Server: SNA Resource Definition Reference*, SC31-8565 and *eNetwork Communications Server: SNA Resource Definition Samples*, SC31-8566.

1.12.4.1 Local non-SNA major node

See A.1, “Major node definitions” on page 339 for an example of a local non-SNA major node to define coax-attached 3270 terminals connected to a channel-attached 3174-type controller.

An XCA major node is used to define an External Communication Adapter to VTAM. For an example of an OSA device attached to a token ring network definition, see A.2, “XCA Major Node” on page 342.

For an example of a switched major node to define switched communication links to VTAM, see A.3, “Switched major node” on page 343.

1.13 TCP/IP

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

The TCP/IP component of eNetwork Communications Server supports the following functions:

- Log on to a remote host
- Transfer data sets
- Send and receive electronic mail
- Print on remote printers
- Authenticate network users
- Display IBM GDDM/MVS graphics on X Window System workstations
- Run a command on another host
- Monitor the network
- Query name servers
- Manage network resources

In the following topics we will present an overview and a brief discussion of some of the more significant concepts or setup steps to help you get going.

For more information, see *eNetwork Communications Server: IP User's Guide*, GC31-8514, and *OS/390 TCP/IP OpenEdition: Configuration Guide*, SC31-8304.

The Layered Structure of TCP/IP

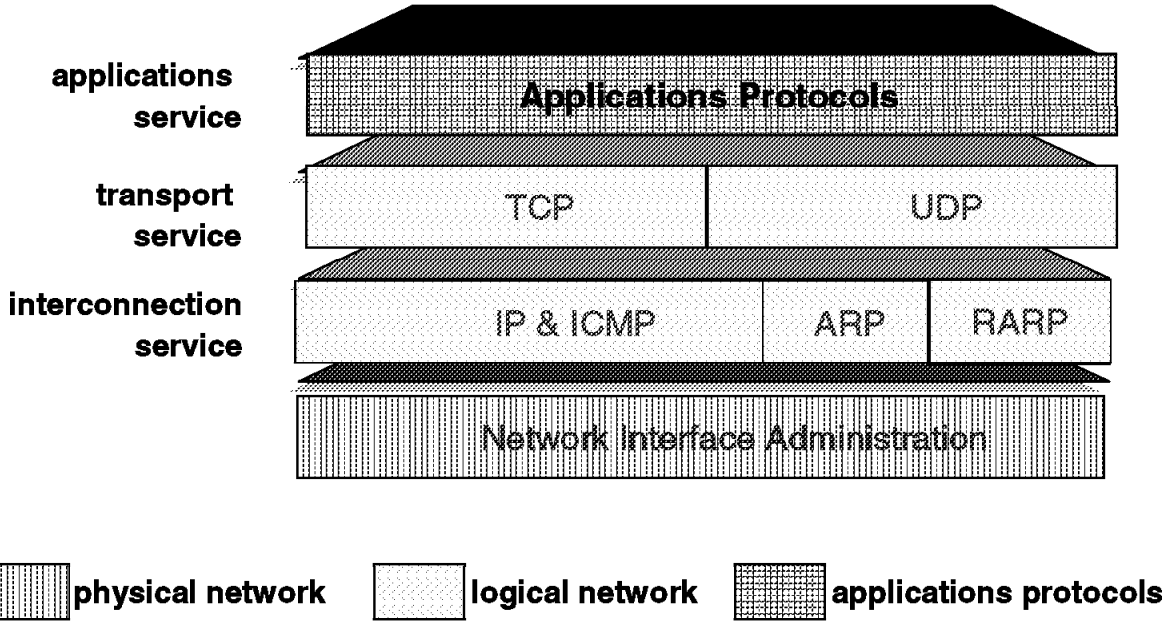


Figure 20. TCP/IP layers

1.14 TCP/IP layered structure

The four layers that make up the TCP/IP protocol suite are:

1. *Network Interface* - The lowest layer is the physical layer and is not actually specified by TCP/IP. The user can access any physical transmission approach including both wide area (WAN) and local area networks (LAN). The common protocols at this level have been X.25 and Ethernet. This independence from the type of underlying physical network is one of the benefits of TCP/IP.
2. *Internetwork layer* - This layer provides for routing of datagrams (messages) from one device to another. This could be from one local device to another, or it could be across a wide area network with many intervening networks that must be crossed. Every TCP/IP host has a unique *internet address* which is a combination of a network ID that is assigned centrally, and a local host address that is locally administrated. This permits the routing of datagrams (messages between, as well as within, enterprises since all hosts have a unique address.
3. *Transport layer* - The TCP implementation of the transport layer provides for reliable transmission of data. At the lower layers, all transmissions are in individual packets. The TCP level involves both error checking, re-transmission and assembly/disassembly of packets into/from logical messages. The TCP level is optional, and an alternative protocol, User Datagram Protocol (UDP) may be used. UDP provides no error checking and no assembly/disassembly of packets.
4. *Applications* - The application layer consists of many different application types that may or may not be implemented in a particular TCP/IP installation. Certain applications are implemented in nearly all TCP/IP installations. These include TELNET (terminal emulation), SMTP (simple mail transfer protocol - E-mail), and FTP (file transfer protocol).

With TCP/IP it is possible to have connectivity between different vendors′ systems, but not have all the applications that a user may require. Therefore, the user must compare each vendor′s implementation and set of functions supported in order to insure that all the requirements are met.

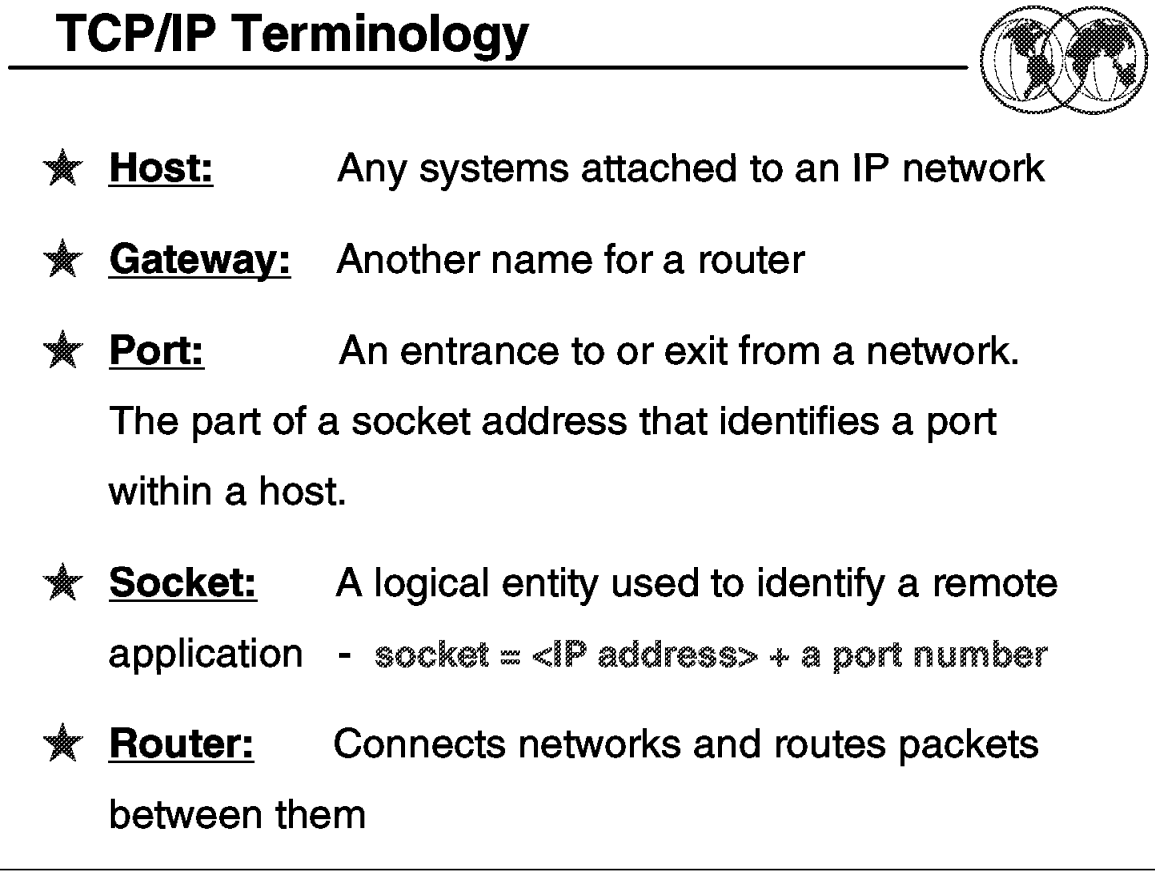


Figure 21. TCP/IP terminology

1.14.1 TCP/IP terminology

The following are commonly used terms:

Host In the Internet suite of protocols, this is an end system. The end system can be any workstation; it does not have to be a mainframe.

Gateway A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. In TCP/IP, this is a synonym for router.

Port Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number, used by the host-to-host protocol to identify to which higher level protocol or application program (process) it must deliver incoming messages. There are two types of port:

Well-known Well-known ports belong to standard servers, for example TELNET uses port 23. Well-known port numbers range between 1 to 1023 (prior to 1992, the range between 256 to 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port. The well-known ports are controlled and assigned by the Internet central authority (IANA) and on most

systems can only be used by system processes or by programs executed by privileged users. The reason for well-known ports is to allow clients to be able to find servers without configuration information.

Ephemeral Clients do not need well-known port numbers because they initiate communication with servers and the port number they are using is contained in the UDP datagrams sent to the server. Each client process is allocated a port number for as long as it needs by the host it is running on. Ephemeral port numbers have values greater than 1023, normally in the range 1024 to 65535.

Socket An endpoint for communication between processes or application programs. A synonym for port.

Socket address The address of an application program that uses the socket interface on the network. In Internet format, it consists of the IP address of the socket′s host and the port number of the socket. The application program is usually not aware of the structure of the address.

Socket interface A Berkeley Software Distribution (BSD) application programming interface (API) that allows users to easily write their own programs.

Router A router interconnects networks at the internetwork layer level and routes packets between them. The router must understand the addressing structure associated with the networking protocols it supports and make decisions on whether, or how, to forward packets. Routers are able to select the best transmission paths and optimal packet sizes. The basic routing function is implemented in the IP layer of the TCP/IP protocol stack, so any host or workstation running TCP/IP over more than one interface could, in theory and also with most of today′s TCP/IP implementations, forward IP datagrams. However, dedicated routers provide much more sophisticated routing than the minimum functions implemented by IP.

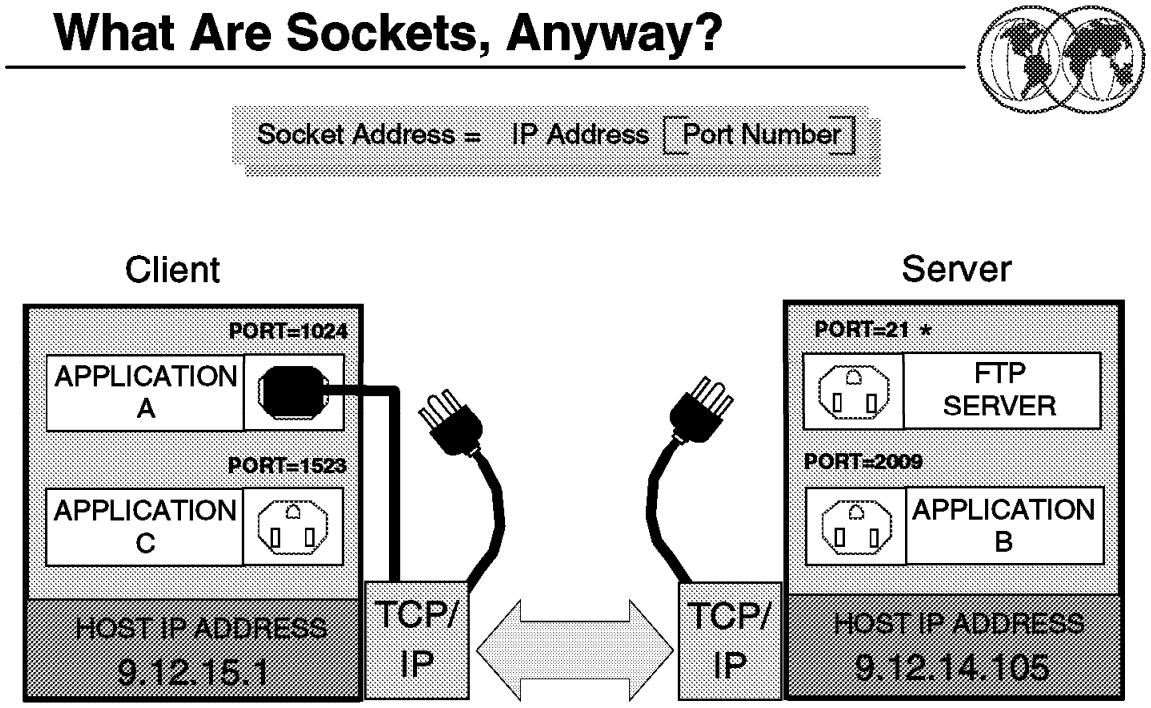


Figure 22. What are sockets, anyway?

1.14.2 Sockets

As originally defined by the BSD UNIX, sockets and the associated set of interface calls were developed to enable UNIX processes to communicate across a TCP/IP network in a simple, file-based manner. As POSIX programs in OS/390 UNIX are essentially UNIX processes, sockets provide an important means of process-to-process communication irrespective of process location.

A good SNA analogy to sockets is the Logical Unit (LU). An LU has a “logical” address as represented by the unique LU name and it is used to represent a communications endpoint for an application. The SNA support also provides a set of LU interface calls that an application uses to drive the communication process.

In the visual there are three examples of communication between process A on the left and other processes. In each case, a socket has to be allocated on each side of a communication link with a corresponding socket ID or address. In fact, three elements are needed to totally define a socket:

- The interface IP address
- The port number
- The interface protocol to be used on the socket-to-socket connection. Although the protocol does not affect the socket address, it must be specified to fully define each socket. I have left it out for simplicity.

The term *socket* has two associated meanings. In TCP/IP network terms, a socket represents a *communication end point* in the network, used by an application to communicate with a remote partner.

The term “sockets” is also associated with a particular set of networking *program calls (API)* supported in POSIX/UNIX.

A socket is identified by an Internet *network address* comprising:

- A unique system or IP address for the host system network interface, which is a 32-bit “dotted decimal” number.
- A local *port* address, which is a two-byte binary number expressed in decimal form. A port represents a duplex logical connection to the TCP/IP network, and provides the anchor for queues for sending and receiving session data. Port numbers are divided into *well-known* ports (1–1024), which are reserved for special TCP protocols, and *dynamically allocated* ports (greater than 1024) allocated to ordinary applications on an as-requested basis.

A set of C-based program calls enable applications to use file-based semantics on sockets in order to communicate with applications on a remote socket. Such a program is said to be using the “socket” communications API. Although originally defined to support TCP/IP network communication across the Internet, this interface is increasingly becoming a generic *inter-process communication (IPC)* interface.

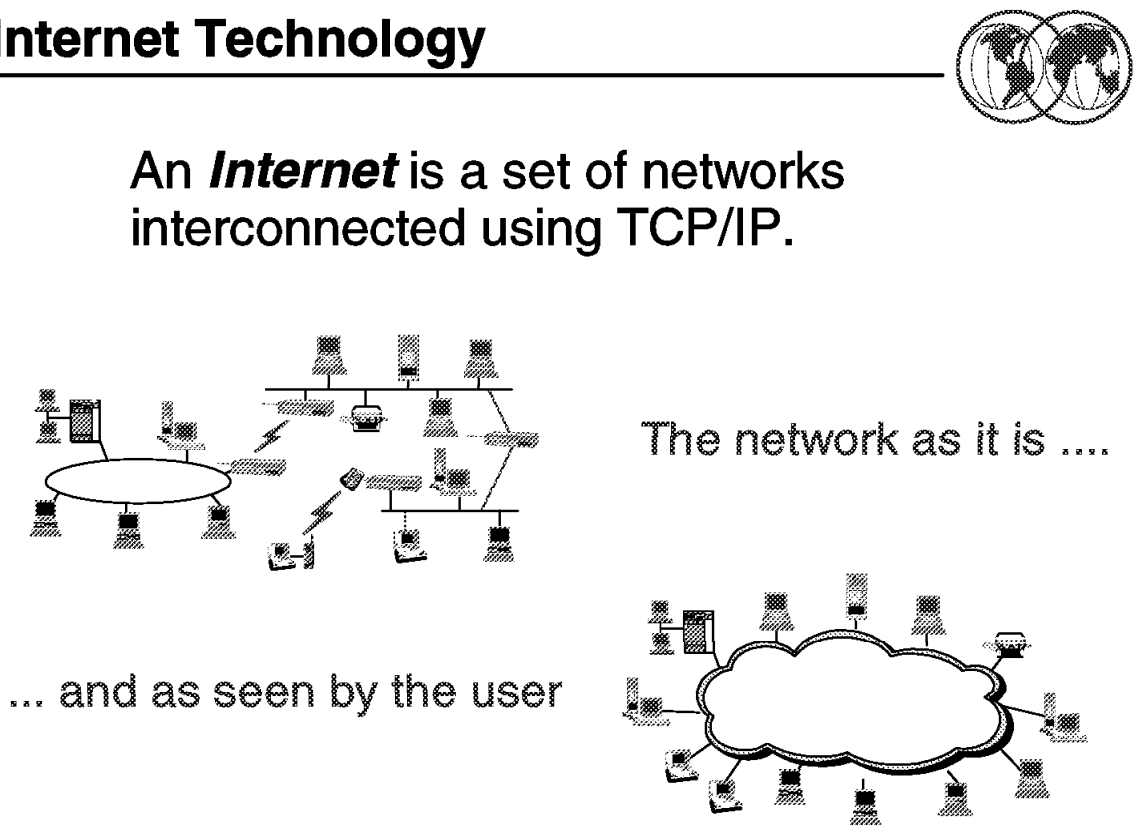


Figure 23. Internet technology

1.15 Internet technology

An *internet* is a set of interconnected data networks and associated protocols which form a coordinated unit, independent of the underlying physical networks, permitting communications between two or or more *processes* (applications) in a manner independent of the underlying physical networks being used.

The goal of an *internet* is to build a unified, cooperative interconnection of networks that supports a common communication service. Within each physical network, computers will use the underlying technology-dependent communication software and hardware (X.25, SNA, Ethernet, etc.). New software (TCP, IP, etc.), inserted between the technology-dependent communication mechanisms and application programs, hide the low-level details and make the collection of separate physical networks appear to be a *single large network*. Such an interconnection scheme is called an *internetwork* or an *internet*.

The first design goal of the TCP/IP protocol suite was to build an interconnection of networks that provide universal communication services, an *internet*.

- Internet technology makes it possible to interconnect different existing physical networks and make them appear to the user as a single coordinated unit.
- This internet technology hides the details of the network hardware and allows computers to communicate independently of the underlying physical network.

- Internet technology supports peer-to-peer communications between computers attached anywhere on the internet.

In this visual, we see the actual internet on the left consisting of different networks, with different physical media, different protocols, and different performances. Using Internet technology, this would appear to a user as a single composite network, as shown on the right.

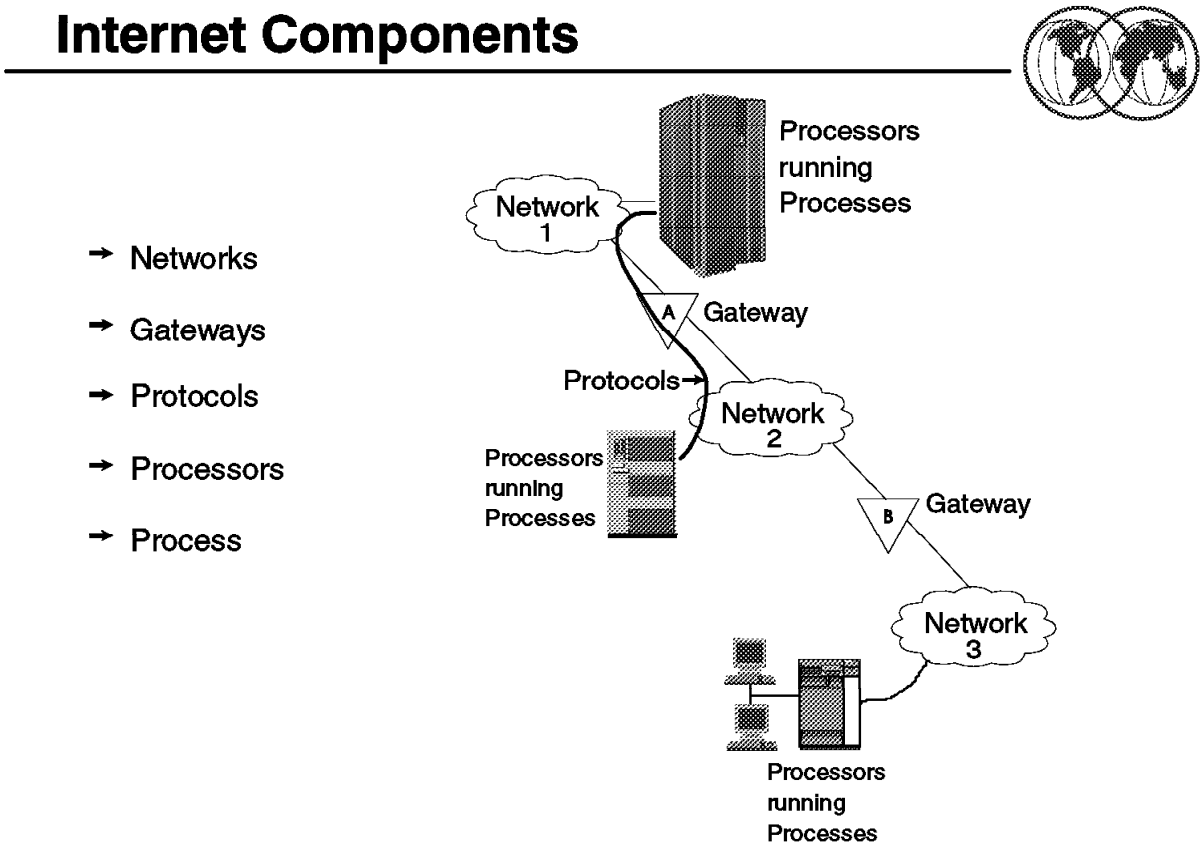


Figure 24. Internet components

1.15.1 Internet components

As shown in this visual, an internet is comprised of various *networks*, *gateways*, and *protocols*. In addition, there are the *processors* and *processes* that we refer to as the attached entities to the internet. Processes can be considered equivalent to users and applications, and processors can be considered equivalent to hosts or machines.

The internet components are as follows:

Processor A processor is a machine that runs the TCP/IP Protocol suite.

Process A process is a program that utilizes, and has defined interfaces to, TCP/IP. Examples include TELNET, FTP, NFS, and so on.

Network A network is any existing physical network, including LANs, WANs, X.25, and so on.

Gateway A gateway is a special processor, or a function added to an existing processor, which supports addressing and routing of datagrams from one physical network to another in an internet.

A protocol is a member of the extended TCP/IP Protocol suite.

These components provide as follows:

- The ability to collect open system data.
- The ability to provide names to managed objects.

- The ability to change the configuration of an internet (for example, delete or add systems) without interrupting communications between hosts currently on the internet.

Internet Concepts

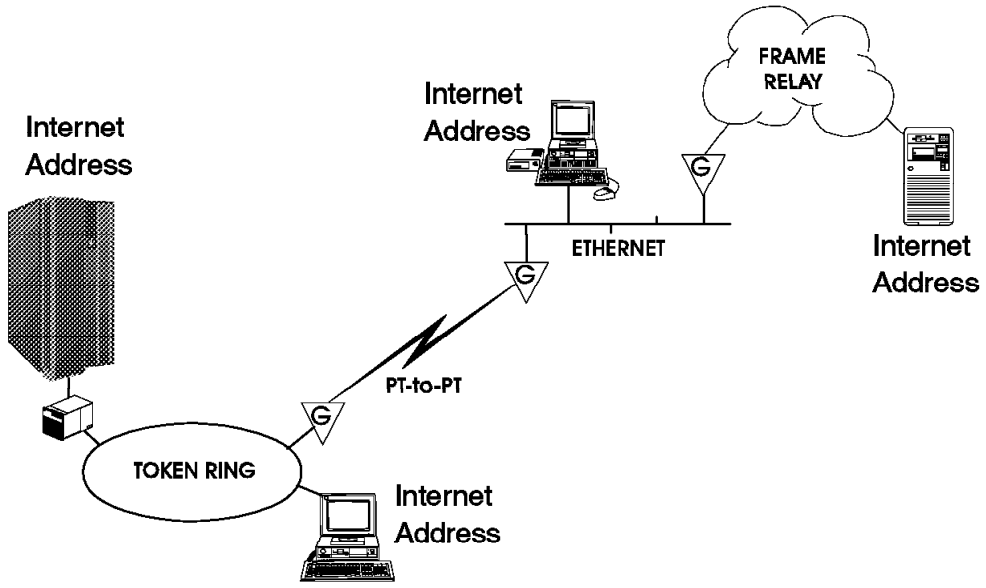


Figure 25. Internet concepts

1.15.2 Internet concepts

The basic concepts behind the internet technology are:

- Network technology independence
  - Use existing transport networks
- Universal interconnection
  - Universal addressing and naming
- End-to-end acknowledgements
  - No acknowledgements within the underlying “unreliable” networks
- Multi-vendor interoperability at the *application* level

The basic concept behind an internet is to utilize the existing network transport mechanisms by adding an additional layer of addressing between an application and the communication facilities. As much as possible, isolate users from the details of the structure of the internet and its operations such as routing, management, and so on.

- Network technology independence - While TCP/IP is based on conventional packet switching technology, it is independent of any particular vendor’s hardware. An internet may include a variety of network technologies ranging from networks designed to operate within a single building to those designed to span large distances. TCP/IP protocols define the internet unit of data transmission, called a *datagram*, and specify how to transmit datagrams on a particular network.

- Universal interconnection - A TCP/IP internet allows any pair of computers to which it attaches to communicate. Each computer is assigned an *internet address* that is universally recognized throughout the internet. Every datagram carries the addresses of its source and destination. Intermediate switching computers use the destination address to make routing decisions.
- End-to-end acknowledgements - The TCP/IP internet protocols provide acknowledgements between the source and ultimate destination instead of between successive machines along the path, even when the two machines do not connect to a common physical network.

The internet technology also allows for distribution of the responsibility and control of an internet as much as possible in the areas of routing, naming, network management, and addressing.

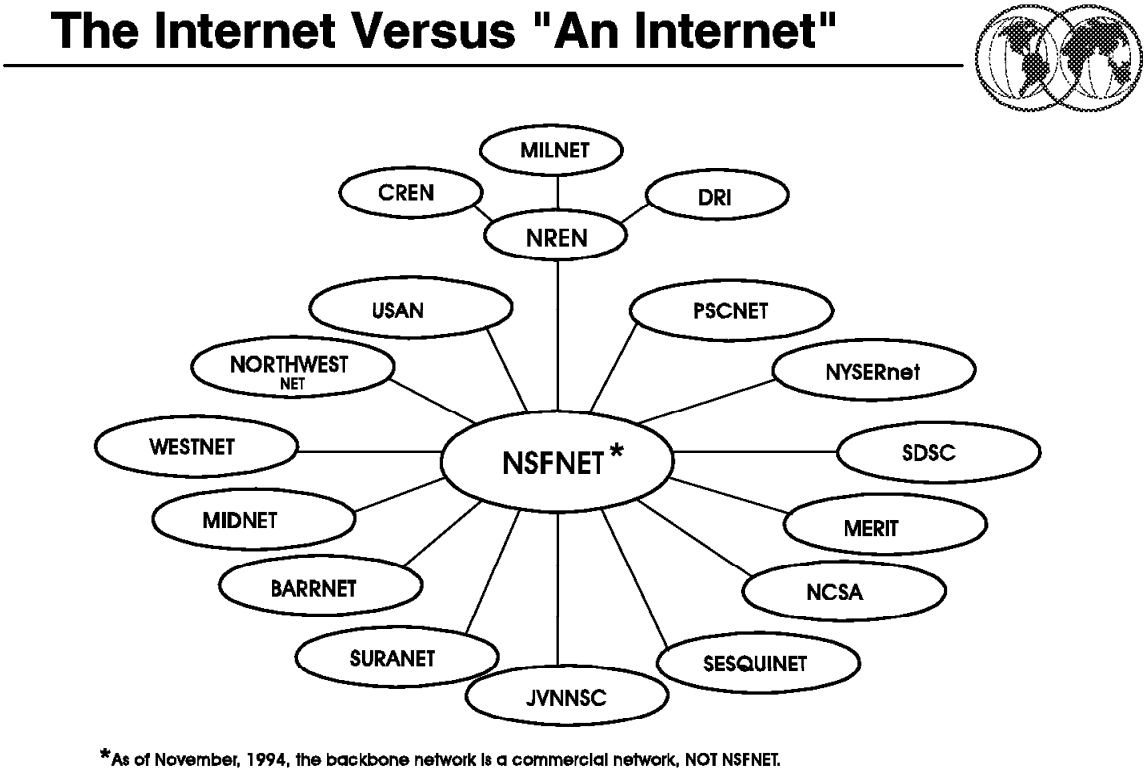


Figure 26. Internet versus an internet

1.16 The Internet versus an internet

A distinction must be made between "*an internet*" and "*The Internet*" (sometimes referred to as *the connected Internet*).

"The Internet" (capital I) is a specific, large internet that connects most major research institutions, including universities, corporate and government labs, businesses (both local and international) your home, and so on. It was created when the Defense Advanced Research Projects Agency (DARPA) began working toward an internet technology in the mid 1970s.

The Internet has experience explosive growth in the past few years. This is especially due to the growth of the *World Wide Web*, a collection of sites on the Internet that support easy, end-user-friendly access and links to data stored at Internet sites all over the world.

Because of this explosive growth, a true picture of the Internet is probably impossible to draw. The picture shown on this visual is a reasonable approximation of the Internet as it appeared around 1990–1991.

Remember from the previous visuals that an internet is a group of separate physical networks connected together allowing for communication between any computers attached to the internet.

However, *the Internet* is a specific internet that has been formed through the work and cooperation of various military, educational, and corporate research organizations. It is sometimes also referred to as the *connected Internet*.

The first pieces of the Internet came together around 1980, when DARPA first started converting its machines to use TCP/IP protocols. The ARPANET, DARPA′s packet-switching network that was already in place, quickly became the backbone of the new Internet and was used for many early experiments with TCP/IP. The transition to the Internet was completed in 1983 when TCP/IP became mandatory for connections to the DARPA network, ARPANET.

The success of the TCP/IP technology and the Internet led other networks to adopt it. The National Science Foundation (NFS) took an active role in expanding the TCP/IP Internet to reach as many scientists as possible. Starting in 1985, the NSF began a program to establish access networks centered around its six supercomputer centers. In 1986 it expanded networking efforts by funding a new backbone network called the NSFNET, that reached all its supercomputer centers and tied them to ARPANET, all becoming part of the Internet. Advances in technology caused the ARPANET to be dropped in favor of the NSFNET.

Today, the NSFNET backbone network is no longer supported by the National Science Foundation. It is instead now a commercial network with installations all over the world.

- MILNET - Military network
- CREN - Corporation for Research and Educational Networking—merger of CSNET (the Computer+Science Network) and BITNET (the Because It′s Time Network)
- DRI - Defense Research Internet
- NREN - National Research and Education Network
- NSFnet - The National Science Foundation network links the supercomputer sites to each other through a high speed backbone. The network then interconnects to the many regional networks forming a network of networks. IBM, in a joint development project with MCI and Merit Inc., provides software and switching system hardware for NSFnet.
- NYSERnet - New York State
- JVNNSC - John von Neuman Center Consortium
- SURANET - Southeastern Universities Regional Network
- SDSC - San Diego State Consortium
- MIDnet - Midwest Networks
- BARRNET - Bay Area Regional Networks

Internet Guiding Entities

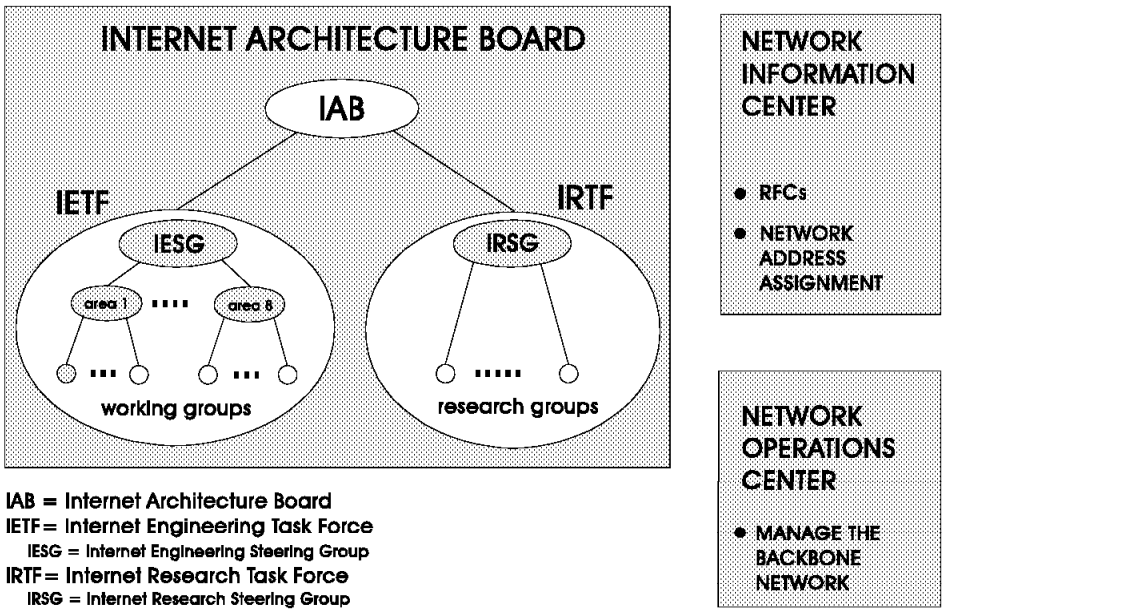


Figure 27. Internet guiding entities

1.17 Internet guiding entities

There are various organizations, or guiding entities, that provide the focus for the research and development of the Internet protocols, as listed here.

There are also “Requests for Comment” or RFCs that are reports of work, proposals for protocols, and protocol standards.

The various Internet guiding entities are listed on this visual. These entities include:

- The Internet Activities Board (IAB) which provides the focus for most of the Internet Research and Development. The IAB consists of:
  - Internet Engineering Task Force (IETF) which is comprised of the Internet Engineering Steering Group (IESG) and members who make up the working groups in each of eight areas. The IETF concentrates on short-term or medium-term engineering problems.
  - Internet Research Task Force (IRTF) which is comprised of the Internet Research Steering Group (IRSG) and members who make up the research groups. The IRTF coordinates research activities related to TCP/IP protocols or internet architecture in general.
- The Network Information Center (NIC) which is run by SRI International under contract and is responsible for the assignment of network addresses. The NIC also maintains and distributes information about TCP/IP and the connected Internet.

Requests For Comments (RFC) contain Internet standards and other useful information. They may be long or short and may be wide or narrow in scope. The Network Information Center distributes the RFCs to the community either electronically or in hard copy.

- The Network Operations Center (NOC) which manages key Internet networks, core gateways, and does network management on behalf of all users. The NOC is operated by MERIT, Inc. and is funded by NSF. It is located in Ann Arbor, Michigan, USA.

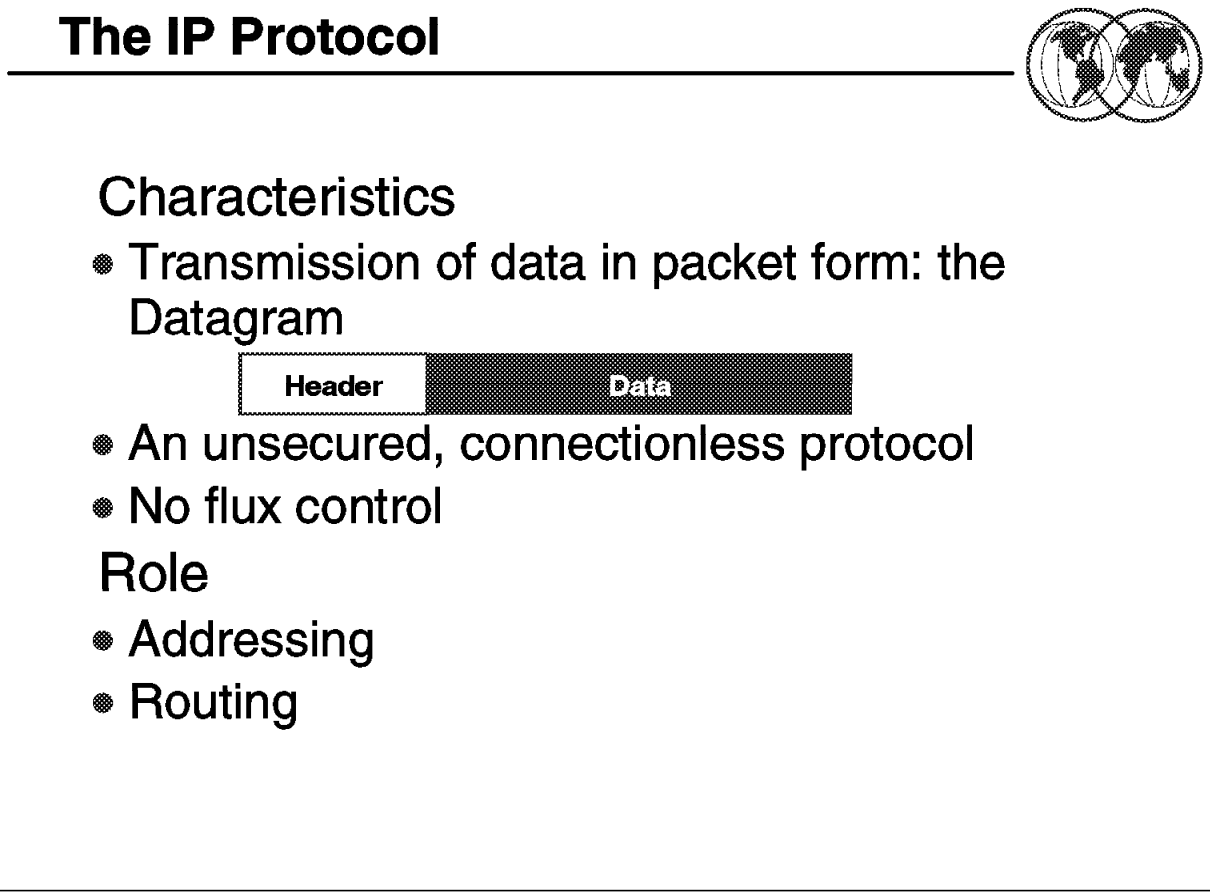


Figure 28. Internet addressing

1.17.1 Internet addressing

Because of the ability to interconnect computers across multiple networks, an appropriate addressing scheme was designed along with the networking protocols. All users of the network (including applications) are located at hosts. “Host” means any computer, whether it be a large system, a small PC, or anything in between.

Each host must have an unique address to communicate with other hosts on the same internet. An internet address contains 32 bits and is divided into four octets (8 bits each). The address consists of two major parts:

- A network address - used to address a specific network in an internet
- A host address - used to address a specific host within a network

The two major parts of the internet address are often shown as follows:

internet address = <network address><host address>

The internet address can be represented in a dotted-decimal form, where each octet is denoted as the decimal representation of the binary octet.

As you can see in the visual, a sample 32-bit address is shown:

10000100 00001101 00001110 00000010

However, humans have a difficult time interpreting a 32-bit binary address. Therefore, a *dotted-decimal form* of notation is used to represent an internet address. So the binary number shown above becomes:

132.13.14.2

As mentioned before, an IP internet address consists of two parts; the <network address> and the <host address>. Depending on the number of octets used to represent the <network address> part of the internet address, there are four main classes of internet addresses. We will discuss those on the next visual.

The important thing to remember is that each host on an internet must have a unique internet address to be attached to the internet. If a host is on a private internet, then the group who manages access to that internet must be consulted for a unique address. If you wish to connect a host to *the Internet*, then you must have the NIC assign the <network address> part of your internet address to be assured of a unique address on the Internet.

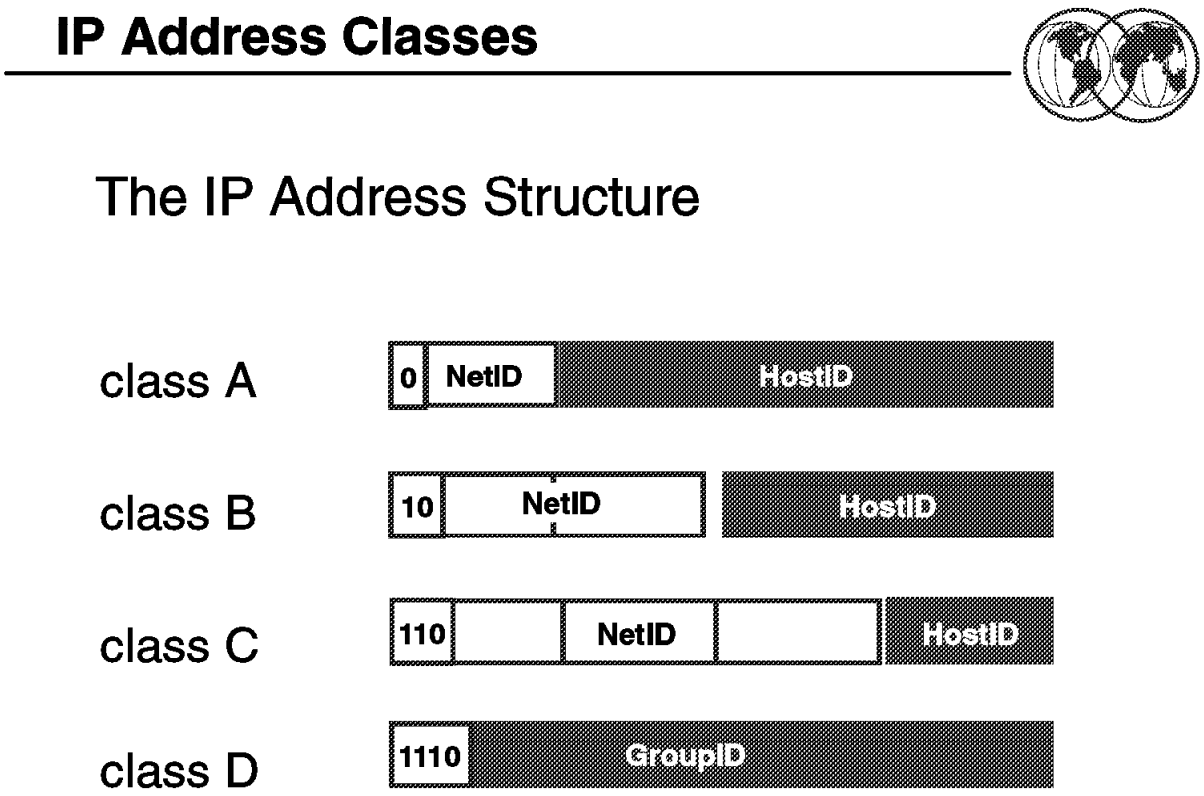


Figure 29. Internet address classes

1.17.2 IP address classes

Each IP internet address is a pair of addresses (<network address><host address>), where <network address> identifies a network, and <host address> identifies a specific host on that network. Each host attached to the same physical network has the same <network address>.

- The <network address> portion of an internet address is assigned by a central authority. In the case of the Internet, the NIC must assign the <network address> part of the Internet address. This <network address> must be unique throughout the big Internet.
- The <host address> is assigned by the local authority and must be unique within the specific network.

The number of bits in an internet address that are assigned to the <network address> part are variable, resulting in four main classes of networks:

- Class A networks are very large networks (many hosts)
- Class B networks are moderately large networks
- Class C networks are small networks (256 or less hosts)
- Class D addresses are used for ′multicast′ addresses

For the four classes mentioned above the division between the <network address> and the <host address> is:

- Class A allows seven bits of the internet address to be used for the <network address> portion; and the remaining 24 bits to be used for the hosts in the network. It allows for a total of 128 different networks with 16.777.216 hosts in each network.
- Class B allows 14 bits of the internet address to be used for the <network address> portion; and the remaining 16 bits to be used for the hosts in the network. It allows for a total of 16.384 different networks with 65.536 hosts in each network.
- Class C allows 21 bits of the internet address to be used for the <network address> portion; and the remaining eight bits to be used for the hosts in the network. It allows for a total of 2.097.152 different networks with 256 hosts in each network.
- Class D is used for “multicast” address (limited broadcast addresses).

When hosts are moved from one network to another they must have the network address portion of their internet address changed and will probably also require a change in the host address portion of their internet address.

Address nuances:

- The <host address> is never set to zero, so an address with host address equal to zero refers to the network itself. For example

  The IBM network is 9.0.0.0

  Further convention: eliminate trailing zeros in dotted-decimal notation, so the IBM network is often written as simply 9.
- A <host address> of all ones is a broadcast address (on the local network).
- A <network address> of 0 is taken to mean “this network.”

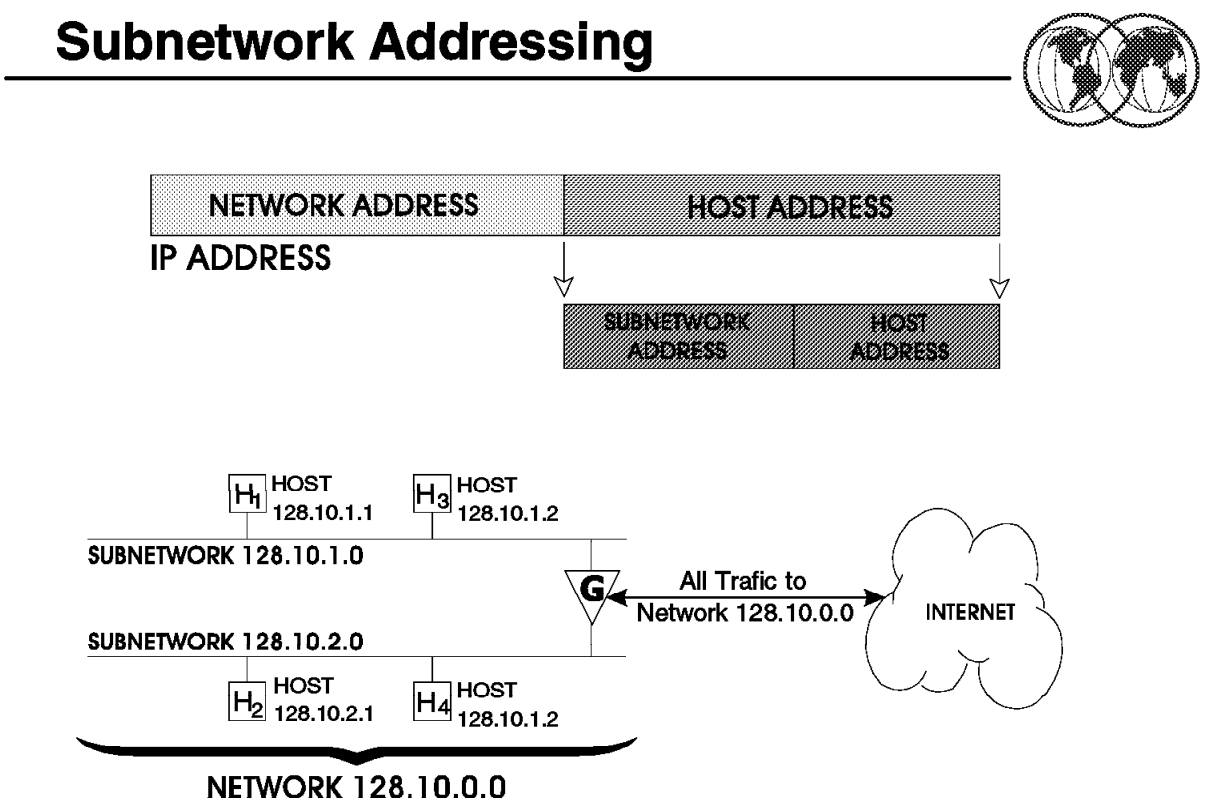


Figure 30. Subnetwork addressing

1.17.3 Subnetwork addressing

Subnetwork addressing allows large networks to be broken into smaller networks called *subnets*.

Where:

- Subnets are implemented by maintaining the integrity of the <network address> portion of the internet address, but dividing the <host address> portion into two separate parts:
  - Subnetwork address (sometimes referred to as a “local network address”)
  - Host Address
- The resulting IP address can then be represented as follows:

  <network address><subnetwork address><host address>

When the IP addressing scheme was developed, the designers worked in the world of expensive mainframes. They planned for tens of networks with hundreds of hosts. They did not plan for *tens of thousands* of small networks (LANs) with *hundreds of thousands* of personal computers. Growth in the connected Internet is most apparent. A large number of trivial networks stresses the entire Internet design because it means:

- Large administrative overhead is required to manage network addresses.
- Routing tables in gateways are extremely large.

Therefore, the principle of assigned IP addresses is too inflexible to allow easy changes to local network configurations. Those changes might be:

- A new type of physical network installed at a location
- Growth of the number of hosts (personal computers) requires splitting the local network into two or more separate networks
- Growing distances require splitting a network into smaller networks with gateways between them

To avoid having to request additional IP network addresses for every change in a local network, the concept of *subnets* was introduced. Subnets are an extension to the base IP addressing scheme that considers a part of the <host address> to be a “local network address” or <subnetwork address>. IP internet addresses are then interpreted as:

<network address><subnetwork address><host address>

- The assignment of subnets can be done locally, as the whole local network still appears to be on IP network to the outside world.
- Division of the original <host address> part into a <subnetwork address> and <host address> part can be chosen freely by the local administrator. However, once it has been established, it must be used consistently throughout the whole local network.
- Implementation of subnets requires that all the hosts have the IP code which supports subnetworking.

We will discuss subnets in more detail when we discuss routing.

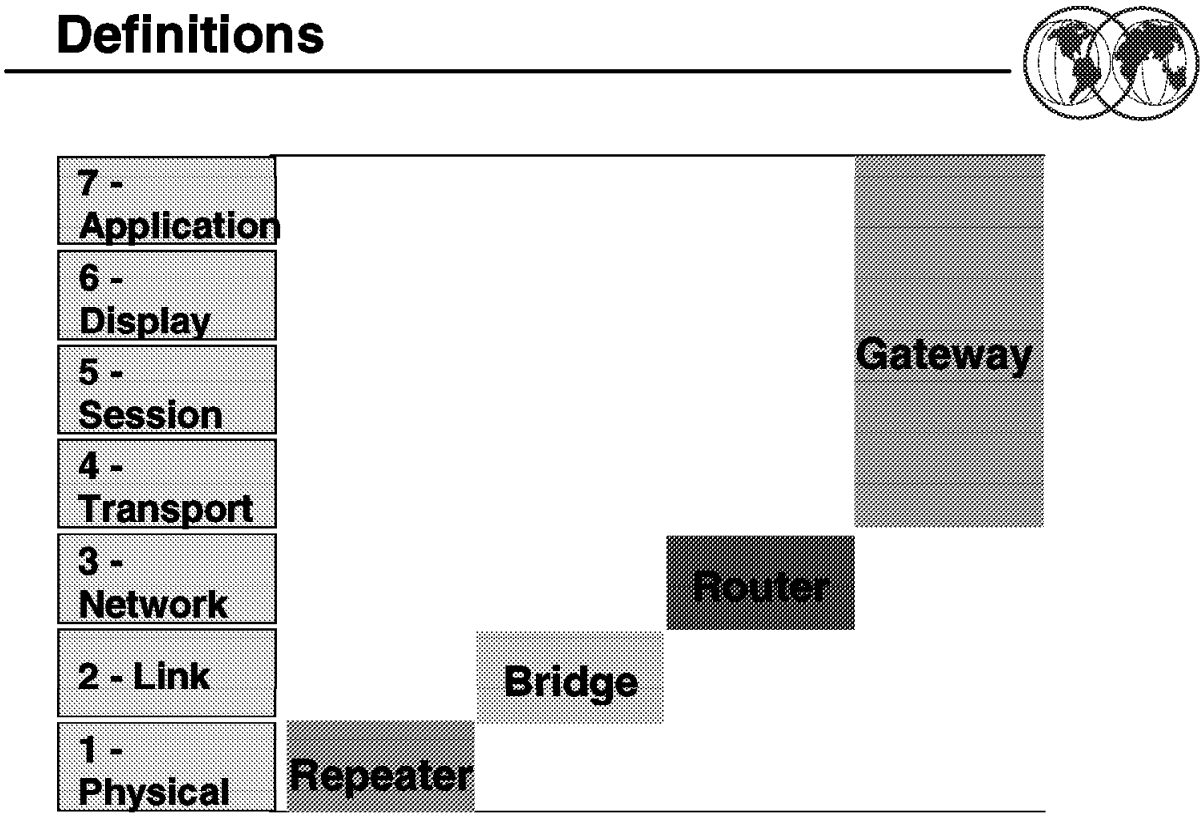


Figure 31. Definitions

1.18 Network definitions

There are several ways to interconnect networks together. Bridges, routers, and gateways all interconnect networks; however, each does so differently.

- Bridges
  - A bridge connects networks at the Link Control Layer. It is protocol independent. It does *not* have an IP addresses.
  - Bridges connect networks, or segments of the same network, at a very low functional level.
  - Portions of the link header and trailer may need to be changed, but most of the data in the frame is left unaltered.
  - Bridges are the most efficient, since they operate at the Data Link Control layer and usually require the least amount of processing.
- Routers
  - A router connects networks at the Network Layer. It is protocol dependent. *In TCP/IP, a router is often referred to as a gateway.*

    A TCP/IP router has an IP address for each network it is attached to.
  - Routers are used to route frames (or packets as they are also called) from one network to another.

  - Routing requires changing one or more addresses, making decisions as to the best route or path to take through the interconnected networks and possibly changing link headers and trailers.
  - Routers usually are less efficient than bridges, but more efficient than gateways.
  - A source of confusion in the TCP/IP world is that computers which perform routing functions are often referred to as “gateways.” A TCP/IP “gateway” is not a gateway as described below; it is a router.
- Gateways
  - A gateway (non-TCP/IP gateway) connects networks of same or different protocols. It is protocol dependent.
  - Gateways are used to change protocols from one set to another. This usually means removing the information from all the protocol envelopes and repackaging it in another set of protocol envelopes.
  - Gateways are the least efficient way to interconnect networks, but allow the interconnection of networks with different protocol implementations.

TCP/IP “gateways” can connect networks with different physical implementations (for example, LAN and X.25), as long as the same higher-level network implementation (that is, IP) is used.

Therefore TCP/IP gateways are in reality routers.

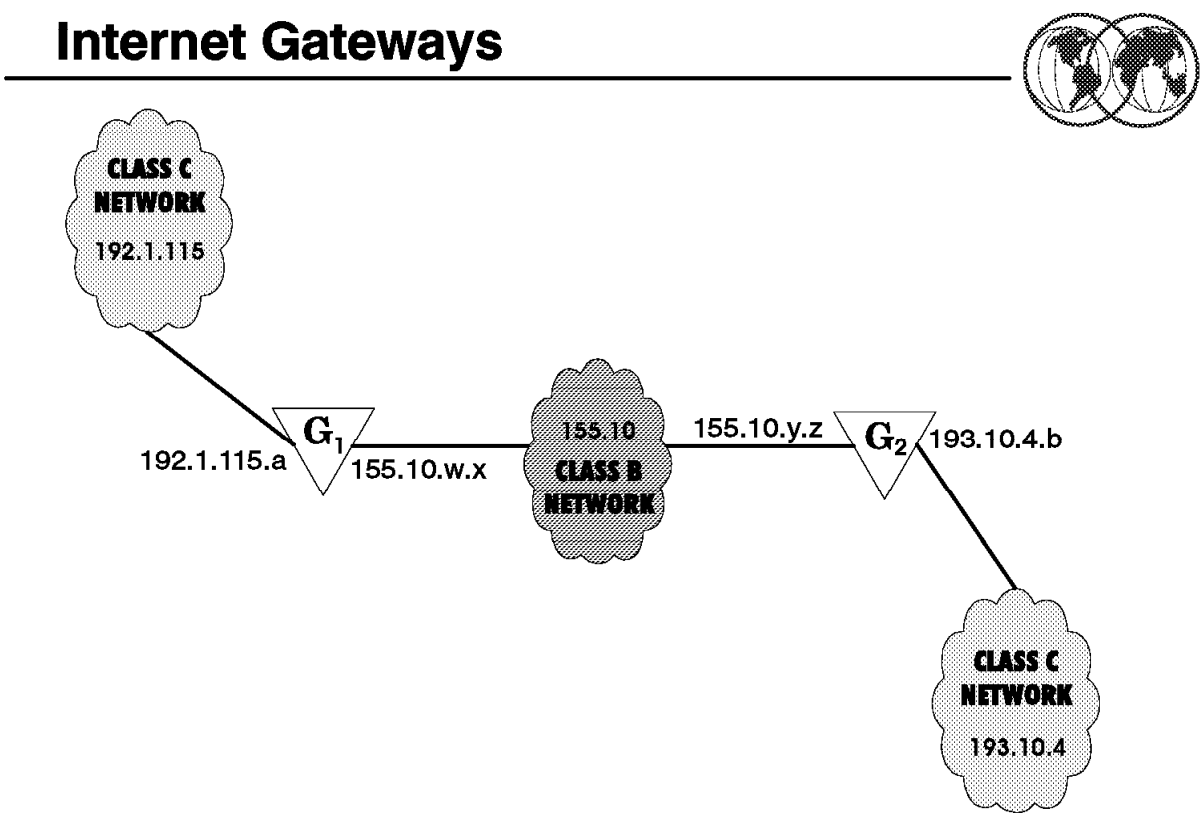


Figure 32. Internet gateways

1.19 Internet gateways

TCP/IP can be found in many different types of networks, and, in fact was designed to support the interconnection of these networks using what is known as the “gateway” function.

Internet gateways have more than one address; an address is used to represent the gateway on each network the gateway is attached to. TCP/IP hosts with addresses on more than one network are called “multi-homed.”

TCP/IP gateways can connect networks with different physical implementations (for example, LAN and X.25) as long as the same higher-level network implementation, that is, IP is used. As mentioned earlier, the TCP/IP gateway function is really a “router” function. TCP/IP gateways perform their routing function based upon the destination <*network* address>, not the destination <*host* address>. Packets are routed through the network based upon their target <network address>. The <host address> portion of the internet address is ignored during inter-network routing.

Hosts and gateways know nothing about the subsequent routing. Routing is a basic gateway function and part of the base IP code; it is available in any node running TCP/IP.

On the next two visuals we will take a brief look at basic and full-function gateways.

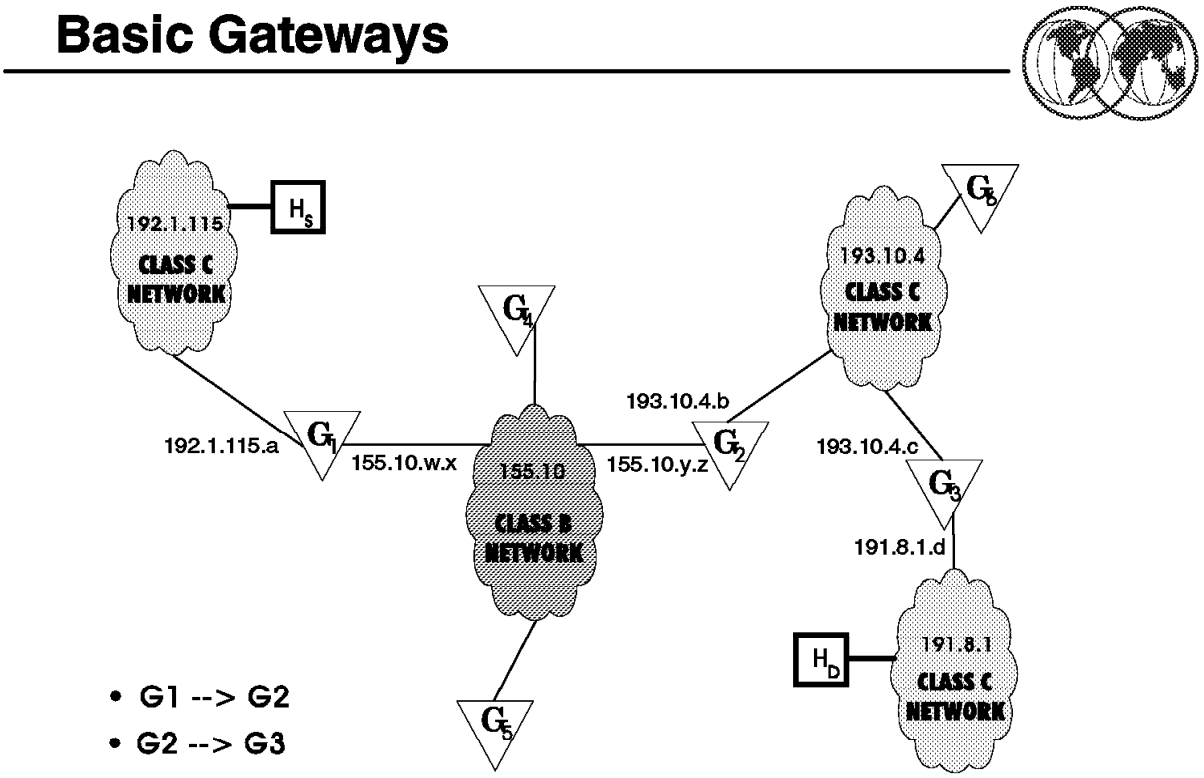


Figure 33. Basic gateways

1.19.1 Basic gateways

Basic gateways provide the function to connect two networks together. The gateway has limited knowledge of the network; it only knows about the hosts on the networks to which it is connected. Remember that gateways are “multi-homed”—they have IP addresses on each network they are attached to.

Basic gateways support “direct routing” if the destination and source hosts are on the same physical networks (same <network address>). Basic gateways also support “indirect routing” if the destination and source hosts are on different networks (routing must be done via one or more gateways). Datagrams are simply routed to the next gateway along the path.

- A source host only needs to know the address of the first gateway.
- Each gateway will forward the packet until the destination network is reached; the packet is then sent using direct routing.
- In the example on the visual, a datagram from network 192.1.115 that is destined for network 193.10.4 will be routed by gateway G1 to gateway G2.

Each host must keep an IP routing table which maps:

- Local addresses as direct routes.
- Off-network destination host addresses as indirect routes through a locally attached gateway.
- A default route for packets with a destination network address not listed in the routing table.

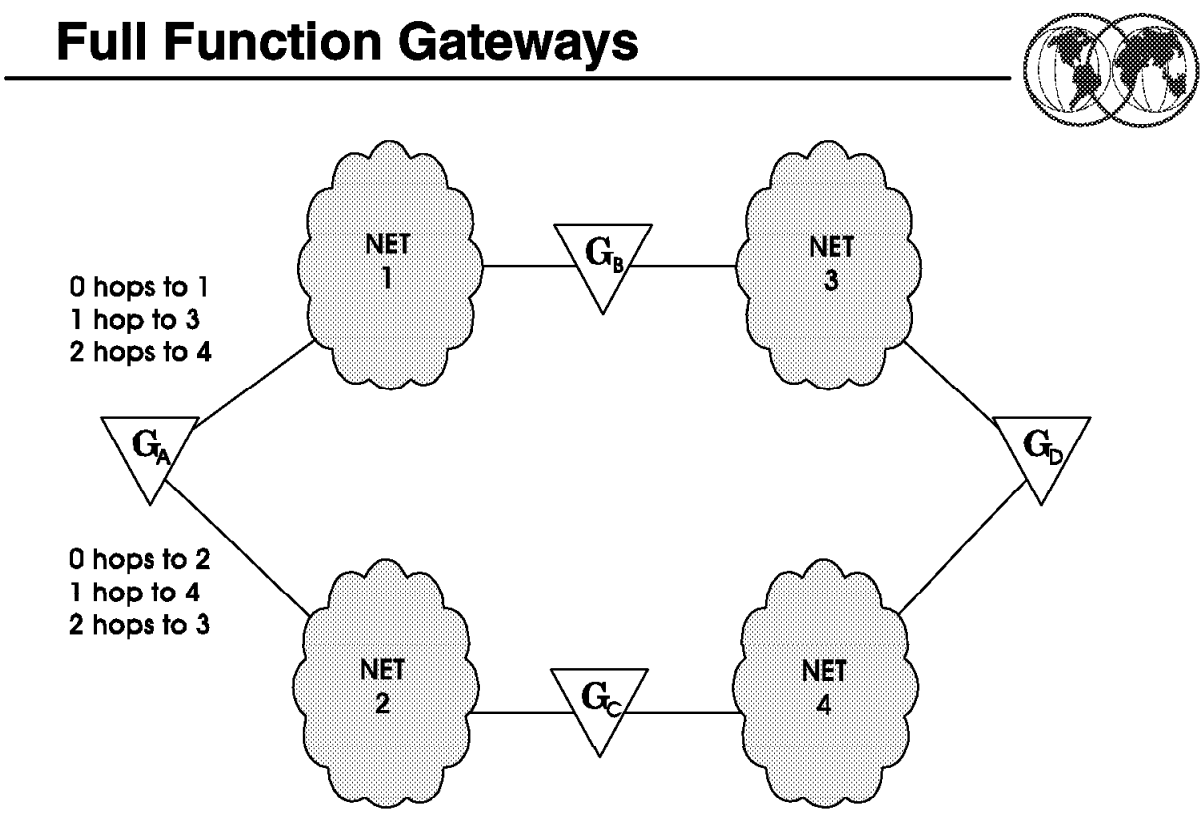


Figure 34. Full-function gateways

1.19.2 Full-function gateways

Full-function gateways provide more sophisticated functions, where the gateway knows about all interconnected networks. They maintain and update routing information dynamically to reflect network topology changes.

Each gateway has information about:

- Networks that can be reached
- Number of hops to each network (gateway hops)

Here, for example Gateway A′s routing table is shown, with the information on reaching Networks 1, 2, 3, and 4.

Full-function gateways are divided into two classes: core and non-core gateways.

- Core gateways are maintained by NIC and are used to interconnect major user networks to NSFNET.
- Non-core gateways are used to interconnect multiple user networks together.

Both core and non-core gateways must know about all of the networks they support. This is accomplished using one of the different gateway protocols.

We will discuss the different gateway protocols on the next visual.

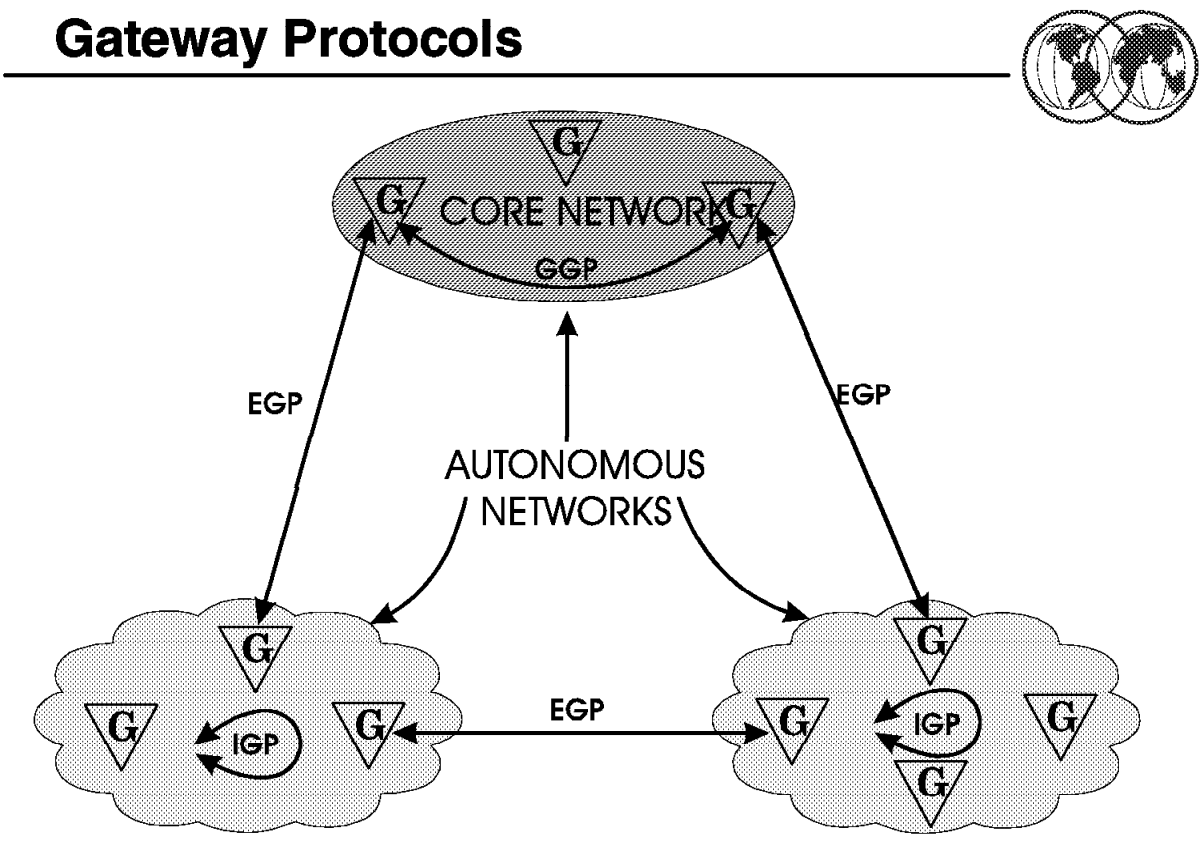


Figure 35. Gateway protocols

1.19.3 Gateway protocols

There are several protocols associated with the transfer of information between gateways (routers). The specific protocol(s) used are determined by the gateway type:

- Gateway-to-gateway protocol (GGP)
  - Used to transfer information between “neighbor” gateways.
  - Each gateway will update its tables based upon the received information and will send information about changes to its neighbors.
  - Information transferred includes the networks reachable by this gateway, and the path length for reaching the networks.
- Exterior gateway protocol (EGP)
  - Used to transfer information between “exterior” gateways. (Exterior gateways are ones that connect physical networks in different *autonomous networks*, where an *autonomous network* is a group of physical networks connected together and administered by a single authority.)
  - The EGP protocol supports passing of route information as well as other dialogs between the exterior gateways.
  - Exterior gateways collect information from interior gateways (gateways within an autonomous network.
- Interior gateway protocols (IGP)
  - A set of protocols used to transfer information between “interior” gateways.

- Examples of IGP protocols include the Routing Information Protocol (RIP) and the HELLO protocol.
- RIP is a class of protocols based upon the Xerox network XNS routing protocols.

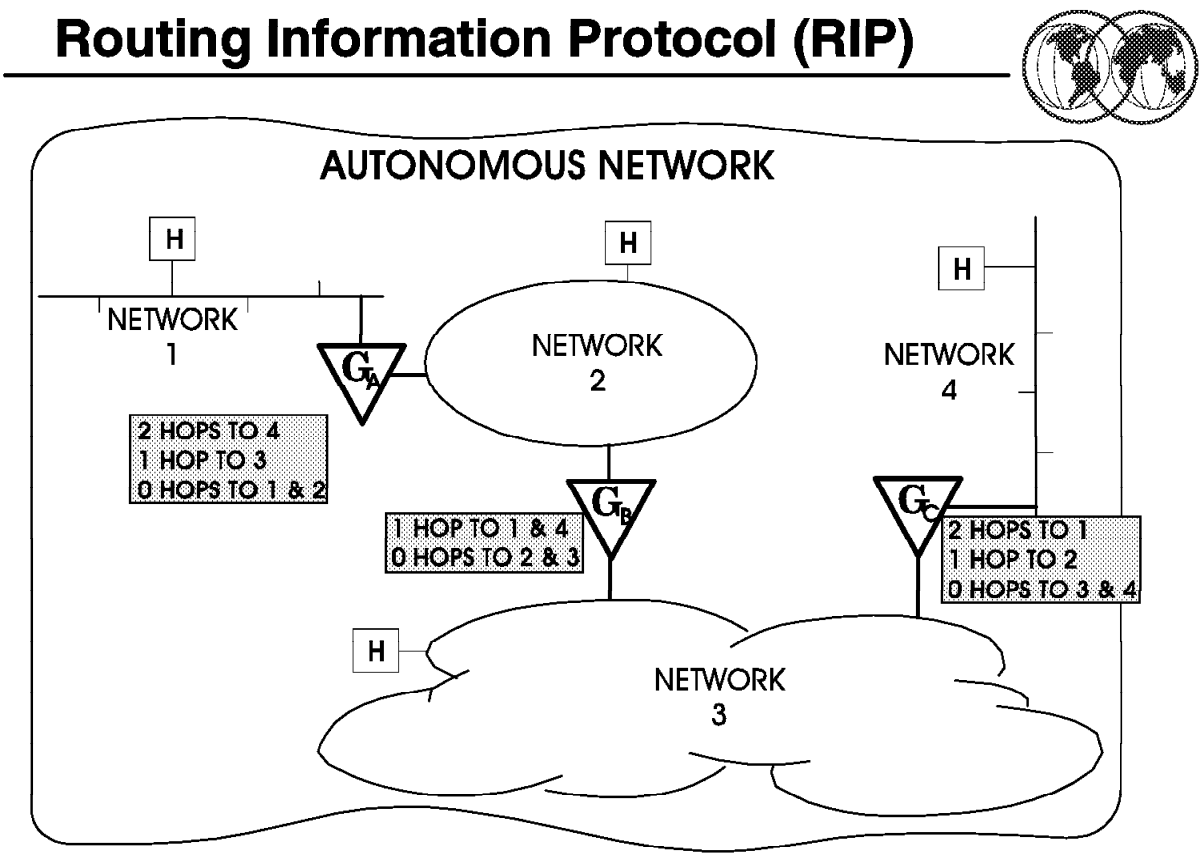


Figure 36. Routing information protocol

1.20 Routing information protocol

Autonomous networks implement the RIP protocol. RIP or routing information protocol is a widely used interior gateway protocol. It relies on physical network broadcasts to make routing exchanges quickly.

RIP routing tables consist of the distance to each physical network in the autonomous network that the gateway is aware of. The distance to that network is measured as the number of hops (gateways between the sending gateway and the network).

RIP partitions hosts in the autonomous network into *active* and *passive (silent)* machines. Active gateways advertise their routes to others; passive machines listen and update their routes based on advertisements, but do not advertise. Typically, gateways run RIP in active mode, while hosts use passive mode. The routing information that is broadcast is that of the sender of the broadcast. Active gateways broadcast their routing table every 30 seconds.

In this example, Gateway A sends Gateway B its current routing database, consisting of the information that Network 1 is 0 hops from A and Network 4 is 2 hops from A. Similarly, Gateway B sends Gateway A and C, its current routing database, with the information that Network 1 is 1 hop from B and Network 4 is 1 hop from B. Each gateway updates its routing database with the latest information that it receives.

A hop count of 15 denotes infinity, so as to avoid routing loops.

RIP uses vector-distance routing for local networks. RIP is also known as route-d.

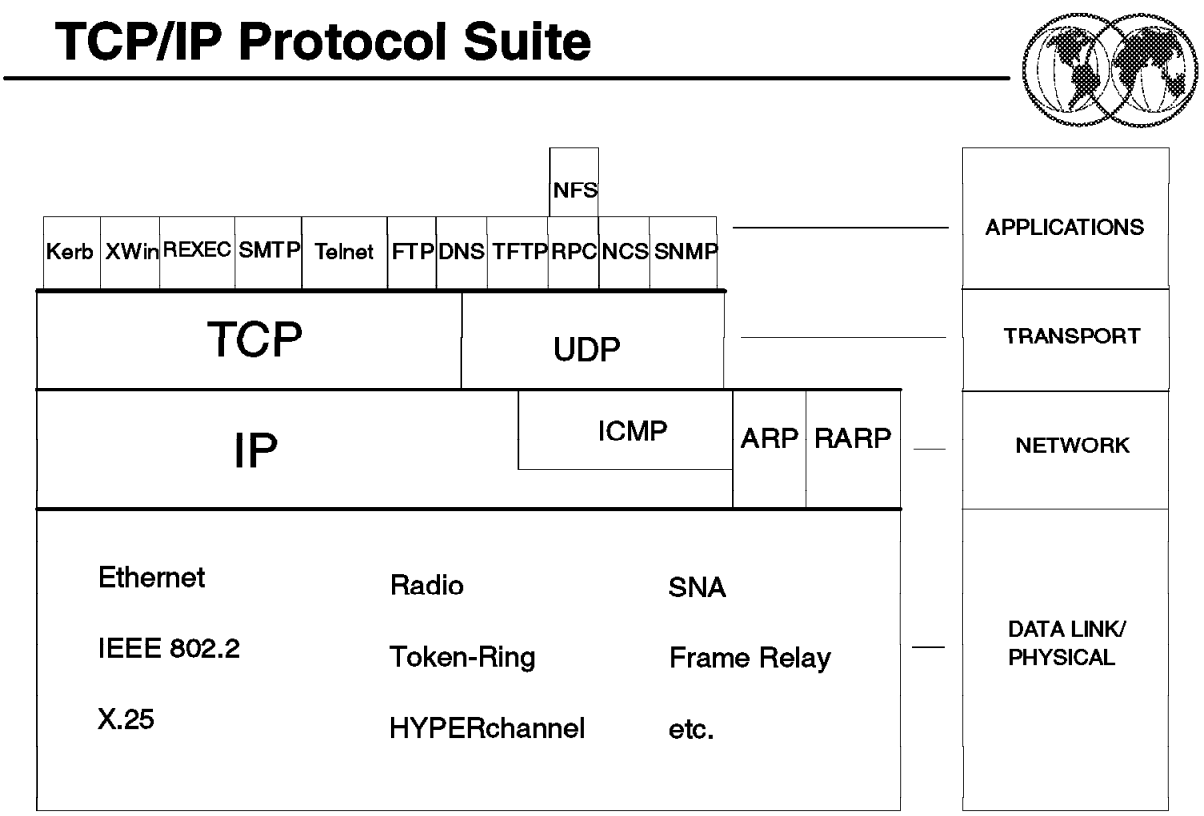


Figure 37. TCP/IP protocol suite

1.21 TCP/IP protocol suite

Remember that the TCP/IP protocol suite does not refer just to TCP and IP. It consists of:

- Application protocols, such as TELNET, FTP, SMTP, and others.
- The key inter-networking protocols:
  - Transmission Control Protocol (TCP)
  - the Internet Protocol (IP)
- Other networking protocols include User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), and so forth.
- The TCP/IP protocol suite also includes the Network Access Interface protocols, which are the interface protocols for the underlying physical network.

In this subtopic, we will go into a little more detail about some of the TCP/IP protocols and what they are used for. The purpose of this architecture review is to make sure you are familiar with the purpose of these protocols before we get into implementation specifics.

Remember that architecturally, TCP/IP is a layered architecture, consisting of a protocol stack comprised of four basic layers as seen in this visual. They are:

- The application layer, with the application protocols, which provide applications that the users can invoke to access network services. The application layer is built on the services of the transport layer.

Some of the application protocols are shown here. We will cover these in more detail in the section on TCP/IP Applications.

- The transport layer consisting of the two different protocols: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP)—TCP and UDP. TCP is a connection oriented protocol, whereas UDP is a connection-less protocol.
- The internet layer consisting of the protocols Internet Protocol (IP), Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), Routing Information Protocol (RIP), and Reverse Address Resolution Protocol (RARP).

  IP is a connection-less protocol.
- The data link or physical layer with protocols such as Ethernet, X.25, Token Ring, and so forth.

First, we will discuss the concept of protocol layering in the next visual and then over the next series of visuals we will provide an overview of the TCP/IP layers and the protocols at each layer.

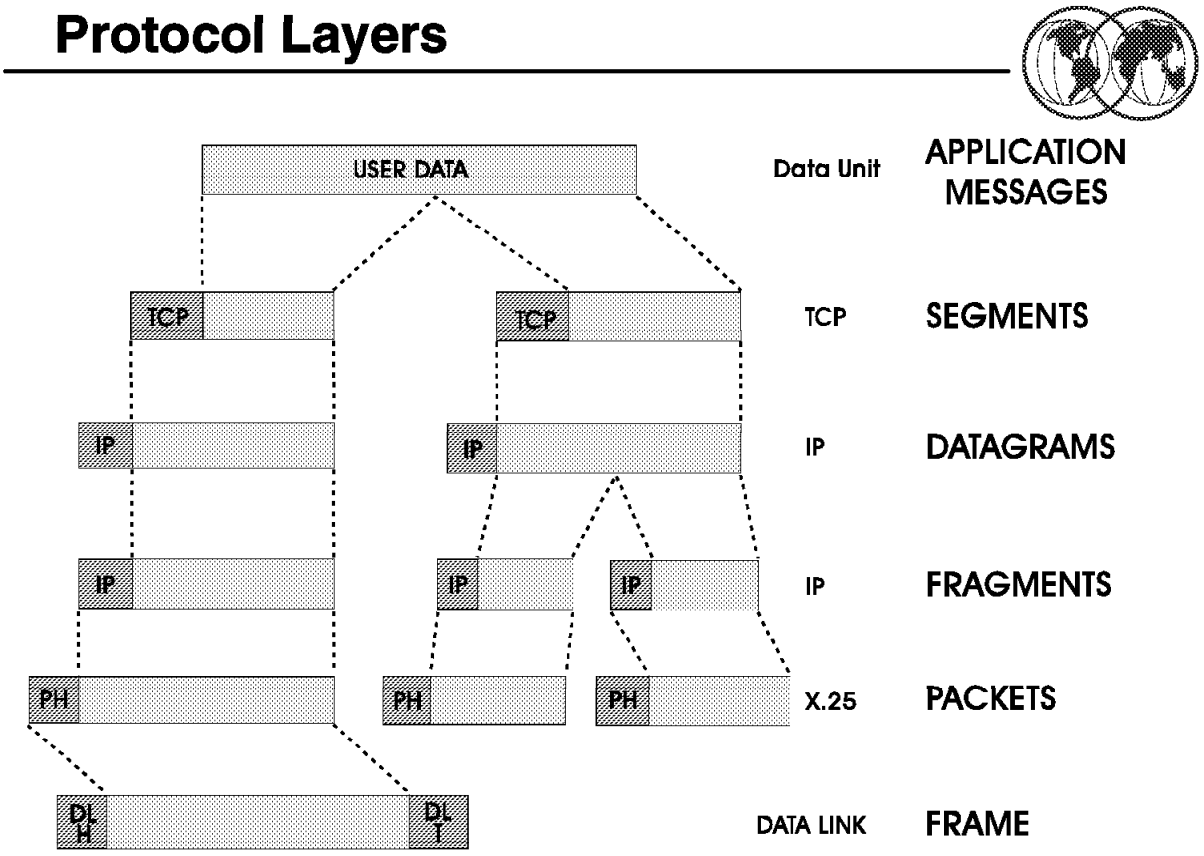


Figure 38. Protocol layers

1.21.1 Protocol layers

Each separate layer of a layered protocol is responsible for adding its own headers (and sometimes trailers) for successful peer-to-peer communication with the same layer in a protocol stack in a different machine. Each layer “N” passes transparently the information it receives from the next higher layer “N+1” after it adds the required headers (and trailers, if required). A layer “N” *Protocol Data Unit (PDU)* is simply a layer “N+1” *Service Data Unit (SDU)* plus the required layer “N” headers.

As you can see from the visual, the TCP/IP protocol suite has different names to describe the PDUs created at each layer.

The operation of layered protocols is based on a fundamental idea called the “layering” principle. It can be described as follows:

> Layered protocols are designed so that layer “N” at the destination receives exactly the same object sent by layer “N” at the source.

The layering principle explains why layering is such a powerful idea. It allows the protocol designer to focus attention on one layer at a time, without worrying about how lower layers perform. Also, as enhancements and advances are made to both software and hardware, a specific item in a protocol stack can be improved without having to completely replace all the protocol software/hardware in a network.

- The layer N Protocol Data Unit (PDU) = Layer N+1 Service Data Unit (SDU) + layer N headers.
- TCP PDUs are called *segments*.

- IP PDUs are called *datagrams*, in recognition of IP Connectionless Orientation.

  Depending on the requirements of the underlying physical network, IP may further divide datagrams into fragments, each carrying a replica of the IP header. These fragments will be less than or equal to the maximum size of the physical data unit supported by the physical network.
- PDUs for the Network Access Layer(s) depend upon the type of network utilized:
  - X.25 PDUs are called *packets*.
  - Data Link PDUs are called *frames*.

Within a layered protocol such as TCP/IP, each layer invokes the services of the layer below it while providing services to the layer above it. The commands and responses between the layers of a protocol stack are called *primitives* and are distinct from the flow of peer-to-peer information described above (PDUs, segments, datagrams, and so forth).

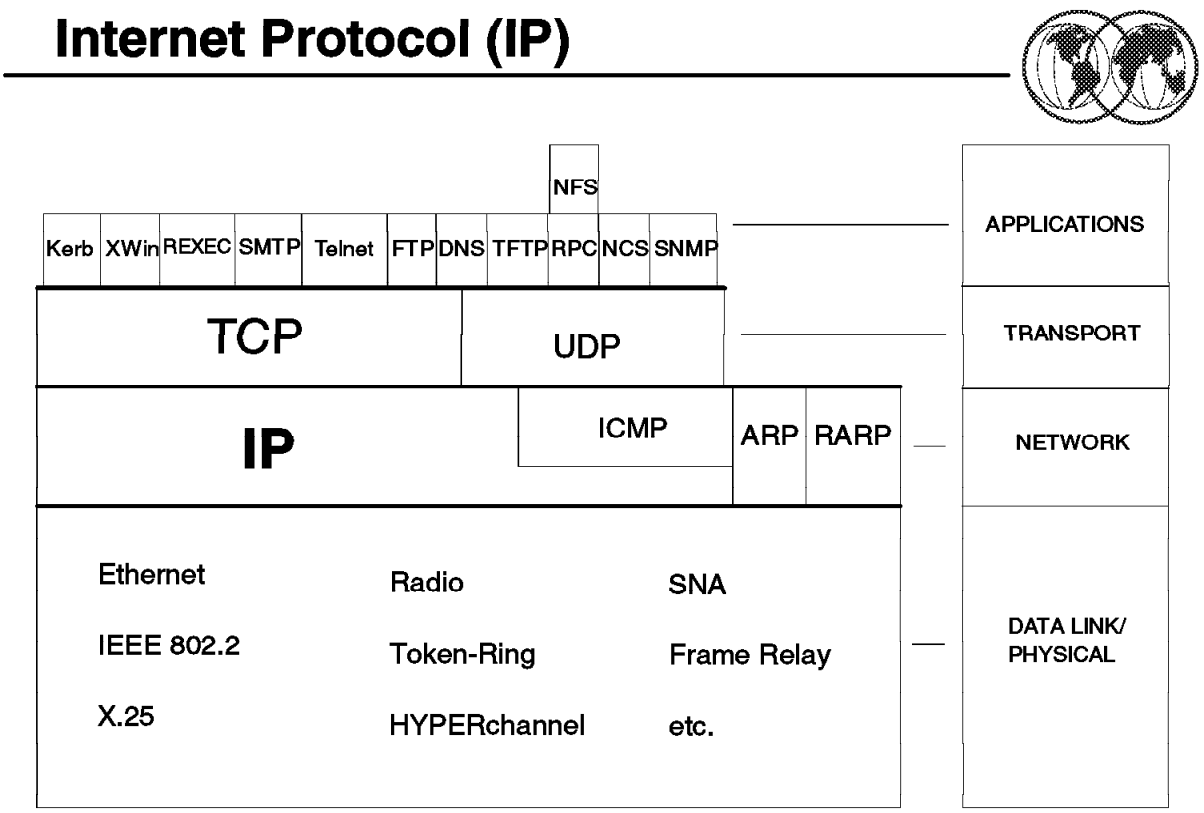


Figure 39. Internet Protocol

1.21.2 Internet Protocol (IP)

Internet Protocol or IP is the inter-network layer protocol in the TCP/IP protocol suite. It provides connectionless “datagram” delivery of data packets on an unreliable “best effort” basis. It provides no error recovery or protection against data loss. IP is faster, simpler, and more efficient than connection-oriented oriented protocols; but with less or no control of route selection and priority.

IP provides the foundation upon which the rest of the TCP/IP services rest. The most fundamental internet service consists of a packet delivery system. The service is defined as an unreliable, best-effort, connectionless packet delivery system. The service is *unreliable* because delivery is not guaranteed. The packet may be lost, duplicated, delayed, or delivered out of order, but the service will not detect such conditions, nor will it inform the sender or receiver. The service is called *connectionless* because each packet is treated independently from all others. A sequence of packets sent from one machine to another may travel over different paths, or some may be lost while others are delivered. Finally, the service is said to use *best-effort delivery* because the internet software makes a concerted attempt to deliver packets. That is, the internet does not discard packets for no apparent reason; unreliability arises only when resources are exhausted or underlying networks fail.

IP provides three important definitions:

- IP defines the basic unit of data transfer used throughout a TCP/IP internet. It specifies the exact format of all data as it passes across an internet.
- IP software performs the *routing* function, choosing a path over which data will be sent.

- IP includes a set of rules that define the unreliable packet delivery. The rules define how hosts and gateways should process packets, how and when error messages should be generated, and the conditions under which packets can be discarded.

The data travels across the network in “packets.” The basic packet used to move information is called the *IP datagram*. Since IP is a connectionless protocol, it does not provide delivery verification, guaranteed sequential delivery of packets, or error recovery. All these functions will be the responsibility of some higher-level function, such as TCP or an application.

Because of the underlying physical network limitations, IP will sometimes have to fragment a packet.

- IP expects all physical sub-networks to be able to support a packet size of at least 576 bytes.
- Data packets are assigned a unique identification number prior to fragmentation.
- When the packets are fragmented, each is sent with an offset value.
- The receiving IP can reassemble the packet using the identification number and the offset values.

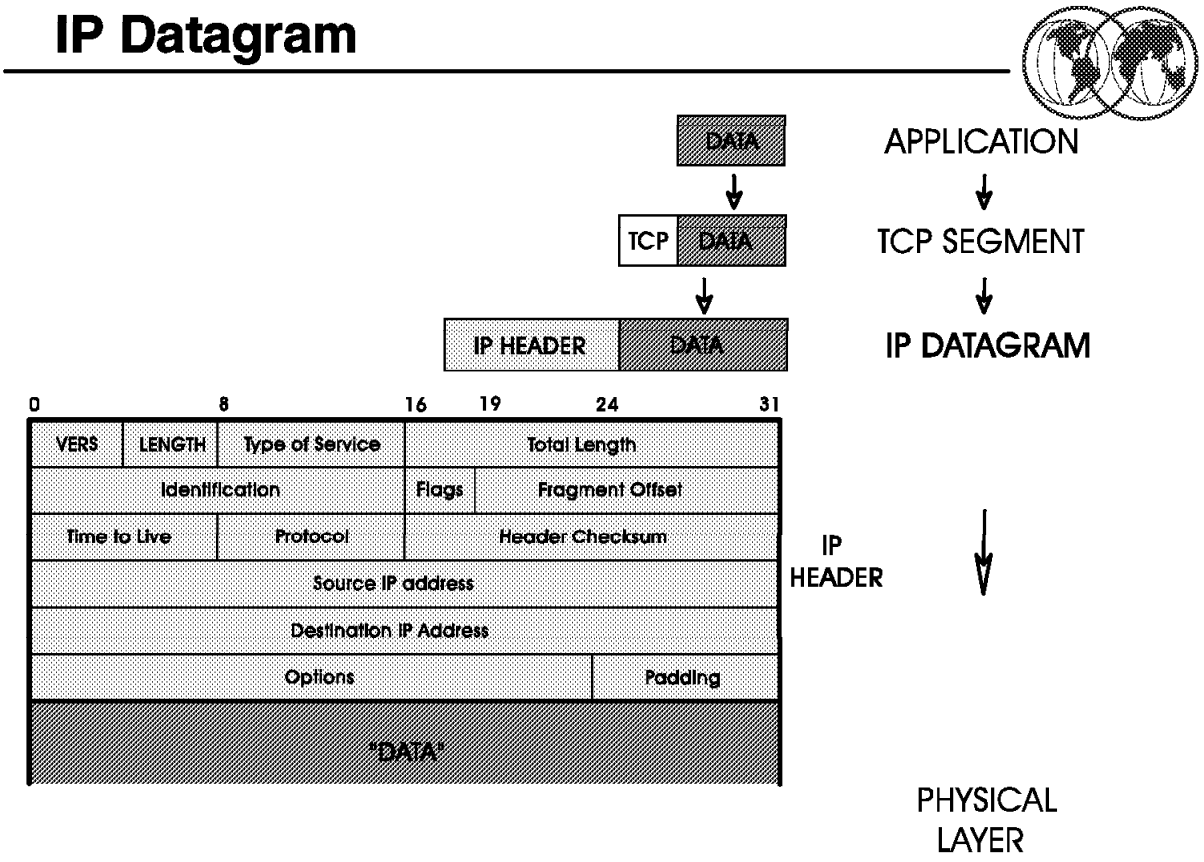


Figure 40. IP datagrams

1.21.3 IP datagrams

An IP datagram consists of:

- *Header*

  The *header* is built by IP using the data received from the next higher layer along with “primitives” from the next higher layer giving additional information on how to handle the data.

- *Data*

  Data is made up of information that is received from the next higher layer, or information received from the network that is passed up from the physical layer.

Some of the fields contained in the IP header include:

- The version of IP being used.
- Length of the IP header and total packet.
- Identification number (and offset values if fragmentation is being used).
- Higher level protocol (for example, TCP) to which IP should deliver the packet.
- Source and destination IP address.

When the IP datagram fields are completed, the packet is passed to the DLC layer where it is further enveloped.

At the destination host, IP reassembles the fragments (if any), removes the IP headers and delivers the original segments to the correct high-level protocol (TCP, UDP, and so forth).

Segments are delivered to the higher layer in whatever order they are received.

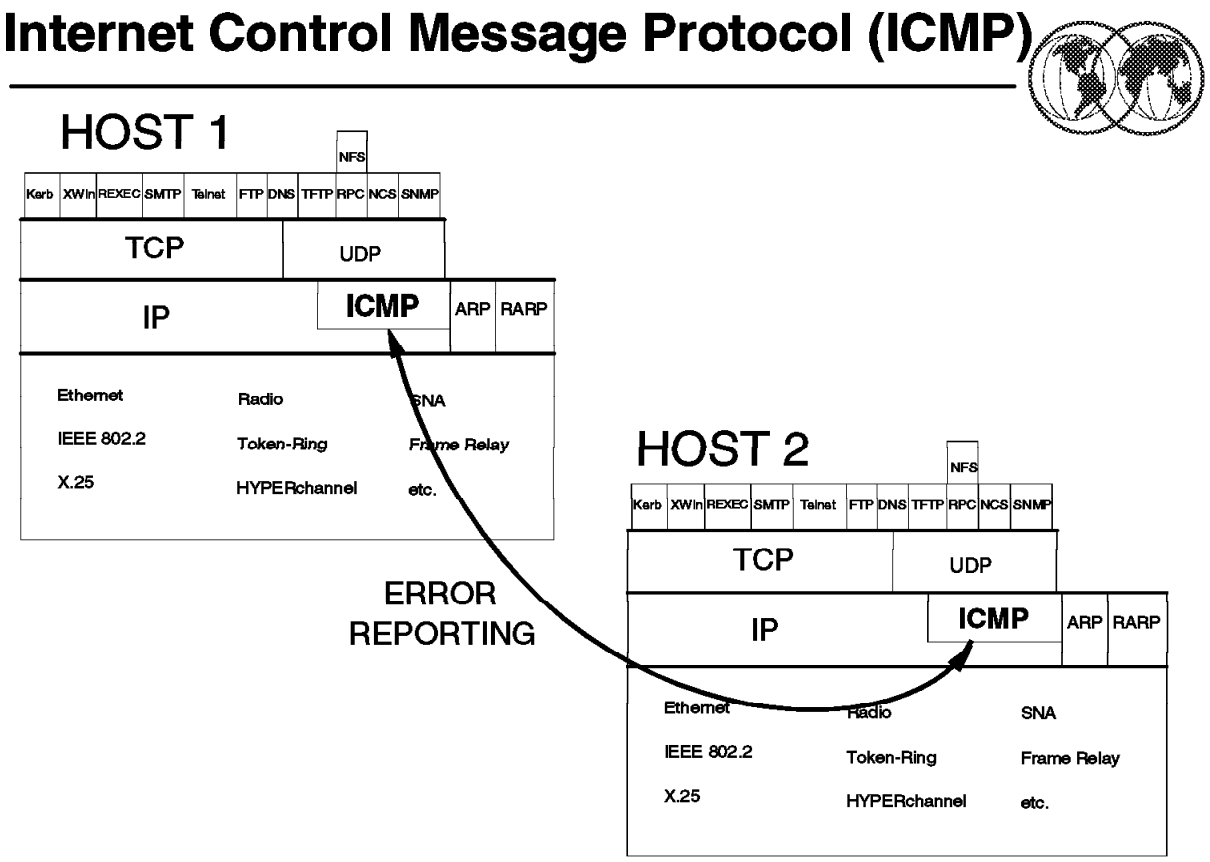


Figure 41. ICMP

1.21.4 Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol allows hosts and gateways to send error or control messages to other hosts or gateways. It provides communications between the internet software on one machine with the internet software on another machine.

ICMP messages travel across the network encapsulated in the data portion of IP datagrams. There is no added reliability, therefore the messages themselves may be lost or discarded (IP is connectionless).

ICMP is a required part of IP, and cannot be considered a higher-level protocol.

The final destination of an ICMP message is the IP software itself on the destination machine. Therefore, the ICMP error message is handled by the IP software and *not* sent to the application program. However, if ICMP determines that a particular higher-level protocol or application program has caused a problem, it will inform the appropriate module. ICMP messages are of various types, some of the types include:

- Echo request and reply (Packet InterNet Groper - PING)
- Destination unreachable
- Redirect (change a route)
- Time exceeded for a datagram
- Time stamping request/reply
- Overrun conditions
- Information (address) request/reply

So ICMP is used for basic error reporting (*not correction*) between two hosts at the inter-network level, and is an integral part of the Internet Protocol (IP) that handles error and control messages.

The intent of ICMP is to support some level of error reporting, not to address every error situation. In other words, ICMP does *not* relieve the upper-level protocols from the responsibility of ensuring error free data.

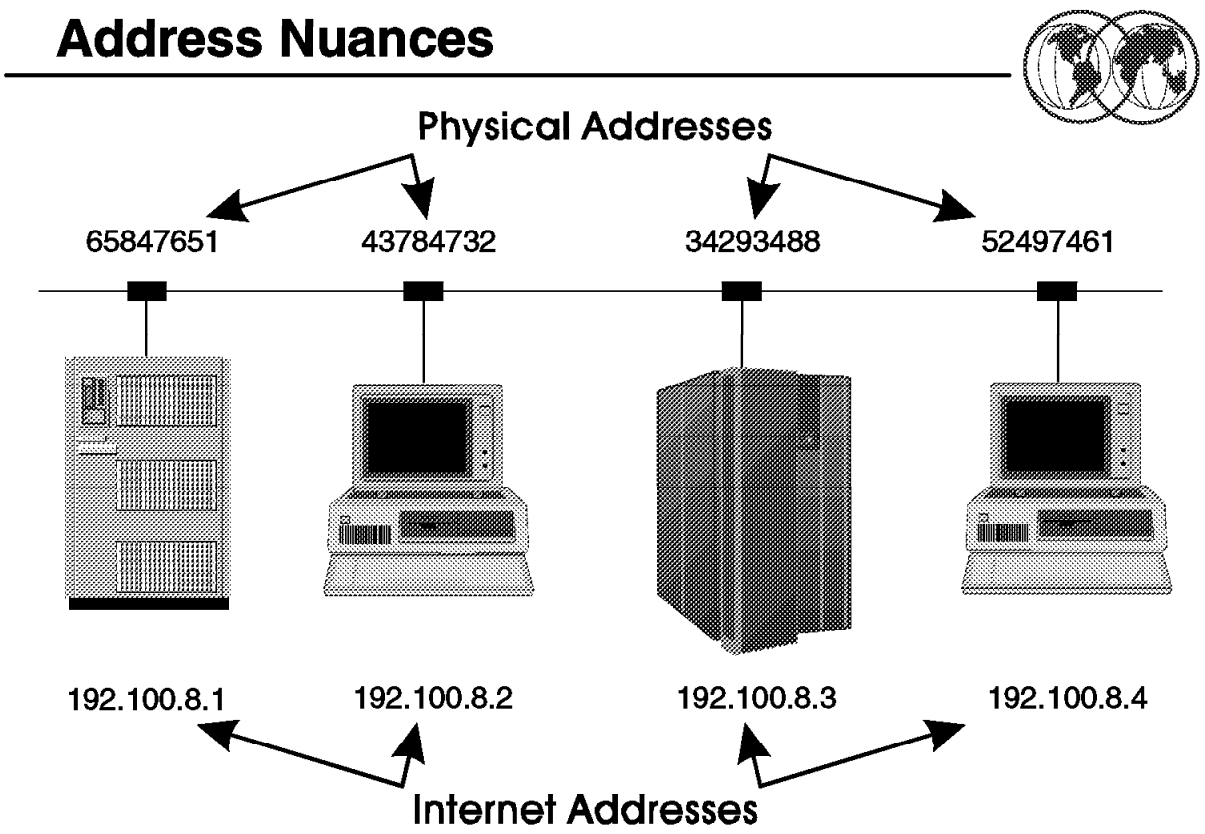


Figure 42. Address nuances

1.21.5 Address nuances

The problem:

- Hosts in an IP network are known by their 32-bit internet address.
- Networks need to know the physical network address of hosts to allow communication between hosts on the network.

Therefore, some mechanism is required to translate a 32-bit internet host address to the hardware address for that host on its physical network.

Remember that each host in an internet must have a unique 32-bit IP address. Also remember that an internet may consist of many different kinds of physical networks. Each one of these physical networks may have a different method of physical hardware addressing for the hosts on that network. In general, the hardware address for a host on a network is much different from the 32-bit software address used by IP for software running on that host.

We need to address the problem of mapping internet addresses (software addresses) to the actual physical hardware addresses.

One option is to set the host IP address equal to the physical network address. However, this does not work because in some cases:

- Physical address length exceed the host internet address
- Physical address can change. (for example, Ethernet)

The solution could be either:

- Simple algorithms to translate software IP addresses to hardware addresses
- Address Resolution Protocol (ARP)
- A mapping table
- Reverse Address Resolution Protocol (RARP)
- Proxy ARP

We will discuss ARP, RARP, and ProxyARP in greater detail.

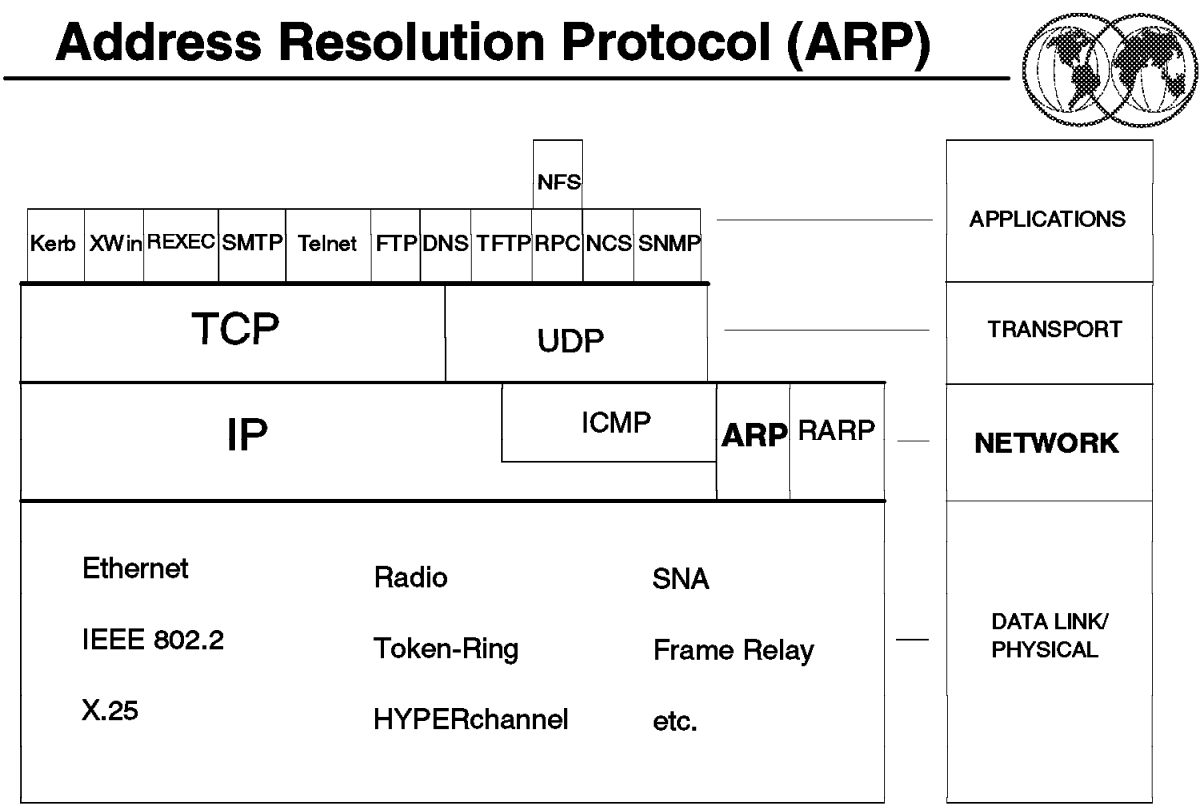


Figure 43. Address Resolution Protocol (ARP)

1.21.6 Address Resolution Protocol (ARP)

The Address Resolution Protocol, ARP, allows a host to find the physical address of a target host *on the same physical network* given only the target host's IP address.

- The basic notion behind ARP is:
  - The sending host broadcasts a packet containing the target internet address to all the stations on the local network.
  - All stations on the local network receive the packet.
  - The correct station will send back a reply identifying its local network address.

Suppose a host (source) wants to send a packet to another host (destination) on the same physical network. The source host will turn the packet over to IP. The IP routing function will attempt to match the destination IP address to a physical address.

Nodes supporting the ARP function will maintain a table that equates IP addresses to physical addresses. Packets containing an IP address that is not in the table (also called the ARP "cache") will result in an ARP broadcast being sent across the physical (local) network in an attempt to locate the address. A host recognizing its IP address in the ARP packet will respond allowing the originator of the broadcast to identify the physical address of the destination. (This is similar to the way NETBIOS works.)

Some refinements to ARP include:

- "Caching" of address resolution information.

- Utilization by all stations of information contained in the special broadcast packet.

An ARP packet contains information such as:

- Type of hardware and protocol being used (for example, Ethernet).
- Source and target IP address (32 bits).
- Source and target hardware (physical) address (48 bits).

An important advantage of the ARP process is that it is dynamic—no tables are maintained by systems programmers.

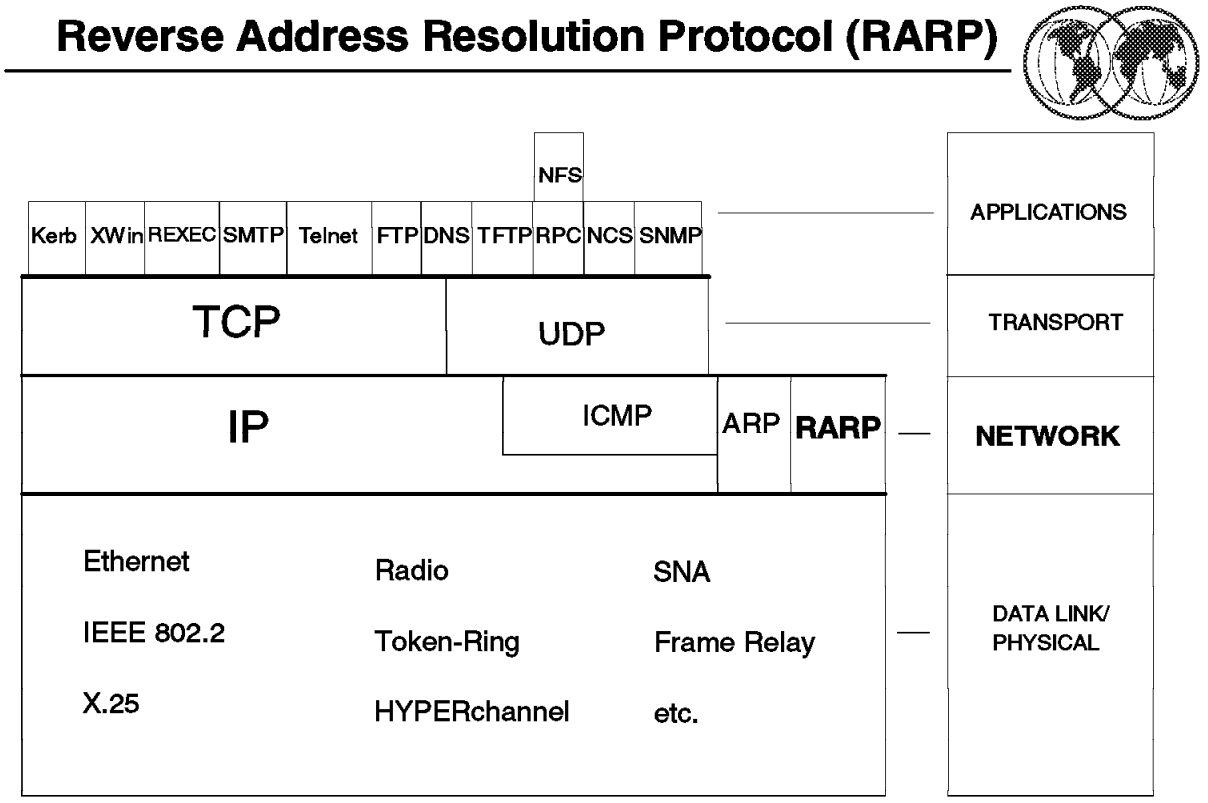


Figure 44. Reverse Address Resolution Protocol (RARP)

1.21.7 Reverse Address Resolution Protocol (RARP)

There are some kinds of workstations referred to as “diskless workstations.” They have no access to secondary storage, and therefore cannot store an IP address for use to communicate with other hosts on an internet. A method is required that allows a workstation that knows its hardware address to acquire an internet address.

The basic notion behind RARP is:

- How to determine the internet address given a physical address.

The RARP process works as follows:

- The host diskless workstation (source) sends a broadcast request to special RARP server(s). The host workstation probably does *not* know the IP address of the RARP server, so it sends its broadcast request at the physical address level to all machines on its local network.

  The broadcast request includes the physical address of the host workstation.
- Only machines authorized to supply the RARP services process the request and send a reply. The RARP server will have a pre-defined mapping table to support this function. It receives the broadcast request and searches its tables for the IP address that matches the physical address received in the broadcast request. The RARP server then responds to the requestor with the required internet address.
- This requires the specific function of an RARP server in the network.
- The contents of the RARP packet are very similar to the ARP packet.

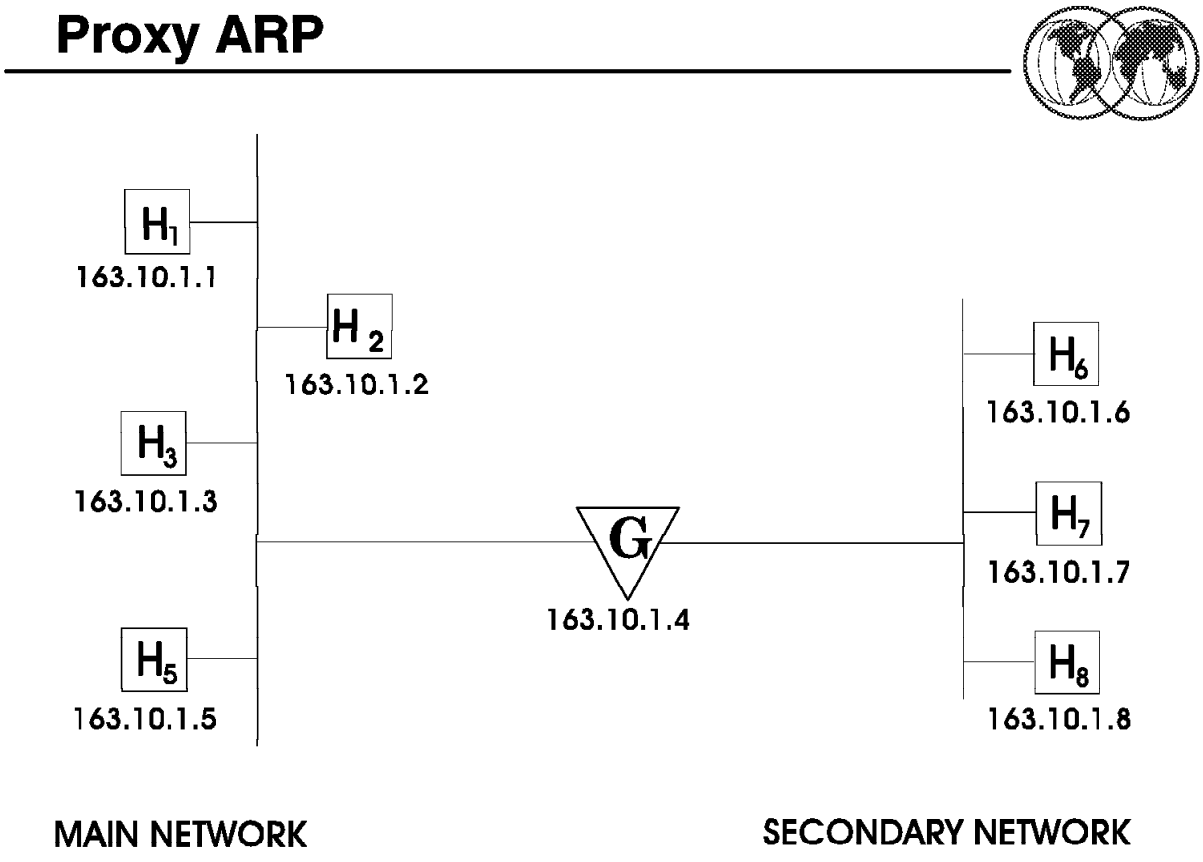


Figure 45. Proxy ARP

1.21.8 Proxy ARP

The goal of Proxy ARP is to allow *multiple* physical networks to share a *single* internet <network address> and thereby simplify internet routing and administration. In the visual above, note that there are two separate physical networks joined by gateway 163.10.1.4. Also note that the <network address> for all the hosts on both networks is 163.10.1!

Proxy ARP works on a network that uses ARP to do the internet-to-physical address translation. In this example, the MAIN NETWORK is the original network and the SECONDARY NETWORK was added at a later date. The gateway "G" connecting the two networks uses ARP to maintain the illusion that only one network exists. To make the illusion work, G keeps the location of hosts completely hidden, allowing all other machines on the network to communicate as if directly connected.

- H1 (163.10.1.1) has an IP datagram destined for H8 (163.10.1.8).
- H1 invokes ARP to map H8′s IP address to a physical address.
- Because gateway "G" runs proxy ARP software, it captures the broadcast ARP and responds with its (the gateway′s) local network address, hence it assumes the proxy role.
- H1 receives the ARP response and installs the mapping in its ARP cache.
- H1 sends the datagram that is destined for H8, but actually sends it to gateway "G."
- When G receives a datagram, is searches a special routing table to determine how to route the datagram.
- G forwards the datagram destined for H8 over the secondary network.

To allow hosts on the secondary network to reach hosts on the main network, G performs the proxy ARP service on that network as well. Proxy ARP is good only in limited configurations.

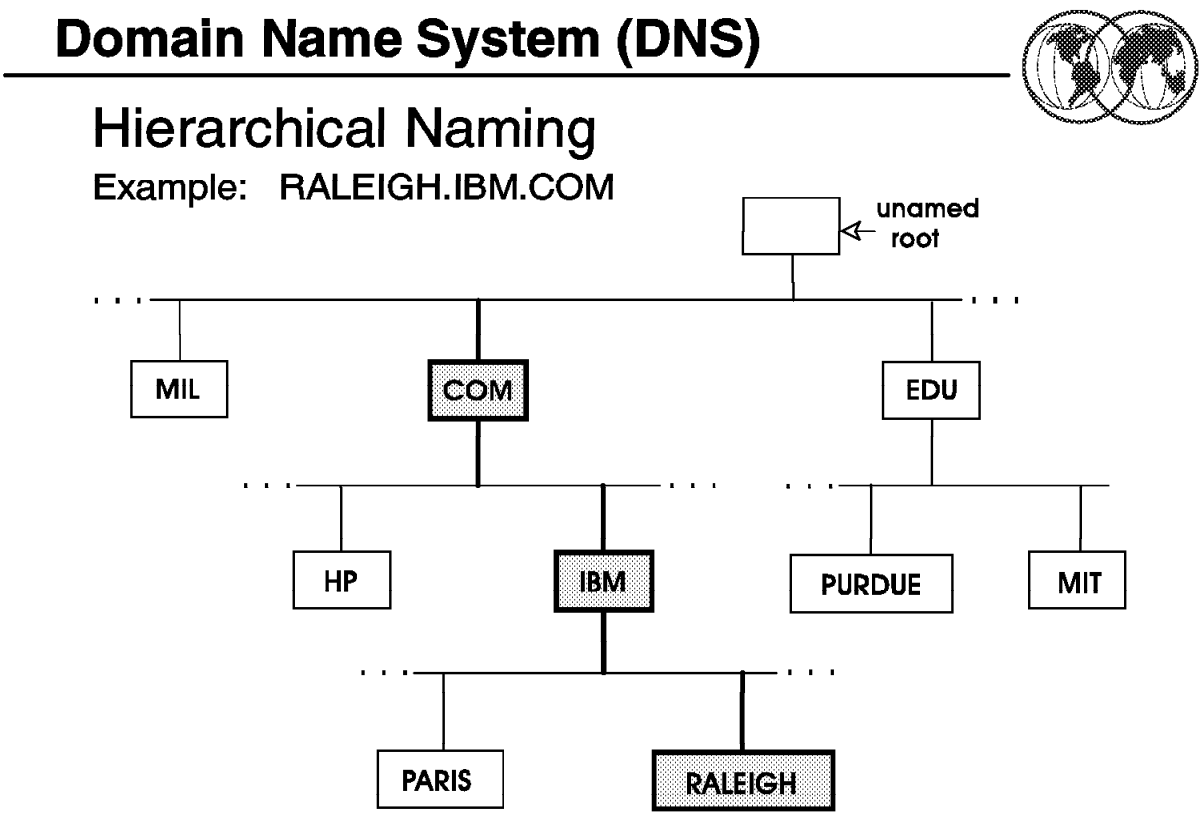


Figure 46. Domain Name System

1.22 Domain Name System

It is difficult for a user to remember the 32-bit IP address for each host that the user may require communications with. The use of symbolic high-level machine names makes it much easier for users to remember host names and establish communications with applications running on many different TCP/IP hosts.

As the acceptance and use of TCP/IP grew, the use of numeric IP addresses was replaced by the use of symbolic names. Initially, host names-to-address mappings were maintained by the Network Information Center (NIC) in a single file which was fetched by all hosts using the File Transfer Program (FTP). Due to rapid growth in the number of hosts on the Connected Internet, this mechanism became too slow and inefficient and was replaced by the Domain Name System.

- Domain names, like IP addresses, are concatenated. For example nm1.nm2.nm3.
- The Domain Name System is used to structure network names in a hierarchical fashion. This means that the Internet is divided into hierarchical domains.
- In this system, and shown in this visual, the highest level domain (below the unnamed root) is a major network type. For example, as seen here COM is the highest level domain for commercial organizations and EDU is the highest level domain for educational institutions.

  The top-level domain namespace is partitioned by the Internet authority. The current top-level names are:

Domain Name	Meaning	
edu	Educational institutions	
com	Commercial organizations	
gov	Government institutions	
mil	Military groups	
net	Major network support centers	
org	Organizations other than those above	
arpa	Temporary ARPANET domain (obsolete)	*
int	Internation organizations	
country code	International standard two-letter identifier	

- The next level represents major networks within a network type. In this example, IBM is a major network within the COM or commercial organizations domain.
- At the next level, the network domains are divided into individual networks, such as network RALEIGH shown here.
- The network name is a concatenation of various domain names.
- In this example, the name VM9.RALEIGH.IBM.COM represents a host, VM9 on the network RALEIGH, which is a domain on the IBM network, which is connected to the top level network COM.

An important feature of the Domain Name System to remember is that the Internet authority is responsible for partitioning and naming only the top level domain namespace. Authority for names and partitioning at the lower levels is the responsibility of the designated agent at that level For example, IBM might choose to partition its namespace based on *site name* and to delegate to each site responsibility for maintaining names within its partition (an administrative group at RALEIGH would be responsible for names and further subdivision within its location). The idea is to keep subdividing the namespace until each subdivision is small enough to be manageable. The topmost level (the Internet authority) divides the namespace and delegates authority for each division; it is not bothered by changes *within* one division.

For these reasons, the TCP/IP Domain Name System naming scheme allows delegation of authority for the hierarchical namespace *without regard to physical connections*.

Now we must have some way of mapping the names maintained by the Domain Name System to the actual IP addresses of the hosts. This function is performed by cooperative systems called *name servers*.

Name Servers

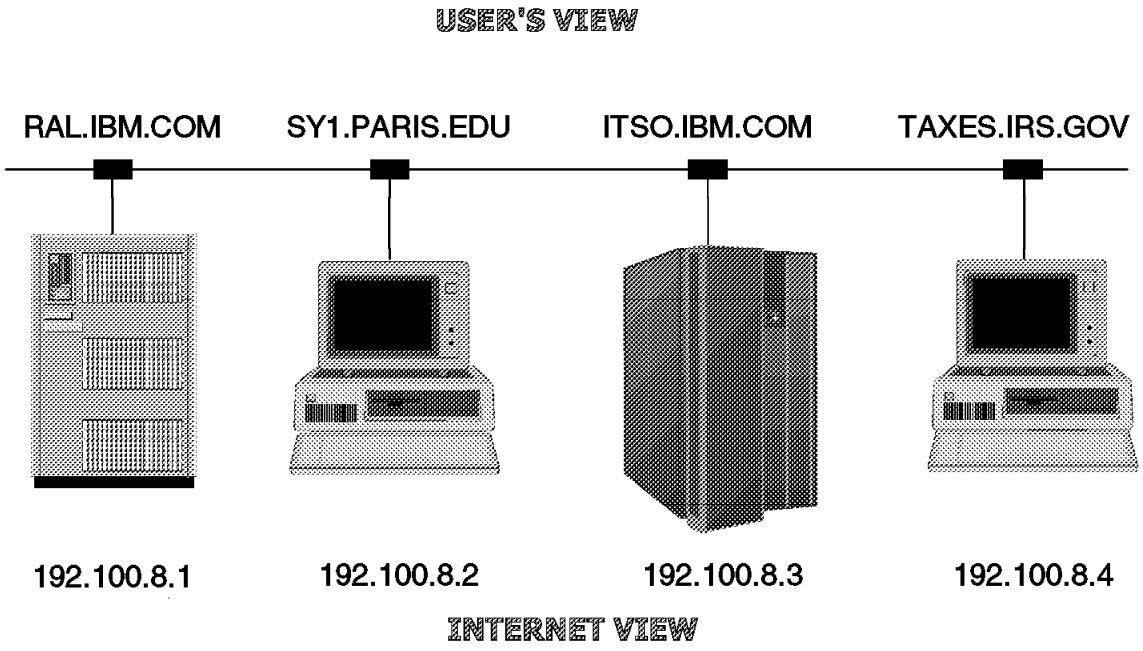


Figure 47. Name servers

1.22.1 Name servers

Before a packet can be sent through the network, the target host name identified by the user sending the packet must be mapped to a valid IP (numeric) address.

This mapping function may be done by using a table at the source host, or the services of a *name-server*.

A name server is a server program that supplies name-to-address translation (mapping) from Domain Names to IP addresses.

Name servers map (translate) user-friendly host Domain Names into network friendly IP addresses. Often name server software executes on a dedicated processor, and the machine itself is called the *name server*.

The client function (running on the host that needs a Domain Name translated into an IP address) is called a *name resolver* since it has the responsibility for converting the name to an IP address. If the client name resolver function cannot locate the name in its cached resident mapping table, it will send the request to a “name-server.”

- The IP address of the local name server must be known to the client.
- The client can request recursive or non-recursive services to be used in the event that the local name server does *not* have the target host name in its cache mapping table.

  - *Recursive* service means that the local name server will contact other name servers and report the results back to the client (IP address or a negative response).
  - *Non-recursive* service means that the local name server will return the IP addresses of *other* name servers back to the client; the client must then contact them directly.

- In the interest of efficiency, name servers cache resolved names to improve the overall performance of the name server system. Each entry has a separate time-to-live so that after the specified amount of time, the entry will be deleted and the name server must go back to the authoritative source to update its cache entry.

Another protocol called the “Domain Protocol” is used for communicating between the domains. Domain protocol messages can be sent using either TCP or UDP.

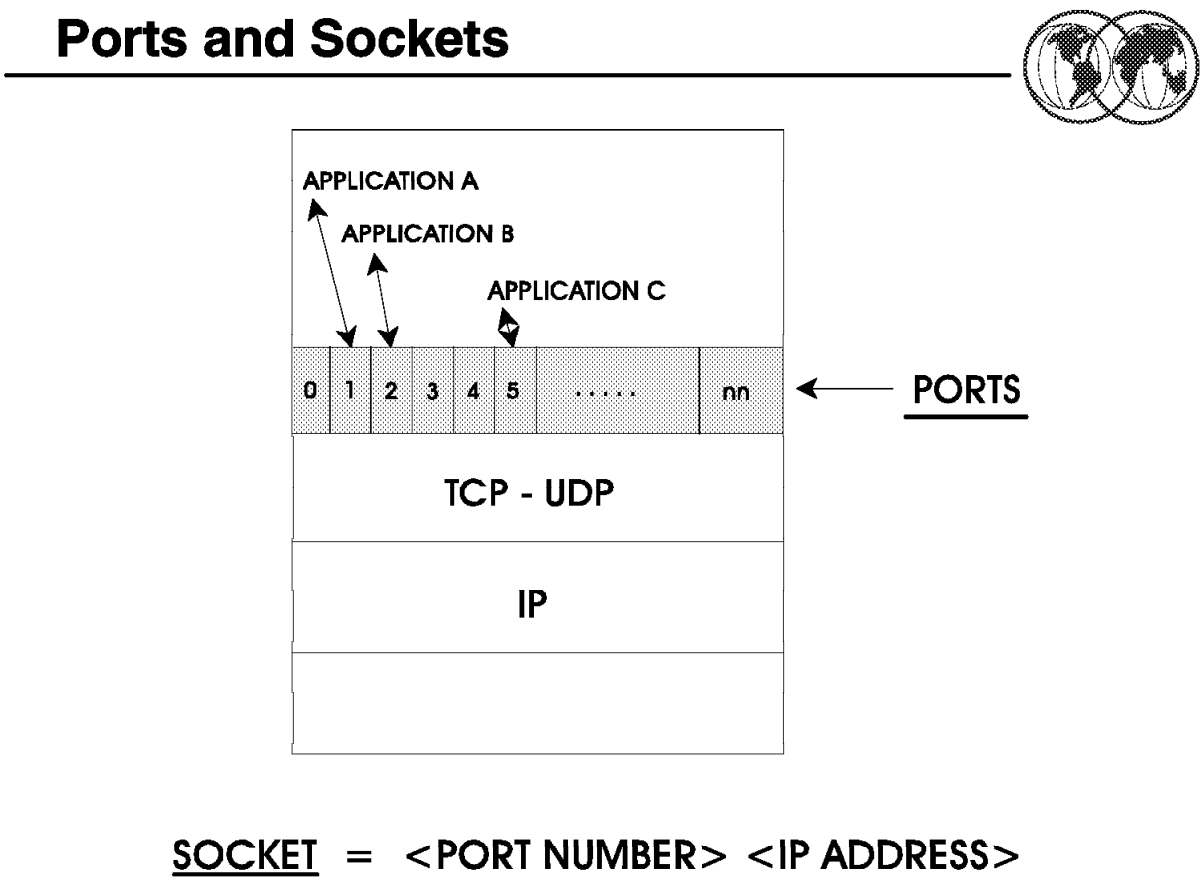


Figure 48. Ports and sockets

1.23 Ports and sockets

Each host has one or more “processes” (applications) that have a need to communicate with processes on other hosts (an example of a process is an application such as TELNET). In order to use the functions of TCP/IP, each process must identify itself to TCP by one or more *ports*.

A port or ports may be thought of as a logical connection point that TCP uses to distinguish among multiple processes within a given host computer. Therefore ports can be thought of as process addresses used by TCP. *Sockets* are a concatenation of a port number and an internet address.

So a process (application) utilizing TCP is associated with a particular TCP *port*. The port number is selected when a process begins communications with TCP (OPEN primitive). This port provides access to and from TCP and any process or application in the same host or a different host. A port is a 16-bit number used by the host-to-host protocol (TCP) to identify to which higher-level protocol or application program (process) it must deliver incoming messages.

- A TCP port corresponds to the Service Access Point (SAP)
- Some commonly used processes or applications (for example, TELNET, FTP, and so forth) will always use the same port number. These ports used by these processes are termed “well-known” ports.

The range of well-known port numbers is between 0 and 255. User applications should use port numbers above this range. Different user-written applications can request an *available port* from TCP/IP, avoiding any confusion that might occur if two applications request the same port number.

Ports are used by both TCP and UDP.

Processes are known to the network by a concatenation of their port address and the IP address. This total identifier is also called a “socket” or a “socket address.”

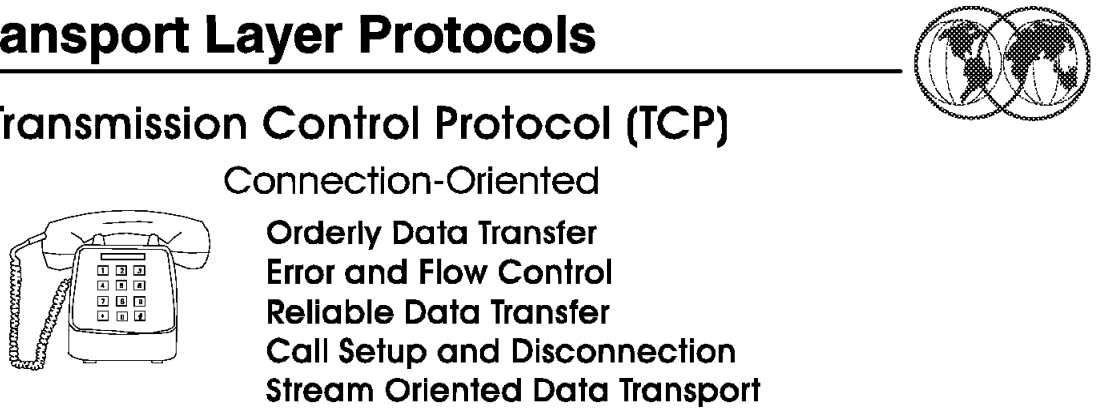


Figure 49. TCP/IP transport layer protocols

1.24 Transport layer protocols

The primary duty of the *transport layer* is to provide communication from one application program to another. Such communication is often called *end-to-end*. The transport layer may regulate the flow of information. It may also provide reliable transport, ensuring that data arrives without error and in sequence. To do so, it arranges to have the receiving side send back acknowledgements, and it retransmits lost packets. The transport software divides the stream of data being transmitted into small pieces (called packets in the ISO terminology and segments by TCP/IP) and passes each packet, along with a destination address, to the next layer (IP) for transmission.

A general purpose computer can have multiple application programs accessing an internet at one time. The transport layer must accept data from several user programs (processes) and send it to the next lower layer. To do so, it adds additional information to each packet, including codes (*port numbers*) that identify which application program should receive it, as well as a checksum. The receiving machine uses the checksum to verify that the packet arrived intact, and uses the destination code (port number) to identify the application program to which it should be delivered.

The two protocols at the TCP/IP transport layer are TCP and UDP.

- TCP is a connection-oriented protocol. Before transferring data, a connection is established with the destination node. When data transfer is ended, the connection is terminated. This process is analogous to making a telephone call, where connections are established, with the partner before there is an exchange of information. Hence, TCP is a connection-oriented protocol.

- UDP is a connection-less protocol. No connection is established prior to the transfer of data. The data is sent into the network and is assumed to reach the destination. This process is similar to the mailing of a letter, where the letter reaches the destination, without a connection being established between the origin and destination.

UDP provides a procedure for application programs to send data to other programs with a minimum of protocol intervention. UDP data is sent in the form of packets or “datagrams.” UDP provides an unreliable mode of communication between the source and destination hosts. UDP does not offer a guarantee of datagram delivery or duplication protection.

Applications that require reliable delivery or streams of data should use TCP.

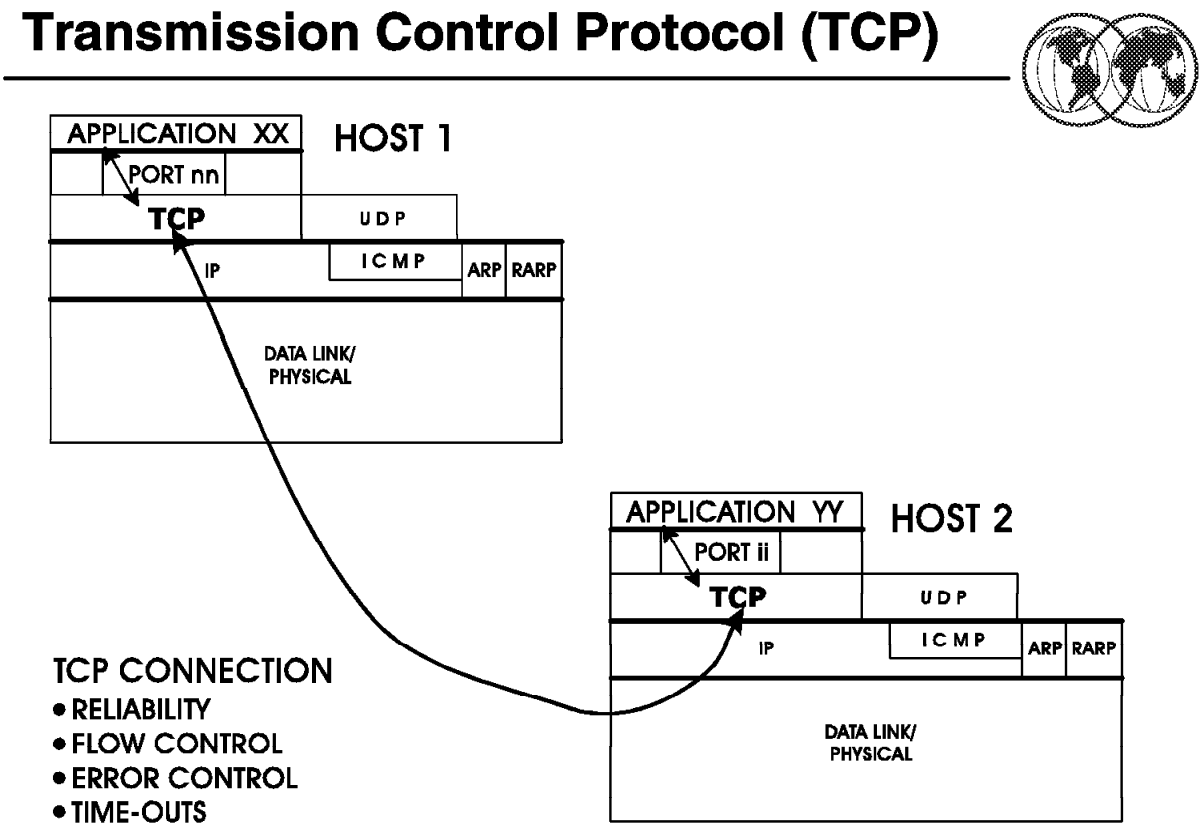


Figure 50. TCP

1.24.1 Transmission Control Protocol (TCP)

TCP is a transport layer protocol that provides connection type services, including reliability, flow control, and error recovery between pairs of processes (applications). It does *not* assume reliability from the lower-level protocols (such as IP), so TCP must guarantee this itself.

A connection represents a logical circuit between two *sockets*. This logical connection must be established prior to data flow.

TCP takes care of segmentation of the data as required by the lower-level network functions. It can be characterized by the following facilities it provides for the applications (processes) using it:

- Stream data transfer

  From the application's point of view, TCP transfers a *contiguous stream of bytes* through the internet. The application does not have to bother with chopping the data into basic blocks or datagrams. TCP does this by grouping the bytes into TCP *segments* which are passed to IP for transmission to the destination. Also, TCP decides how to segment the data and it may forward the data at its own convenience.

- Reliability

  TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. As the data is transmitted in blocks (TCP segments), only the sequence number of the first data byte in the segment is sent to the destination host.

The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.

- Flow control

  The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive beyond the last received TCP segment, without causing overrun and overflow in its internal buffers. This is sent in the ACK in the form of the highest sequence number it can receive without problems. This mechanism is also referred to as a *window*-mechanism.

- Window size

  Window size is set when the connection is established, but will change dynamically during data transfer.

- Error recovery

  Error recovery is done using byte sequencing; the receiver sends an acknowledgement indicating the last byte number successfully received; the sender will then retransmit any bytes (in the form of packets) not yet acknowledged.

  Once an error-packet is received, the receiver will stop counting even if the following packets are received error free.

- Time outs

  TCP has the ability to adjust time-out values related to data transmission by examining the time it takes for an acknowledgement to be returned for packets sent (round-trip delay).

  How TCP reacts to time-out and error conditions will be determined by how each host (user) implements error recovery.

- Multiplexing

  Multiplexing is achieved through the use of *ports*.

- Logical connections

  The reliability and flow-control mechanisms described above require that TCPs initialize and maintain certain status information for each “data stream.” The combination of this status, including sockets, sequence numbers, and window-sizes is called a *logical connection* (or virtual circuit). Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes (applications).

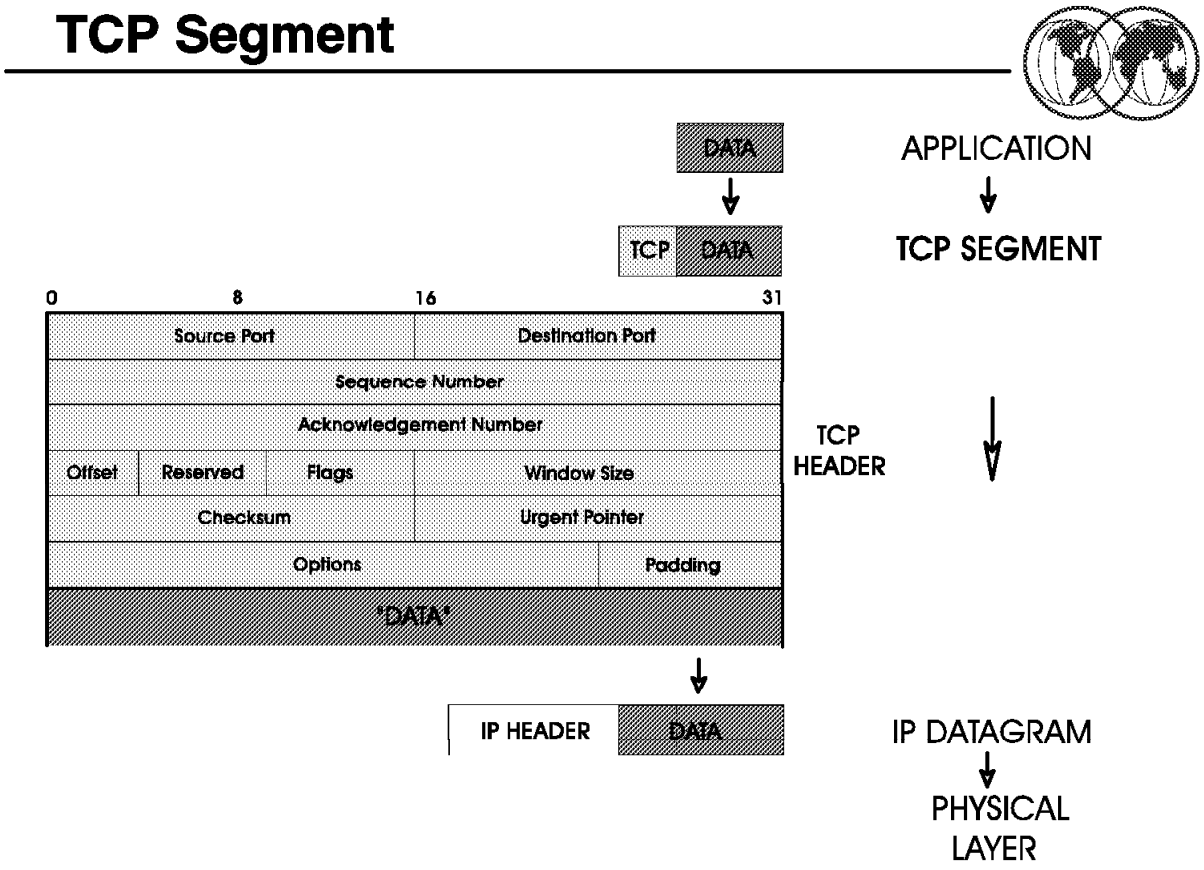


Figure 51. TCP segment

1.24.1.1 TCP segment

TCP takes the data from the higher level application and adds its header information. The resulting protocol data unit (PDU) is called a TCP *segment*.

Remember that TCP will block bytes received from an application into segments and include the sequence number of the first data byte included in the segment as part of the TCP header (sequence number field).

- TCP segments contain a TCP header followed by unaltered application data that has been blocked into segments as required by the network.

  Some of the fields contained in the TCP header include:

  - 16-bit source and destination *port* numbers.
  - Byte sequence numbers - the sequence number of the first data byte in this segment.
  - Acknowledgement number - if the ACK control bit is set (in the Flags field), this field contains the value of the next sequence number that the receiver is expecting to receive.
  - Window value - used in ACK segments, it specifies the number of data bytes (beginning with the one indicated in the acknowledgment number field) which the receiver is willing to accept.
  - Checksum for the TCP header.
  - Flags - urgent pointer flag, acknowledgement flag, and so forth.
  - User data - data received from the application.

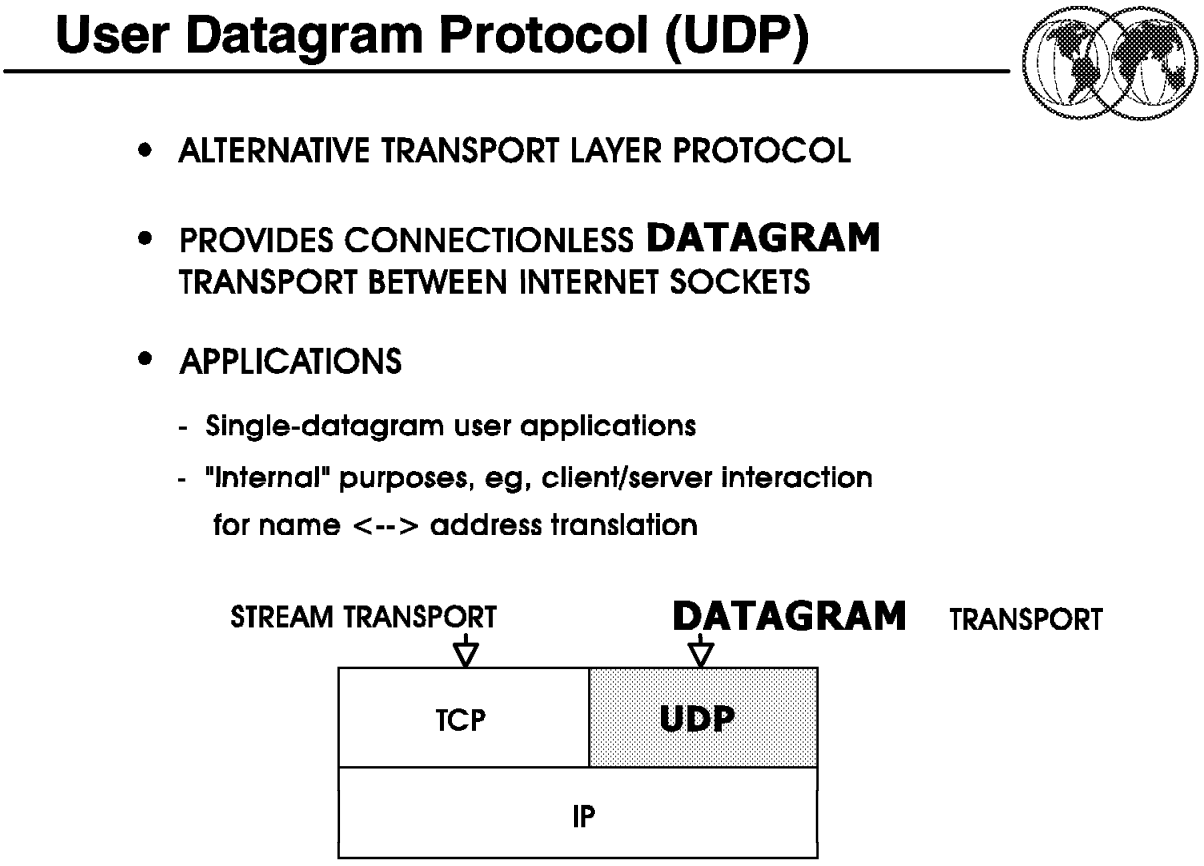


Figure 52. User Datagram Protocol (UDP)

1.24.2 User Datagram Protocol (UDP)

UDP or User Datagram Protocol provides the interface between *ports* and IP for *connectionless* data transfer (that is, using datagrams).

- UDP is an alternative protocol to TCP at the transport layer.
- UDP provides no error recovery, flow control, or reliability.
- UDP will receive the information from a port (process or application) and envelope it in a UDP datagram.
- The UDP datagram contains the application data along with the source and target port addresses, a length value and a header checksum value.
- There are some TCP/IP applications that use the UDP protocol (for example, Simple Network Management Protocol (SNMP), Trivial File Transfer Protocol (TFTP), and so forth).

UDP uses the underlying Internet Protocol to transport a message from one machine to another, and provides the same unreliable, connectionless datagram delivery semantics as IP. It does *not* use acknowledgements to make sure messages arrive, it does *not* order incoming messages, and it does *not* provide feedback to control the rate at which information flow between the machines. Thus, UDP messages can be lost, duplicated, or arrive out of order. Furthermore, packets can arrive faster than the recipient can process them. In summary:

The User Datagram Protocol (UDP) provides unreliable connectionless delivery service using IP to transport messages between machines. It adds the ability to distinguish among multiple destinations (PORTS) within a given host computer.

The ability to distinguish among multiple applications uses the same *port* concept that was discussed on previous visuals. Each UDP message contains two 16-bit port numbers (send and receive ports) in the header. As described earlier, some of these ports may be “well known” and permanently reserved for specific applications (for example, TFTP uses port 69).

An application program that uses UDP accepts full responsibility for handling the problem of reliability, including message loss, duplication delay, out-of-order delivery, and loss of connectivity.

UDP is a small protocol in the sense that it does not add much to the semantics of IP. It merely provides application programs with the ability to communicate using the unreliable, connectionless, packet delivery service.

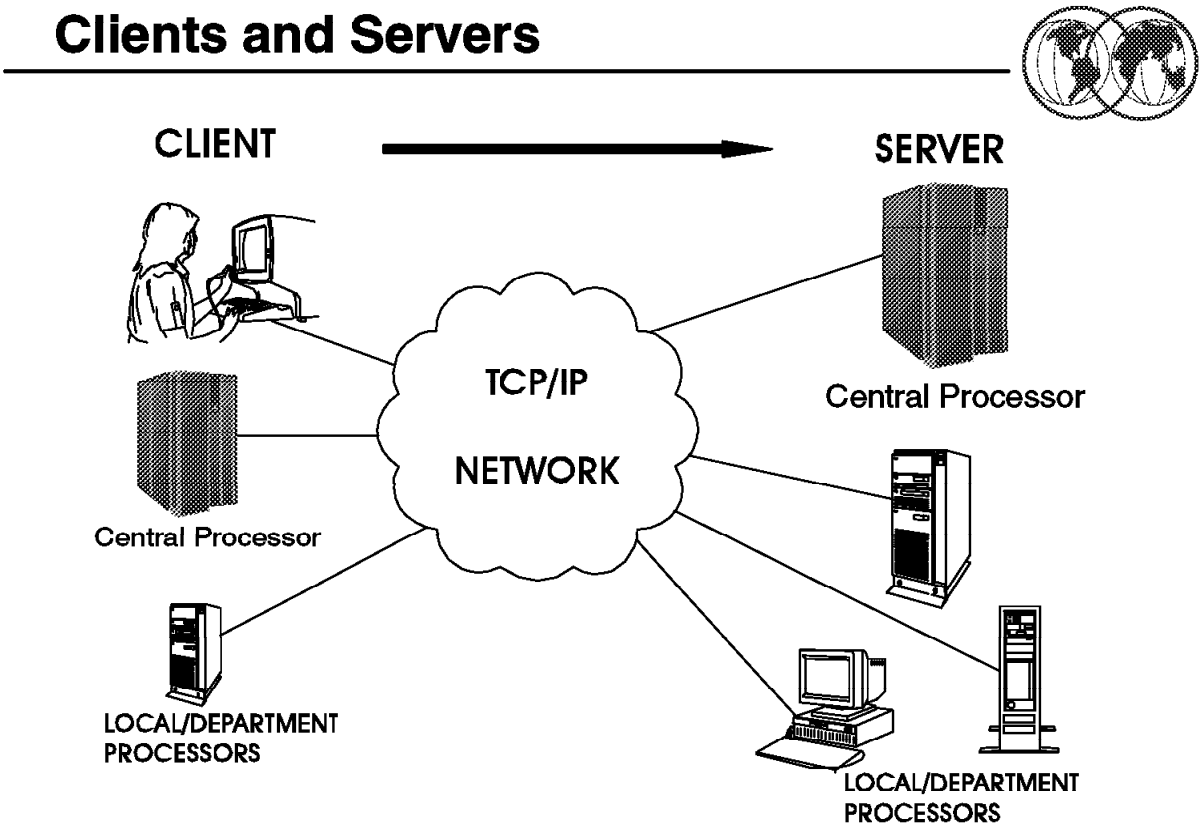


Figure 53. TCP/IP clients and servers

1.25 Clients and servers

TCP is a peer-to-peer, connection-oriented protocol. There is no master/slave relation between hosts. Applications running on those hosts, however, usually use a *client-server* model for communications.

- A *server* is a process or application that provides a service to other users (applications) in the network.
- A *client* is a “requestor” of a service.

A client-server type of application consists of both a server and a client part, which can run on the same or on different systems.

- Like more and more of todays network implementations, TCP/IP uses a “client-server” model for communications between applications.
- Users usually invoke the client part of the application to create a request. The request is then sent by the client code to the server. TCP/IP is the transport vehicle. The server then performs the requested function and sends replies to the client. A server can usually deal with multiple requests (multiple clients) at the same time.
- Some servers (that is, the more popular TCP/IP applications such as FTP, TELNET, an so forth) provide their services through “well-known” ports.
- Server functions require considerably more code than do client functions; TCP/IP products providing support for the various applications may provide only the client function, only the server function, or both.

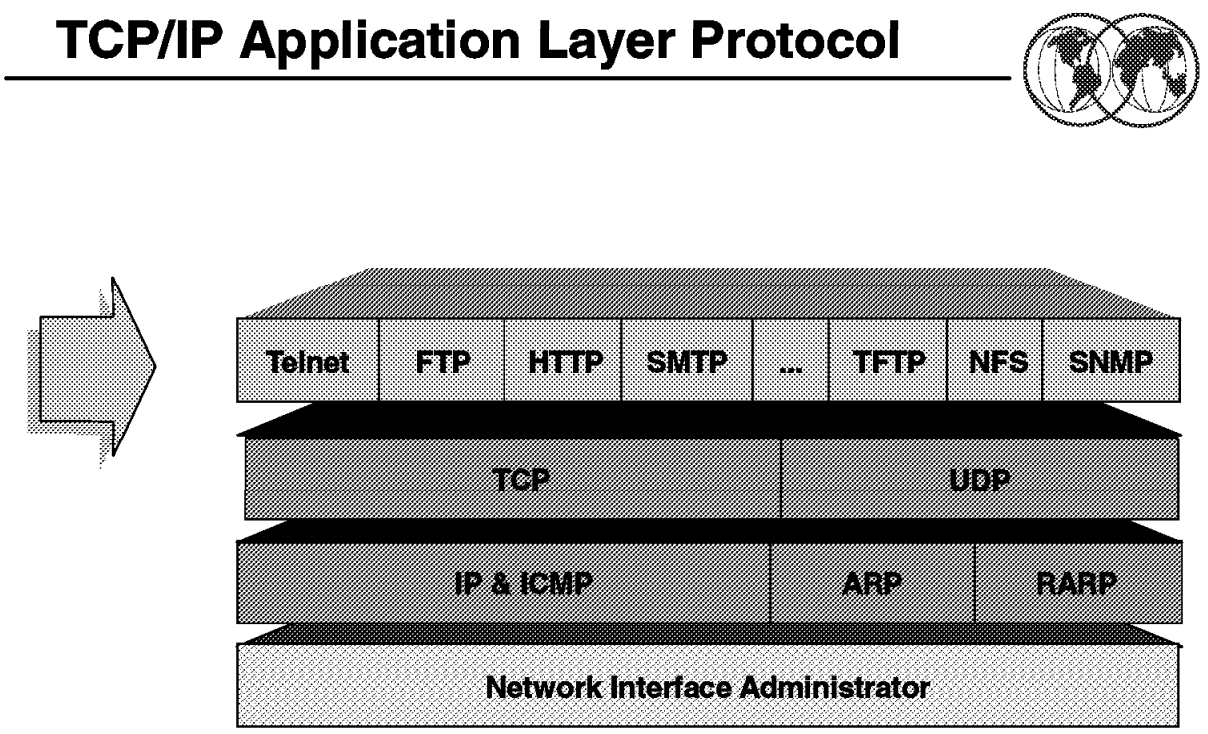


Figure 54. TCP/IP Application Layer Protocol

1.26 TCP/IP Application Layer Protocol

In addition to the Network and Transport layer protocols, the TCP/IP protocol suite includes a number of higher level protocols, called “applications or processes.” Functionally, they provide services such as file transfer, messaging, and virtual terminal support. These applications interface to TCP (or UDP) and are all optional. Inclusion of support for specific applications varies widely from vendor to vendor.

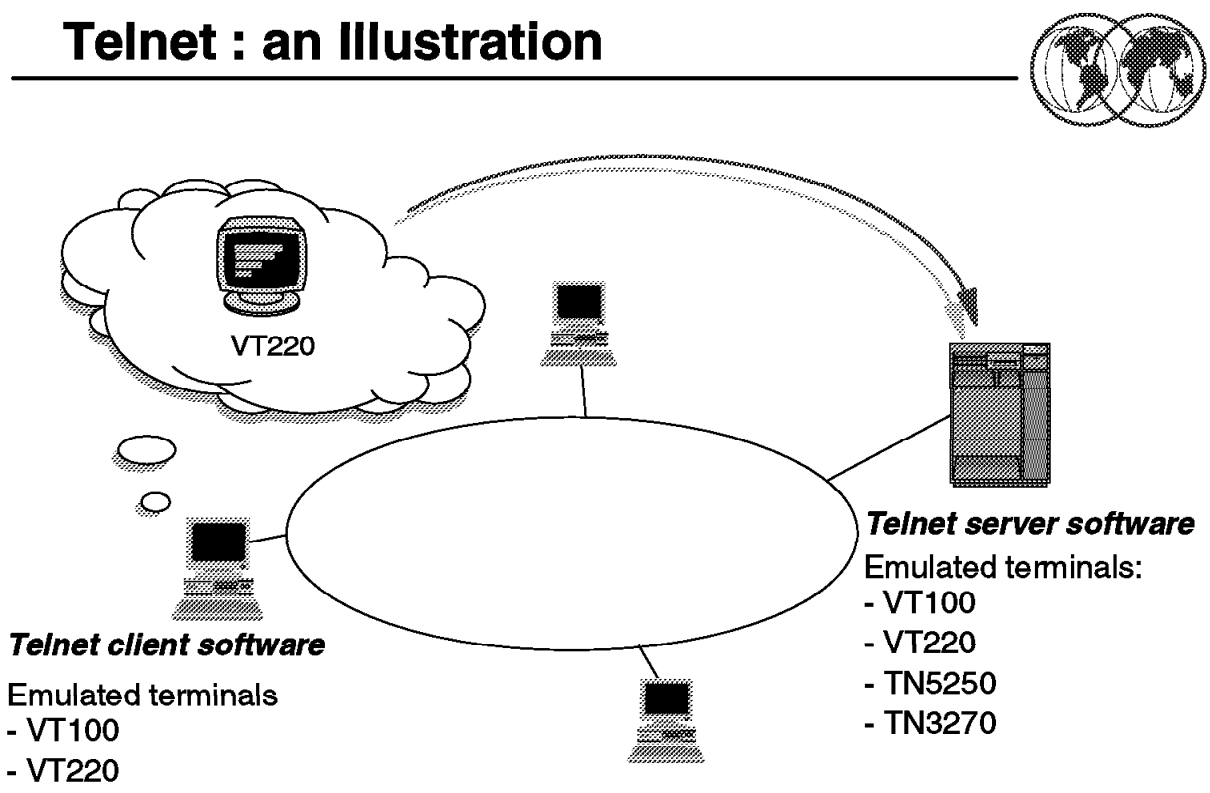


Figure 55. TELNET

1.26.1 TELNET: an illustration

TELNET provides a standardized interface to allow a process (application) at one host (using the TELNET client function) to access a process (application) at another host (using the TELNET server function) as if the client was a local terminal directly connected to the remote host.

TELNET performs this function by offering three basic services:

- It defines a *network virtual terminal* to provide a standard interface to remote systems.
- It allows the client and server to negotiate options for terminal characteristics.
- TELNET treats both ends of the connection the same way—the client side is not required to be a user′s terminal, it could be a program instead. Either end can negotiate options.

TELNET does *not* provide graphics capabilities.

TELNET is a simple remote terminal protocol. It allows a user at one site to establish a TCP connection to a login server at another site, and then it passes keystrokes from the user′s terminal directly to the remote machine as if they had been typed at a terminal on the remote machine. TELNET also carries output from the remote machine back to the user′s terminal. The service is called *transparent* because it gives the appearance that the user′s terminal attaches directly to the remote machine.

Typically, the TELNET server function handles multiple, concurrent connections from TELNET clients anywhere in the IP network.

To make TELNET interoperate between as many systems as possible, it must accommodate the details of many different kinds of computers and operating systems. For example, some systems require lines of text to be terminated by the ASCII *carriage control* character. Others require the ASCII *line-feed* character. In addition, most systems provide a way for a user logged in to an ordinary terminal to interrupt a running program. However, the keystroke used to generate the interrupt varies from system to system.

Therefore, TELNET defines how data and command sequences are sent across the internet. The definition is know as the *network virtual terminal* (NVT). An NVT is an imaginary device, providing the necessary basic structure of a standard terminal. Each host maps its own terminal characteristics to this NVT, and assumes that every other host will do the same.

The TELNET protocol also supports the principle of negotiated options. This is because many hosts may wish to provide additional services beyond those available with the NVT. Various options may be negotiated between the client and server during the connection process. The server and client use a set of conventions to establish the operational characteristics of their specific TELNET connection via the “DO, DON′T, WILL, WON′T” mechanism:

- WILL - will you agree to let me use option X
- WON′T - I will not use option X
- DO - I do agree to let you use option X
- DON′T - I do not agree to let you use option X

Examples of some of these options are:

- Support for specific terminal types (for example, 3278-2 type)
- Window sizes
- Translate table selection
- Full-screen formatting
- Support for echoing

Simple Mail Transfer Protocol

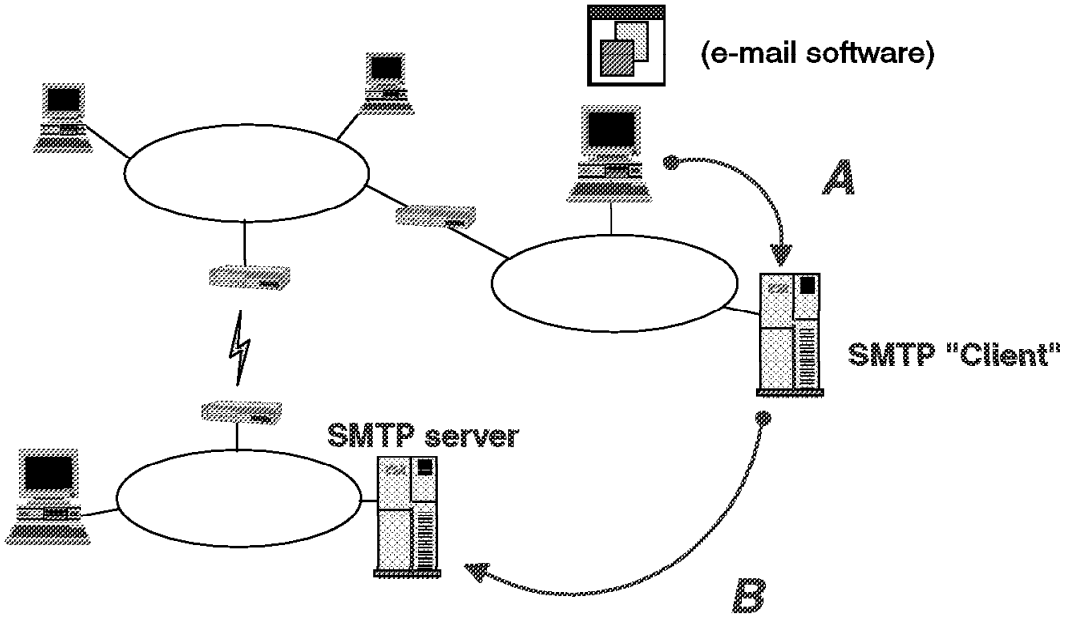


Figure 56. Simple Mail Transfer Protocol

1.26.2 Simple Mail Transfer Protocol (SMTP)

SMTP is an electronic mail protocol that provides support for message and note exchanges between users on the same or different hosts.

Two main standards define SMTP:

1. A standard on the format of the mail messages which deals with the layout and contents of the mail envelope (RFC 822).
2. A standard for the exchange of mail between two computers which specifies the protocol used to send mail to another SMTP (RFC 821).

Simple Mail Transfer Protocol is probably the most widely used TCP/IP application. It provides message and note exchange between TCP/IP hosts but has no support for document translation. Mail delivery differs from other uses of internet networks that we have discussed (TELNET). In other network protocols, they send packets directly to destinations, using timeout and retransmission for individual segments if no acknowledgement returns. In the case of electronic mail, however, the system must provide for instances when the remote machine or the network connections have failed. A sender does not want to wait for the remote machine to become available before continuing work, nor does the user want to have the transfer abort merely because communication with the remote machine becomes temporarily unavailable.

To handle delayed delivery, SMTP uses a technique known as *spooling*. When a user “sends” a mail message, the system places a copy in the spool area along with identification of the sender, recipient,

destination machine, and time of deposit. The system then initiates the transfer to the remote machine as a background activity, allowing the sender to proceed with other activities.

The background mail transfer process becomes a client. It maps the destination machine name to an IP address and attempts to form a TCP connection to the mail server on the destination machine. If it succeeds, the transfer process passes a copy of the message to the remote server, which stores the copy in the remote system′s spool area. Once the client and server agree that the copy has been accepted and stored, the client removes the local copy. If the transfer process cannot form a TCP connection, or if the connection fails, the transfer process records the time it tried delivery and terminates. The background transfer process sweeps through the spool area periodically, checking for undelivered mail. If the software finds that a mail message cannot be delivered after an extended time, it returns the mail message to the sender.

SMTP is *not* a store and forward mail system.

SMTP mail can be sent across other (local) mailing systems through an appropriately configured gateway; however, SMTP will only guarantee mail integrity from the sender to the mail gateway.

- PROFs Extended Mail is a software package that can be used to process mail with SMTP.
- SMTP is fully compatible with RSCS and NJE networks.

SMTP will use the name server functions for address mapping:

- Name format is <name @ host.net1.net2> where net1 and net2 are hierarchical network names.

Interfaces are provided to allow mail to flow into, out of, and through different types of networks without requiring any special action on the part of the user.

FTP: an Illustration

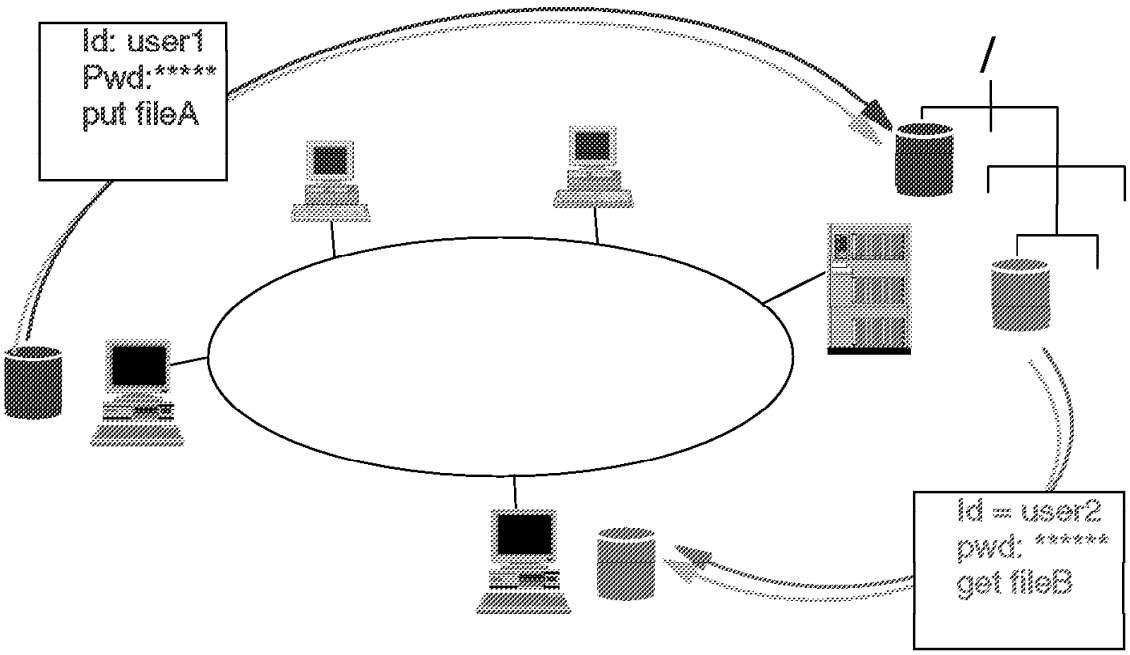


Figure 57. File Transfer Protocol

1.26.3 FTP: an illustration

File Transfer Protocol or FTP provides TCP/IP users with the ability to transfer files to and from remote hosts along with access to directories on remote hosts. In addition, FTP offers facilities beyond the file transfer function:

- Interactive Access - allows users to easily interact with remote FTP servers.
- Format Specification - allows the user to specify the type and format of stored data (text/binary, ASCII/EBCDIC, and so forth).
- Authentication Control - FTP requires clients to authorize themselves by sending a login name and password to the server before requesting file transfers.

Copying files from one machine to another is one of the most frequently used operations. Standard file transfer protocols existed for the ARPANET before TCP/IP became operational. These early versions of file transfer software evolved into the current standard known as the File Transfer Protocol.

TCP/IP file transfer is a disk-to-disk data transfer, as opposed to, for example, the VM SENDFILE command which is considered in the TCP/IP world as a mailing function (you send the data to someone′s mailbox/VM reader).

To support this essential file transfer function, a reliable end-to-end protocol such as TCP is used. FTP supports file transfer between client and server in either direction:

- The client may send a file to the server machine.

- The client may request a file from the server machine.

The user who initiates the FTP connection assumes the client function while the server function is performed by the remote host.

FTP includes functions that allow authorized users to display, define, and delete files and directories, and to transfer files to and from remote TCP/IP hosts.

Most FTP server implementations allow concurrent access by multiple clients using TCP to connect to the server. From an FTP user′s point of view, the link is connection-oriented. This means that it is necessary to have both hosts up and running TCP/IP to establish a file transfer.

FTP uses two connections between the client and server system to perform the file transfer functions:

- The first one is for login from the client to the server and follows the TELNET protocol. This connection is used for authentication of the client user, issuing commands to the FTP server, and so forth.
- The second one is for managing the data transfer. The actual file transfer occurs over this connection.

FTP can interface with RACF to provide access control.

X-Window: an Illustration

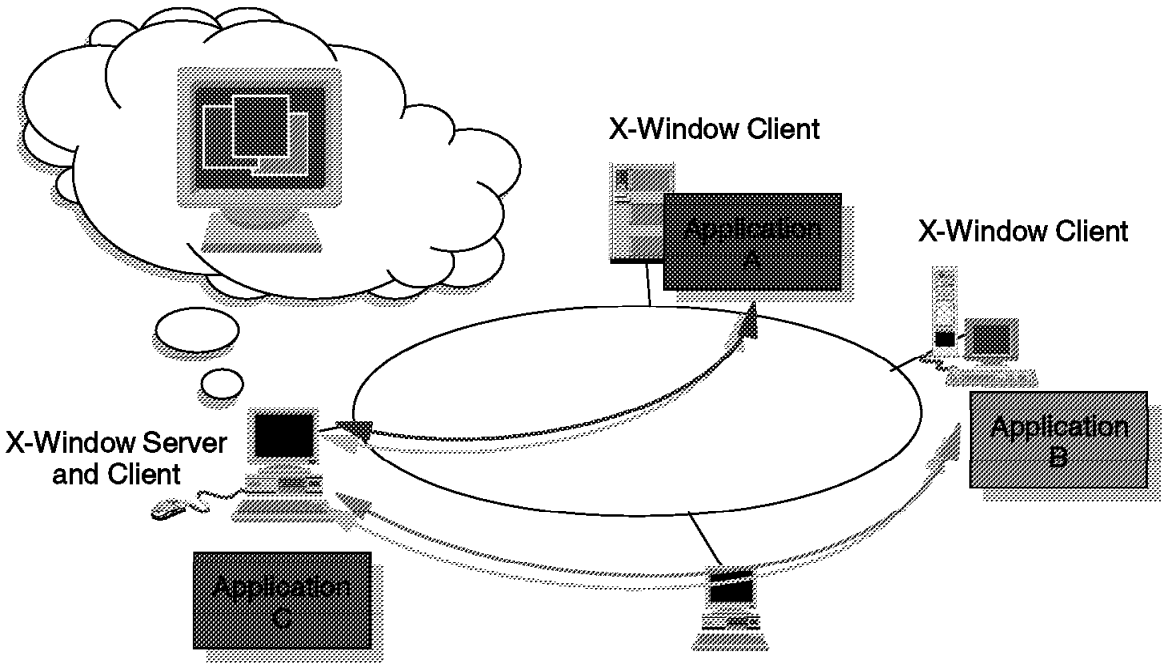


Figure 58. X-Windows

1.26.4 X-Windows: an illustration

X-Windows is a windowing system that allows a user to simultaneously display bitmapped screens from several local and/or remote applications (processes).

Basically, there are two parts communicating with each other:

- The application (called the *X-Client*) which gets input from the user, executes code, and sends output back to the user.
- The user's terminal (called the *X-Server*) which runs display-managing software that receives/sends data from/to the application.

Note that in the previous application protocols we have discussed (TELNET, SMPT, and so forth) the user is typically at the *client* end of the connection. With X-Windows, the user is at the *server* end of the connection. Each application that is displaying data in one of the windows is considered a client.

The X-Windows Application Programming Interface (API) (developed at MIT as part of the Athena Project) allows a TCP/IP user to run applications on several systems through a single workstation image. The X-Window system is one of the most widely used *Graphical User Interface (GUI)*, or bitmapped-window display systems.

The user is able to control all sessions from one screen, with applications either running in a window, or in separate virtual terminals but with an icon on the primary screen reminding the user of the existence of that application (similar to the OS/2 Presentation Manager).

The X-Window system provides the capability of managing both local and remote windows:

- Remote windows are established through TCP/IP.
- Local windows are established through the use of BSD *sockets*.

This function is most useful for applications that require computing resources that are not available at the user's workstation. The X-Windows API allows a client program to access a bit-mapped, high-resolution display connected to a X-Windows system protocol server program.

REXEC: an illustration

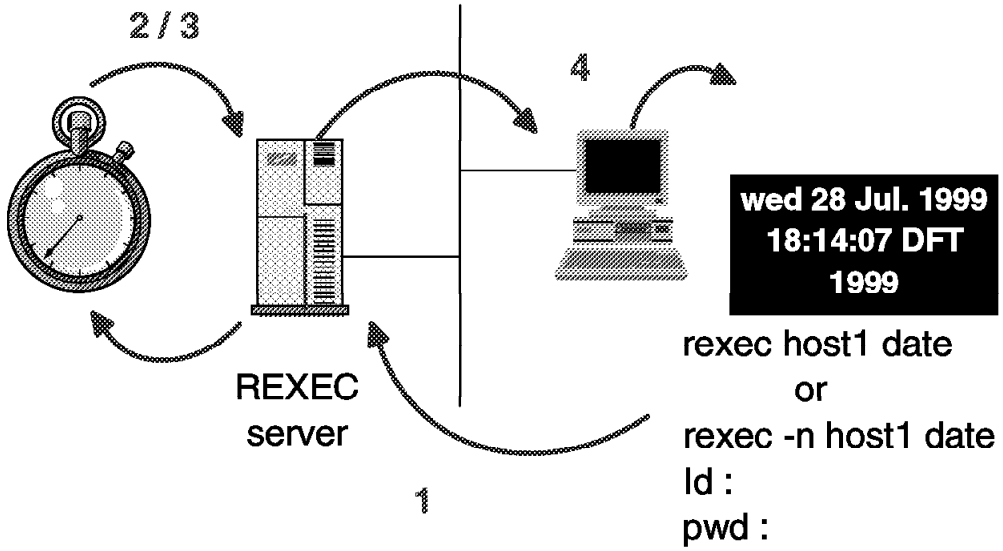


Figure 59. REXEC support

1.26.5 REXEC support

The Remote Execution Command (REXEC) is used in conjunction with the Remote Execution Command Daemon (REXECD). It allows a TCP/IP user to execute commands on a remote host and receive the results on the local system.

- The client function is performed by the REXEC process.
- The server function is performed by the REXECD process.

REXECD is a server (also known as a daemon). It handles commands issued by remote hosts and transfers orders to slave virtual machines for command execution. The daemon performs automatic login and user authentication when user ID and password are entered.

The REXEC command is used to define the user ID, password, host address, and process to be started on the remote host. Both server and client are linked over the TCP/IP network.

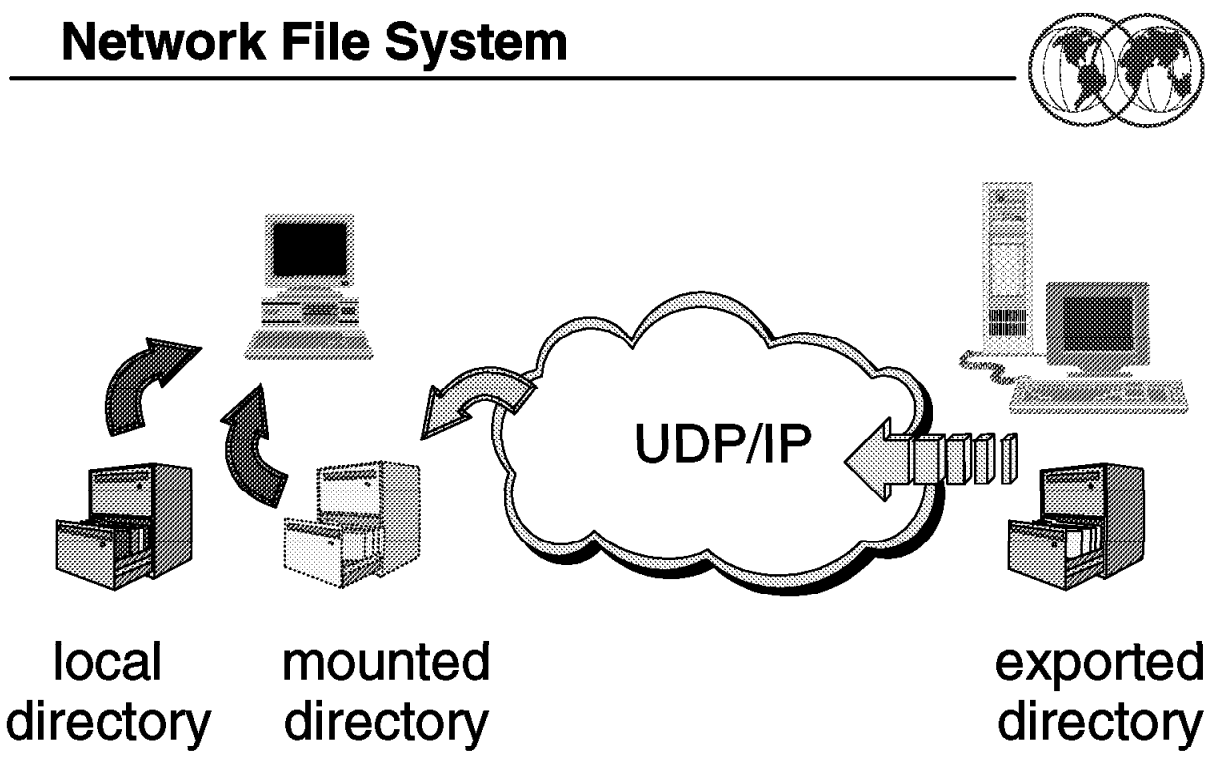


Figure 60. Network File System

1.26.6 Network File System

NFS was developed by SUN Microsystems. It allows host machines to share file systems across a network. NFS allows remote file systems to have the appearance of being local resources to users (protocol is transparent to the user).

The Network File System (NFS) provides on-line shared file access that is transparent and integrated. It allows TCP/IP sites to interconnect their computer file systems. From the user′s perspective, NFS is almost invisible. The user can execute an application program and use any file(s) (local and/or remote) for input or output. The file names do not show whether the files are local or remote.

When an application program executes, it calls the operating system to *open* a file, or to *store* and *retrieve* data in files. The file access mechanism accepts the request and automatically passes it to either the local file system software or to the NFS client, depending on whether the file is on a local disk or on a remote machine. When it receives a request, the NFS client software use the NFS protocol to contact the appropriate server on a remote machine and perform the requested operation. When the remote server replies, the client software returns the results to the application program.

NFS was developed to be machine, operating system, and transport protocol independent; this was accomplished by using the RPC interface.

NFS implements two protocols: Mount protocol and NFS protocol:

- Mount protocol is used to establish the connection and access the appropriate set of files (directory).

The options supported include password security, code conversion, remote disk addresses, and so forth.

- NFS protocol is used for the actual disk functions.

  Supported functions include searches, reading and writing, and directory maintenance.

Both Mount and NFS protocol are RPC applications (caller/server concept) and *transported by UDP*.

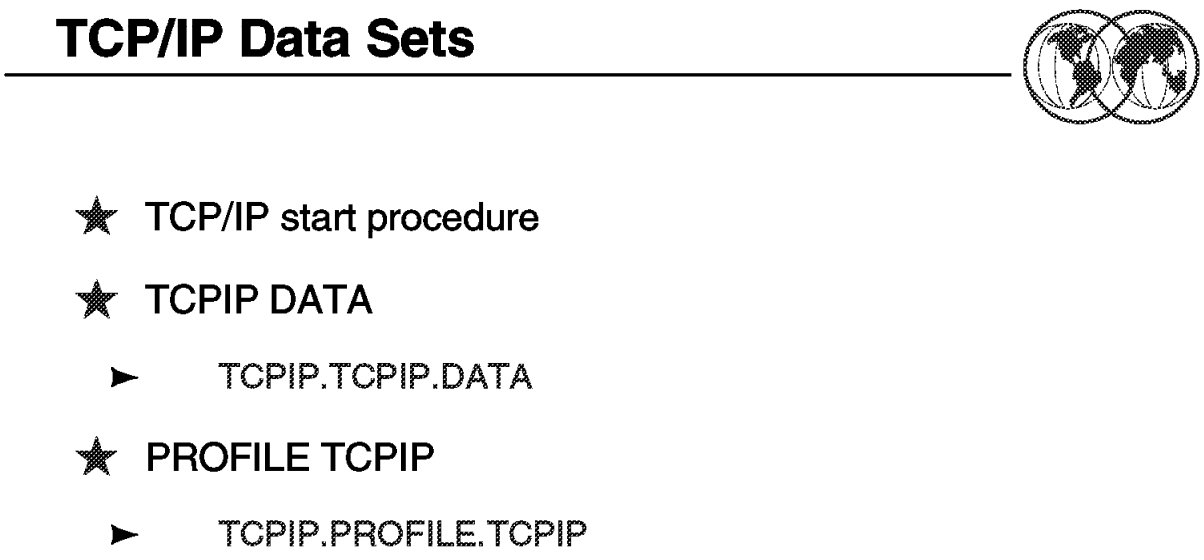


Figure 61. TCP/IP data sets

1.27 TCP/IP data sets

Required for TCP/IP start-up and initialization are the procedure from SYS1.PROCLIB and the parameter data sets:

- TCP/IP start procedure

  Copy the TCP/IP cataloged procedure in hlq.SEZAINST(TCPIPROC) to your system or recognized PROCLIB and modify it to suit your local conditions. Specify TCP/IP parameters and remove or change the DD statements as required. The jobname associated with the started task of the TCP/IP system address space must match the NAME parameter on the SUBFILESYSTYPE statement in the BPXPRMxx member of 'SYS1.PARMLIB' used to start OS/390 UNIX.

  When TCP/IP is started via the `S TCPIP` command, it reads its configuration parameters from the data set names specified on the PROFILE and SYSTCPD DD statements in the TCP/IP proc.

  These data sets are generally referred to as: *TCPDATA* respectively.

- PROFILE parameter data set
- TCPDATA parameter data set

In the following example of a basic TCPIP procedure, note the *PROFILE* and *SYSTCPD* DD statements which specify the locations of TCP/IP's parameter data sets.

```
//TCPIP    PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//*  SMP/E Distribution Name: EZAEB01G
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=7500K,TIME=1440
//*
//*     SYSPRINT contains run-time diagnostics from TCPIP. It may be
//*        a data set or SYSOUT.
//*     ALGPRINT contains run-time diagnostics from TCPIP's Autolog
//*        task. It should be SYSOUT.
//*     SYSERROR contains error messages from TCPIP that occurred
//*        while processing the PROFILE.
//*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//*
//*     TCPIP reads the parameters from a data set specified on
//*     the following PROFILE dd.
//*     See the chapter on "Configuring the TCPIP Address Space" in
//*     the Configuration Guide for more information.
//*     A sample of such a profile is included in member SAMOPROF in
//*     the SEZAINST data set.
//*
//PROFILE  DD DISP=SHR,
//            DSN=SYS1.TCPIP.PARMS(PROFILE)
//*
//*     SYSTCPD explicitly identifies which data set is to be
//*     used to obtain the parameters defined by TCPIP.DATA.
//*     The SYSTCPD DD statement should be placed in the TSO logon
//*     procedure or in the JCL of any client or server executed
//*     as a background task.  The data set can be any sequential
//*     data set or a member of a partitioned data set (PDS).
//*
//*     For more information please see "Understanding TCP/IP Data
//*     Set Names" in the Configuration Guide.
//*
//SYSTCPD  DISP=SHR,
//            DSN=SYS1.TCPIP.PARMS(TCPDATA)
```

The names of the data sets or members containing TCP/IP code and parameters are specified in the TCP/IP start procedure, SYS1.PROCLIB(TCPIP).

These members can be located in any data set, but for the purposes of this overview we will assume they are located in SYS1.TCPIP.PARMS.

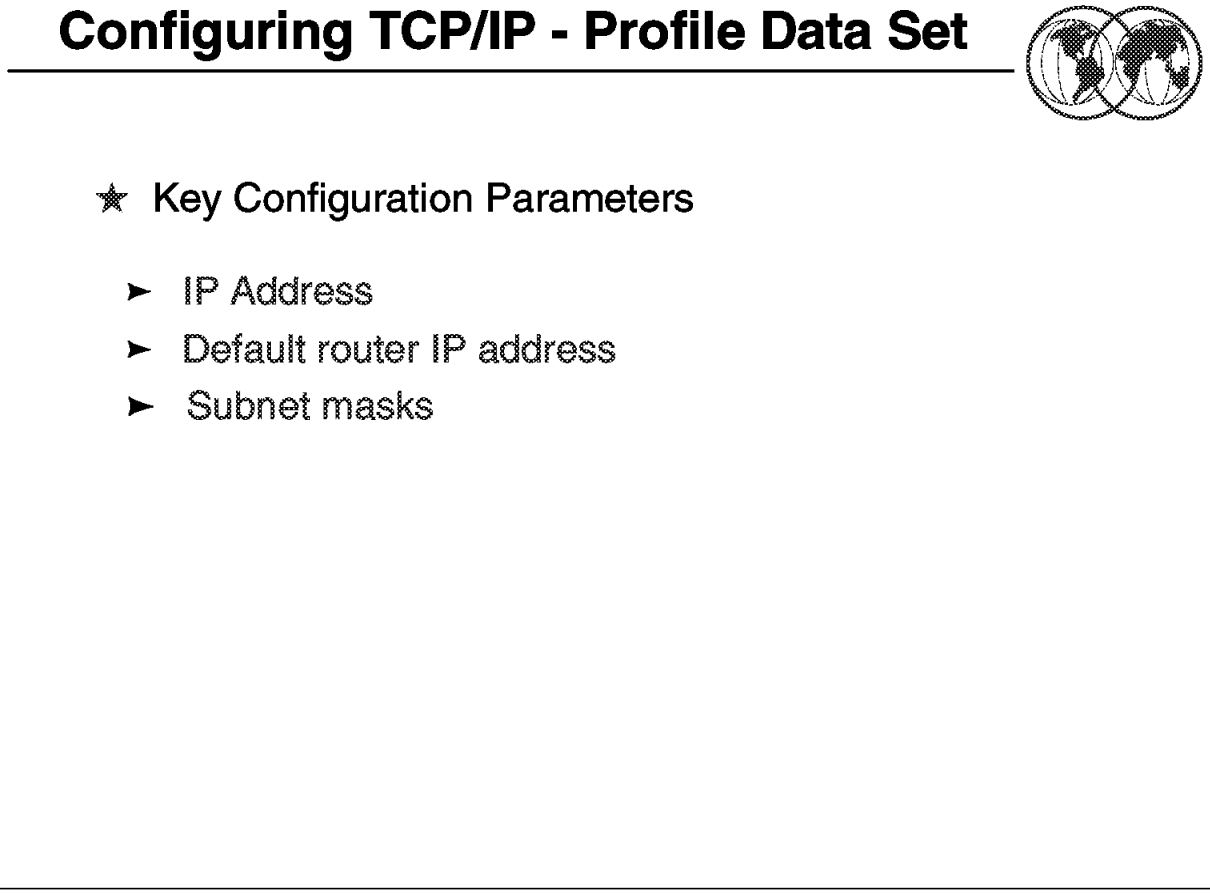


Figure 62. Configuring TCP/IP - Profile data set

1.27.1 Configuring TCP/IP - Profile data set

A sample configuration data set is provided in hlq.SEZAINST(SAMPPROF). This member is used to configure the TCP/IP address space.

A typical TCP/IP Profile includes many configuration options and parameter values too numerous to list here. However, for basic configuration purposes use the defaults in the supplied samples but pay particular attention to the following sections:

- IP address

  The IP addresses of the host on each link. For example, your CPU may be communicating with your TCP/IP network via two ports on an OSA card. Each port would have its own associated link ID and IP address. This example is defining the IP address 9.12.14.187 on Link OSAL2160 (defined by the DEVICE and LINK statements. The DEVICE and LINK parameters are used to define the type of the physical link the CPU is using to communicate with the TCP/IP network.

  There are a number of possible matching DEVICE and LINK parameter values depending on the hardware in use. Refer to *OS/390 V2R7.0 eNetwork CS IP Configuration*, SC31-8513, for a complete list of possible values and their meanings.

```
; Hardware definitions:
;
;
 DEVICE OSA2160   LCS           2160
 LINK   OSAL2160  IBMTR    0    OSA2160
;
 HOME
    9.12.14.187   OSAL2160
```

- Default router and subnet masks

  It is best to either acquire a comprehensive understanding of IP addressing and subnet masking or obtain the help of a competent network specialist before tailoring this section of the TCP/IP Profile data set.

```
;
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network First Hop Link Name Packet Size  Subnet Mask    Subnet Value
  9           =     OSAL2160     4096      0.255.255.0    0.12.2.0
;
; Indirect Routes - Routes that are reachable through routers on my
;                   network.
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
; 193.12.2 130.50.10.1   TR1       2000       0
; 10.5.6.4 193.5.2.10    ETH1      1500       HOST
;
;
; Default Route - All packets to an unknown destination are routed
;                 through this route.
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
  DEFAULTNET 9.12.2.75 OSAL2160  4096         0
;
```

- Start devices

  Lastly, enter the names of the LINKs that TCP/IP will start automatically during initialization.

```
; Start all defined DEVICEs
;
START OSA2160
```

*

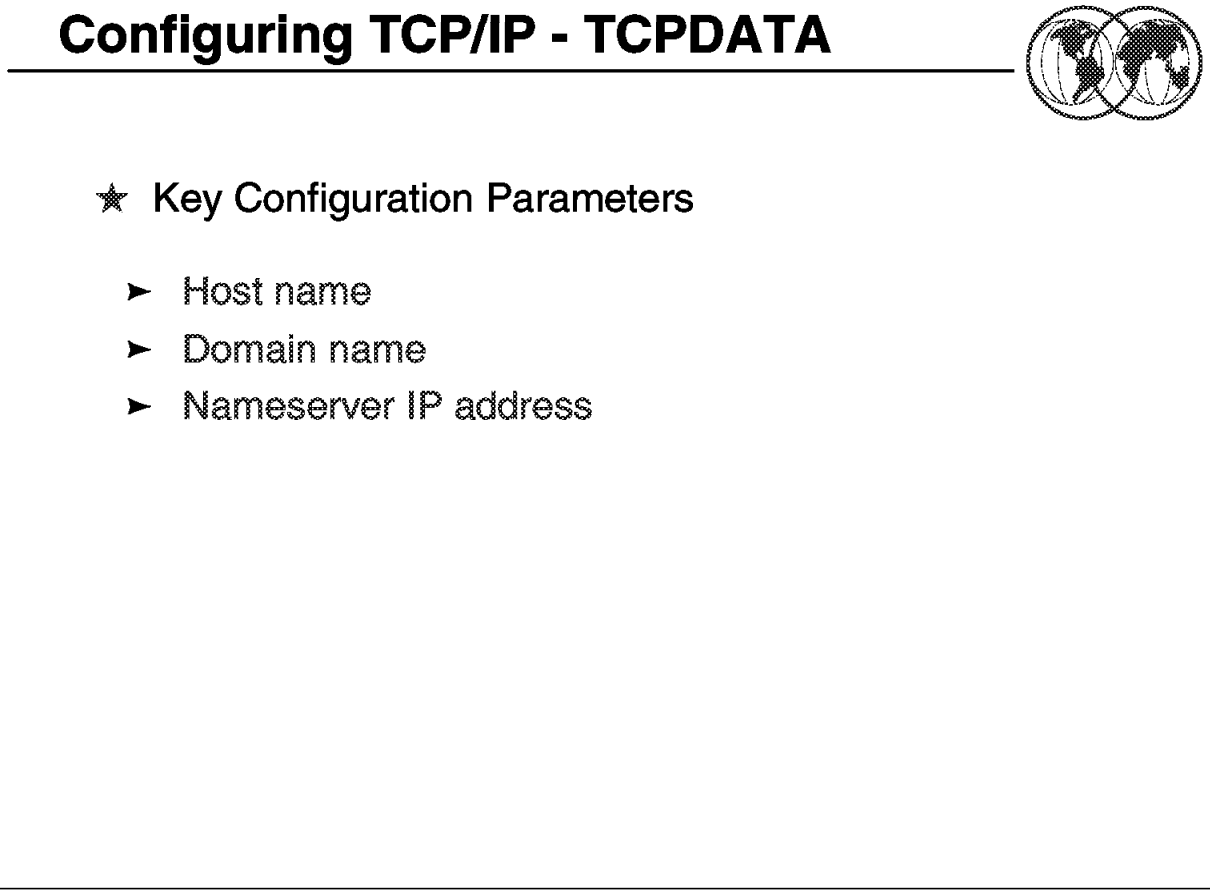


Figure 63. Configuring TCP/IP - TCPDATA

1.27.2 Configuring TCP/IP - TCPDATA

As with the TCP/IP Profile, TCPDATA contains many parameters that you should not change until you have developed a good understanding of TCP/IP and are in a position to know what you are doing.

It is advisable to accept most of the values in the sample member, but pay particular attention to the following sections:

- Host name

  For YOURMVSNAME, WTSC47 in this example, specify your OS/390 system name. This should be the same as your JES nodename. The colon after it is required. The word HOSTNAME should be entered followed by what you want to call your TCP/IP host. It is recommended you have both names the same, as in the example, to avoid confusion. *Case translation is not performed on the host name.*

```
; HOSTNAME specifies the TCP host name of this system.  If not
; specified, the default HOSTNAME will be the node name specified
; in the IEFSSNxx PARMLIB member.
;
; YOURMVSNAME:   HOSTNAME  YOURTCPNAME
WTSC47:   HOSTNAME  WTSC47
```

- Domain name

  The TCP/IP domain suffix that will be appended to the hostname. In this example, our hostname is WTSC47. The suffix ITSO.IBM.COM will be appended to WTSC47, giving a fully qualified domain name of WTSC47.ITSO.IBM.COM.

```
; DOMAINORIGIN specifies the domain origin that will be appended
; to host names passed to the resolver.  If a host name contains
; any dots, then the DOMAINORIGIN will not be appended to the
; host name.
;
DOMAINORIGIN  ITSO.IBM.COM
```

- Nameserver IP address

  Specify the IP address of your name server. If you are unsure of the correct value, speak to your network specialist. If you are not using a name server and this host will be resolving all domain names for itself, the NSINTERADDR statement(s) can be commented out.

```
; NSINTERADDR specifies the IP address of the name server.
; LOOPBACK (14.0.0.0) specifies your local name server.  If a name
; server will not be used, then do not code an NSINTERADDR statement.
; (Comment out the NSINTERADDR line below).  This will cause all names
; to be resolved via site table lookup.
;
NSINTERADDR  9.12.14.204
NSINTERADDR  9.12.14.7
```

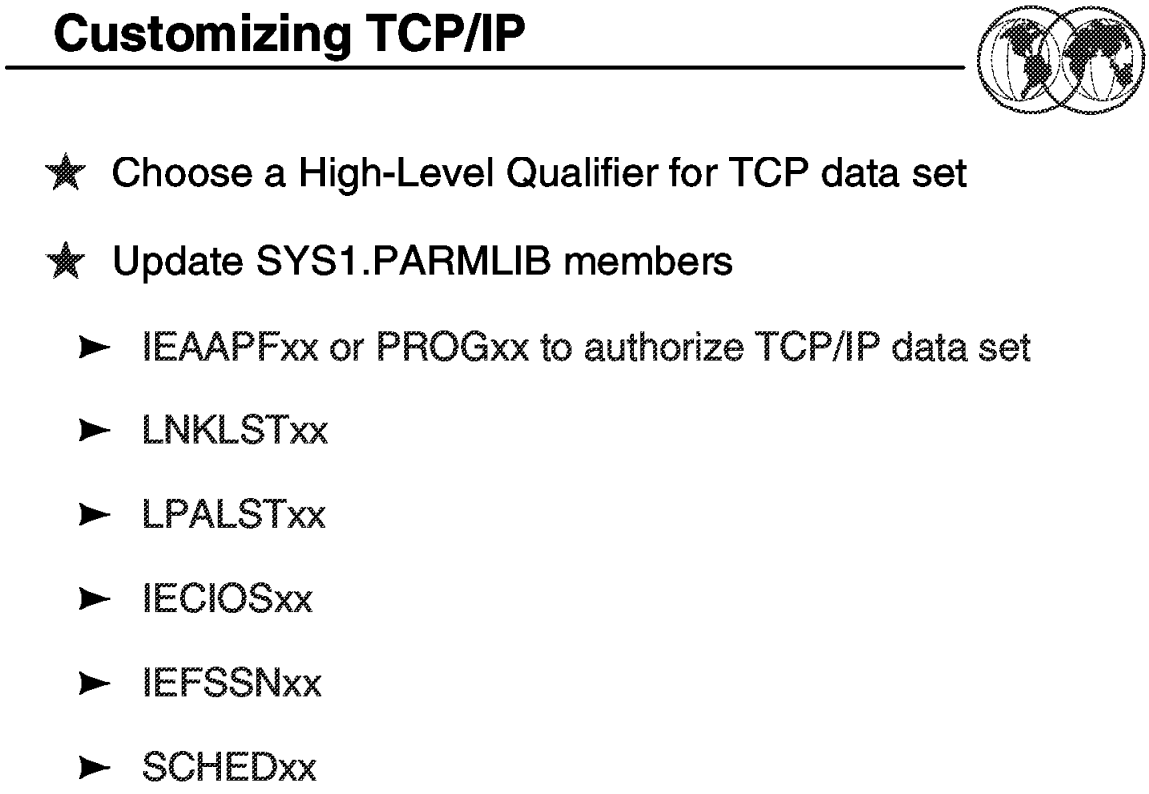


Figure 64. Customizing TCP/IP

1.27.3 Customizing TCP/IP

Steps for customizing TCP/IP are as follows:

- Choose a High-Level Qualifier (hlq).

  TCP/IP uses dynamic allocation with this hlq to get parameters from several TCP/IP data sets. The DATASETPREFIX statement in TCPIP.DATA can be used to override the default hlq. However, it is used as the last step in the search order for most configuration files.

- Update the following PARMLIB members:

IEAAPFxx/PROGxx Authorizes the following libraries:

- hlq.SEZATCP
- hlq.SEZADSIL
- hlq.SEZALINK
- hlq.SEZALNK2
- hlq.SEZALPA
- hlq.SEZAMIG

LNKLSTxx hlq.SEZALINK

LPALSTxx hlq.SEZALPA and hlq.SEZATCP

IECIOSxx Use to disable the MIH processing for the communication device used by TCP/IP

Set MIH TIME=00:00,DEV=(cuu-cuu)

IEFSSNxx Define subsystems to use restartable VMCF and TNF

- SUBSYS SUBNAME(TNF)
- SUBSYS SUBNAME(VMCF)

SCHEDxx Set PPT entries

- PPT PGMNAME(MVPTNF) KEY(0) NOCANCEL NOSWAP PRIV SYST
- PPT PGMNAME(MVPXVMCF) KEY(0) NOCANCEL NOSWAP PRIV SYST
- PPT PGMNAME(EZBTCPIP) KEY(6) NOCANCEL NOSWAP PRIV SYST

Note: The default program properties table (PPT) shipped with the OS/390 V2R7 base control program (BCP) includes entries for Communication Server (CS) for OS/390. Therefore, the SCHEDxx entries previously required are no longer required. If you have those entries in your own parmlib, remove them. If you used the IBM copy of parmlib, they have been removed for you. For details, refer to the *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

Customize TCP/IP Cont.

- ★ Update SYS1.PARMLIB members (cont.)
  - ► COMMNDxx
  - ► IFAPRDxx
  - ► BPXPRMxx

Figure 65. Customizing TCP/IP

1.27.4 Customizing TCP/IP

Update the following PARMLIB members:

COMMNDxx Add the following command to start the SSI:

```
COM='S EZAZSSI,P=sys_name'
```

Where sys_name is the SYSNAME in IEASYSxx or specified in IEASYMxx using the SYSDEF statement.

IFAPRDxx Enable TCP/IP base by adding this:

```
NAME(OS/390) ID(5647-A01)
```

BPXPRMxx Activate TCP/IP support for transport provider as follows:

```
FILESYTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
    DOMAINNUMBER(2)
    MAXSOCKETSR(10000)
    TYPE(INET)
```

Routing

- Routing is done based on the definitions in the routing table
- Each entry has two key definitions
  - Destination
  - Next hop

Figure 66. Routing

1.27.5 Routing

To support IP dynamics, NCP′s Network Definition Facility (NDF) builds a routing information table (RIT) for networks and subnetworks for use by TCP/IP at NCP generation time.

The RIT consists of routing tables that are generated from the NCP IPROUTE and IPLOCAL statements. During NCP generation, the RIT is added as a member of the NCP load library partitioned data set ncp.v7r1.ncpload. You identify the member name of ncp.v7r5.ncpload that NCPROUTE uses at execution time with the NEWNAME parameter of the BUILD statement for each NCP client generation.

Each entry in the routing table is created, for static routing, from the definitions in the TCP/IP Profile data set:

```
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
  193.12.2 130.50.10.1   TR1       2000       0
  10.5.6.4 193.5.2.10    ETH1      1500       HOST
;
```

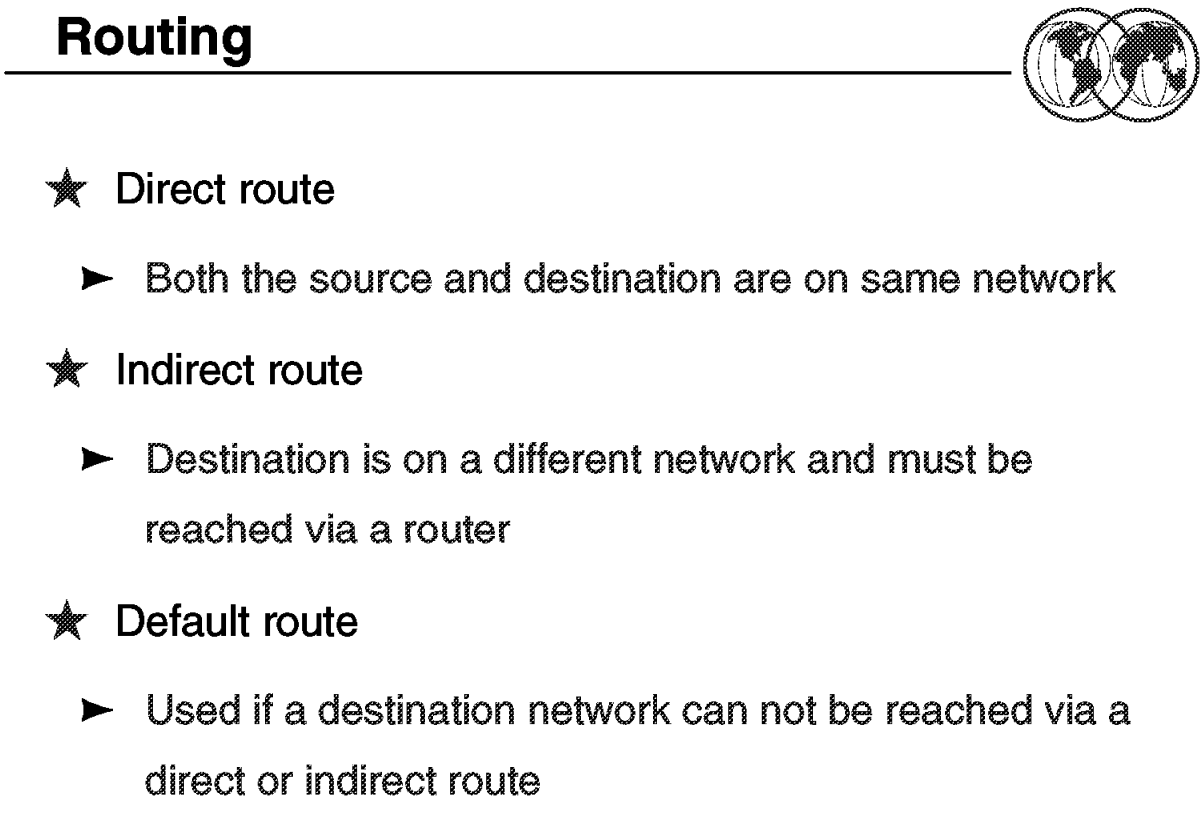


Figure 67. Routing

1.27.6 Routing

The following types of routing are defined in the TCP/IP Profile data set:

Direct Direct routing can take place when two hosts are directly connected to the same physical network. This can be a bridged token-ring network, a bridge Ethernet, or a bridged token-ring network and Ethernet. The distinction between direct routing and indirect routing is that with direct routing IP datagram can be delivered to the remote host without subsequent interpretation of the IP address, by an intermediate host or router.

Indirect Indirect routing takes place when the destination is not on a directly-attached IP network, forcing the sender to forward the datagram to a router for delivery.

Default A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the passive route definition in the gateways file or data set. For example, if the default destination router has a gateway address 9.12.112.1, then add the following entry to the data set:

```
net  0.0.0.0  gateway  9.12.112.1  metric  1  passive
```

Only one default route to a destination gateway or router can be specified. OROUTED currently does not support multiple default routes.

```
IPROUTE  DESTADDR=0.0.0.0,NEXTADDR=9.12.112.1,INTFACE=TR88,
         METRIC=1,DISP=PERM
```

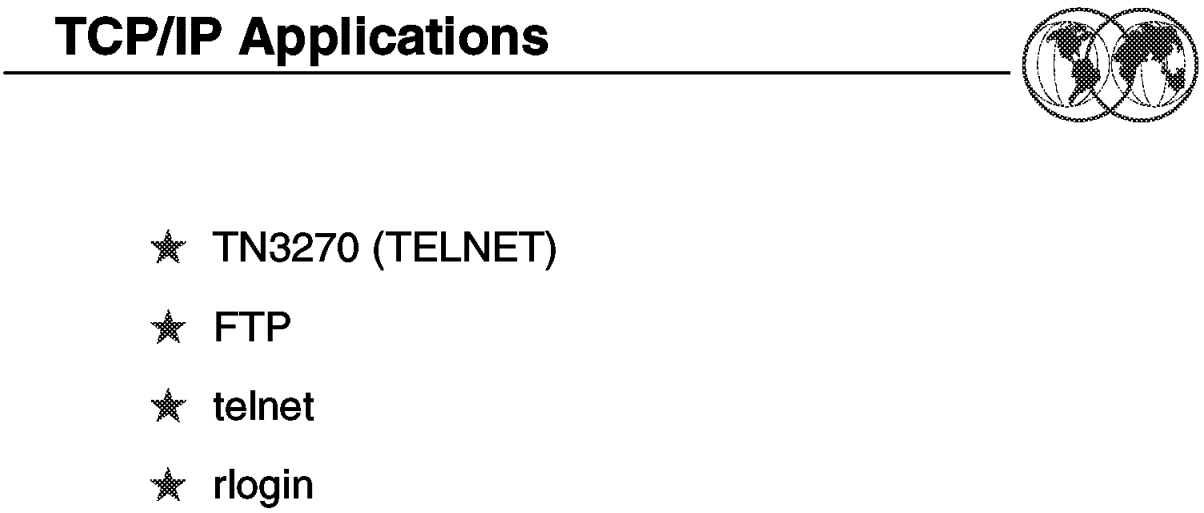


Figure 68. TCP/IP applications

1.28 TCP/IP applications

TN3720 TELNET is most often used as the primary method of connection between client workstations and the SNA mainframe environment. TELNET terminal emulation needs to simulate actual SNA terminals as closely as possible to make this form of remote connection as seamless as possible to end users. RFC1647, also known as TN3270E, adds the ability to simulate specific terminal LU connection and support printer devices and additional SNA functions.

FTP File Transfer Protocol (FTP) lets you transfer data sets between the local host and any other host that supports TCP/IP. Using the FTP command and its subcommands, you can sequentially access multiple hosts without leaving the FTP environment.

telnet The telnet support comes with the TCP/IP OS/390 UNIX feature. It also uses the inetd daemon which must be active and set up to recognize and receive the incoming telnet requests.

OS/390 UNIX telnet code is installed in the hierarchical file system (HFS) (path /usr/lpp/tcpip/sbin/otelnetd with a symbolic link to /usr/sbin/otelnetd) and in the MVS data set hlq.SEZALINK. The hlq.SEZALINK data set needs to be a PADS protected data set if you are running with the BPX.DAEMON facility class defined. OS/390 UNIX checks whether the sticky bit is set on in the HFS. If it finds the sticky bit on, it first checks for an executable file in the MVS data set. If it does not find the executable file in a data set in the MVS search order, OS/390 UNIX then uses the executable file in the HFS.

rlogin

When the inetd daemon is set up and active, you can rlogin to the shell from a workstation that has rlogin client support and is connected via TCP/IP or Communications Server to the MVS system. To login, use the `rlogin` (remote log in) command syntax supported at your site.

The inetd daemon provides service management for a network. For example it starts the rlogind program whenever there is a remote login request from a workstation.

The rlogind program is the server for the remote login command `rlogin`, commonly found on UNIX systems. It validates the remote login request and verifies the password of the target user. It starts an OS/390 shell for the user and handles translation between ASCII and EBCDIC code pages as data flows between the workstation and the shell.

When inetd is running and receives a request for a connection, it processes that request for the program associated for that socket. For example, if a user tries to log in from a remote system into the OS/390 shell while inetd is running, inetd processes the request for connection and then issues a fork() and execl() to the rlogin program to process the rlogin request.

It then goes back to monitoring for further requests for those applications that can be found as defined in the /etc/inetd.conf file.

TN3270 Definitions

★ TN3270 definitions

► Specify in TCP/IP PROFILE data set

► Specify logmode using TELNETDEVICE in VTAM

```
BEGINVTAM
TELNETDEVICE .....
....
ENDVTAM
```

Figure 69. TN3270 parms

1.28.1 TN3270 parms

The TELNET server gets some of its configuration parameters through the TELNETPARMS statements. If you want to specify parameters for the TELNET server, update the TELNETPARMS section of the TCP/IP Profile data set.

```
;
;  TELNET parms
  TELNETPARMS
    PORT 623
    WLMCLUSTERNAME TN3270E ENDWLMCLUSTERNAME
  ENDTELNETPARMS
;
```

The TELNETDEVICE statement lets you specify a logmode for a device type anywhere within the BEGINVTAM/ENDVTAM block, instead of just at the beginning of the block (as with the BEGINVTAM statement). This statement accepts two logmodes: one for 3270 connections and one for 3270E connections.

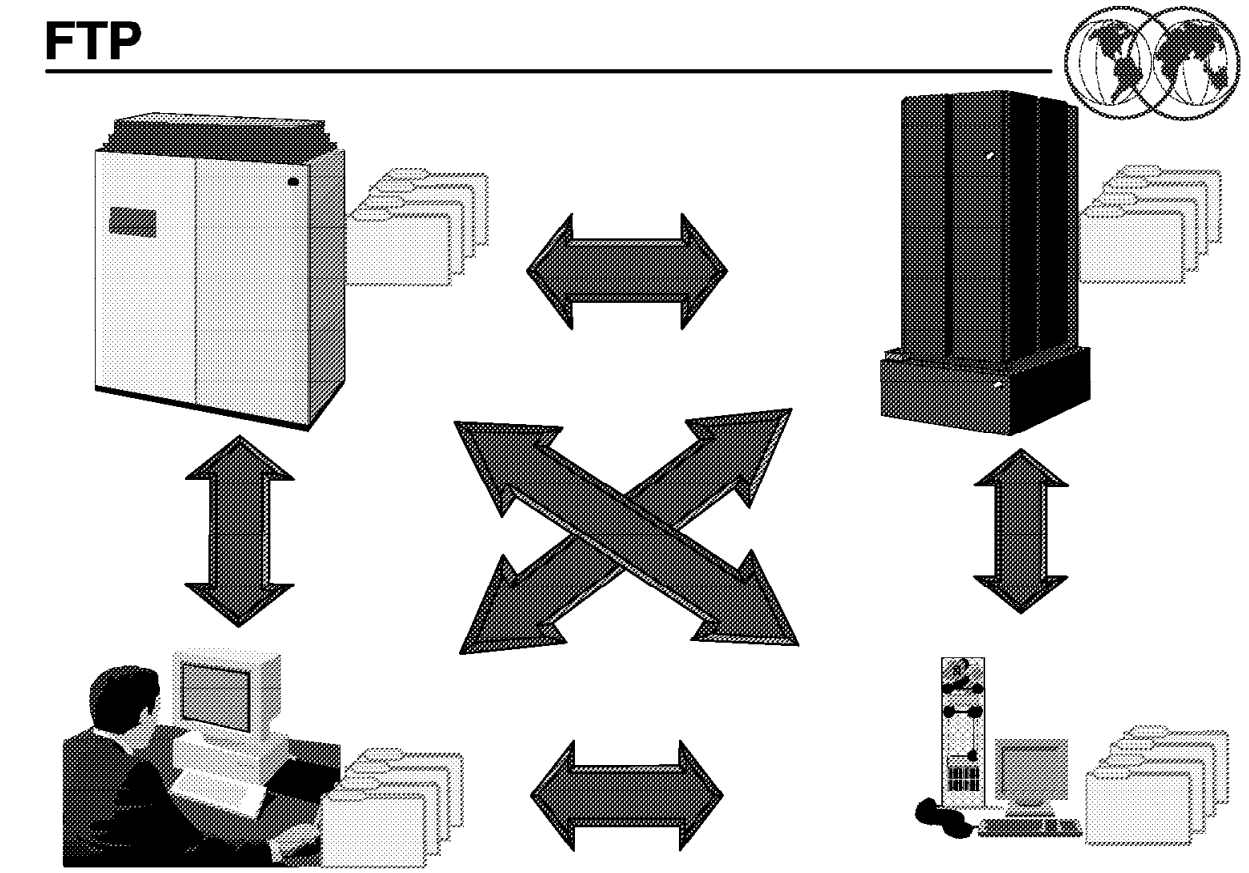


Figure 70. FTP

1.28.2 FTP

File Transfer Protocol (FTP) is the simplest way to transfer files between computers connected by a TCP/IP network. The direction of the file transfer can be either PUSH (send a file to the remote host) or PULL (get a file from the remote host). The local and remote hardware and software are largely irrelevant, and you can select to transfer files in ASCII, BINARY, or EBCDIC format from any machine to any other machine.

For example, a flat text file can be transferred in ASCII format from a PC workstation to VM/ESA, or OS/390, or UNIX and its contents will still be readable as a flat text file when it arrives at its destination. A compressed (zipped) PC file can be transferred in binary format to an OS/390 machine for storage and then transferred from there to a different PC where it will arrive intact and can be successfully decompressed (or unzipped).

FTP is a client-server protocol where the initiating side of the connection is the FTP client and the client connects to an FTP server.

FTP provides various commands to connect to a remote host, change the active directory, define the data transfer format, and obviously, send and receive files. A few of the more useful ones are:

AScii Set transfer mode to text

Binary Set transfer mode to binary

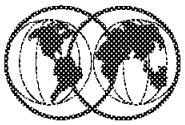
CD Change active Directory on remote side

CLose Close connection

DIR	Get directory listing of remote files
Get	Retrieve files from remote host
MGet	Multiple Get for multiple files (for example, R*.DAT)
MPut	Multiple Put for multiple files
Open	Initiate connection to remote site
PASS	Send password to remote host (if remote host requires a password)
Put	Send file to remote host
PWD	Query Preset Working Directory on remote host
QUIT	Close connection to remote host, or EXIT ftp if no connection active
SYStem	Query the remote operating system type
USer	Initiate a logon to remote host once connection is established

For a complete list of available FTP commands and their usage and syntax refer to *eNetwork Communications Server: IP User's Guide*, GC31-8514.

FTP Setup



- ★ Specify PORT in PROFILE data set
- ★ Update ETC.SERVICES with the port to be used by the FTP server
- ★ Update the FTPD cataloged procedure hlq.SEZAINST(FTPD)
- ★ Specify FTP configuration statements in FTP.DATA
- ★ Specify configuration statements in TCPIP.DATA
- ★ Update /etc/syslog.conf for the FTP server

Figure 71. FTP setup

1.28.3 FTP setup

Following are the steps involved in setting up the FTP server on OS/390:

1. Update the TCP/IP Profile data set with:

```
;
PORT
    20 TCP OMVS               ; OE FTP Server
           DELAYACKS          ; Delay transmission acknowledgements
    21 TCP OMVS               ; OE FTPD control port
```

2. Update etc.services

```
ftp             21/tcp
otelnet         23/tcp
```

3. Set-up the FTP server start procedure SYS1.PROCLIB(FTPD). An example is provided in TCP.SEZAINST(FTPD) and is shown in A.4, “Sample FTP start procedure” on page 344.

   The following two DD statements from the procedure are:

```
//SYSFTPD DD DISP=SHR,DSN=SYS1.TCPIP.PARMS(FTPDATA)   1
//SYSTCPD  DD DISP=SHR,DSN=SYS1.TCPIP.PARMS(TCPDATA)  2
```

Point the SYSTCPD DD (2) at the TCPDATA file (see 1.27, “TCP/IP data sets” on page 117).

4. Customize FTPDATA (1).

   A sample FTPDATA is provided in TCPIP.SEZAINST(FTPSDATA). Copy this member to SYS1.TCPIP.PARMS(FTPDATA).

5. Add FTPD to the AUTOLOG section in the TCP/IP Profile (see the AUTOLOG section under 1.27.1, “Configuring TCP/IP - Profile data set” on page 119).

1.28.3.1 FTP usage

In the following FTP example, the file EXLOCAL VTAMLST is being retrieved from a remote VM/ESA system and stored in the member DAVIETN.STUFF(EXLOCAL). The local OS/390 system in this example is the client side of the connection and the remote VM/ESA system is the server.

```
ftp 9.12.14.1
 IBM FTP CS/390 V2R5  1998 279 14:00 UTC
 FTP: using TCPIPOE instead of INET
 Connecting to:  9.12.14.1 port: 21.
 220-FTPSERVE IBM VM Level at WTSCPOK.ITSO.IBM.COM                        4/19/9
 220 Connection will close if idle for more than 5 minutes.
 NAME (9.12.14.1:DAVIETN):
davietn
 >>> USER davietn
 331 Send password please.
 PASSWORD:

 >>> PASS
 230-DAVIETN logged in; working directory = DAVIETN 191 (ReadOnly)
 230 write access currently unavailable due to other links
 Command:
dir *.vtamlst
 >>> PORT 9,12,2,13,4,22
 200 Port request OK.
 >>> LIST *.vtamlst
 125 List started OK
 EXLOCAL  VTAMLST  F         80         50          1  4/14/99 17:58:23 DAV191
 250 List completed successfully.
 Command:
ascii
 >>> TYPE A
 200 Representation type is ASCII.
 Command:
get exlocal.vtamlst 'davietn.stuff(exlocal)'
 >>> PORT 9,12,2,13,4,25
 200 Port request OK.
 >>> RETR exlocal.vtamlst
 150 Sending file 'exlocal.vtamlst' FIXrecfm           80
 250 Transfer completed successfully.
 4100 bytes transferred in 0.260 seconds.  Transfer rate 15.77 Kbytes/sec.
 Command:
close
 >>> QUIT
 221 Quit command received. Goodbye.
 Command:
quit
 READY
```

As an alternative to the above interactive dialogue, the following simple batch job can be used to achieve the same result.

```
//USERIDX JOB USERID,MSGLEVEL=(1,1),NOTIFY=USERID,MSGCLASS=H
//FTP     EXEC PGM=FTP,REGION=4096K
//INPUT   DD   *
9.12.14.1
davietn
password
type A
get exlocal.vtamlst 'davietn.stuff(exlocal)'
quit
/*
//OUTPUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

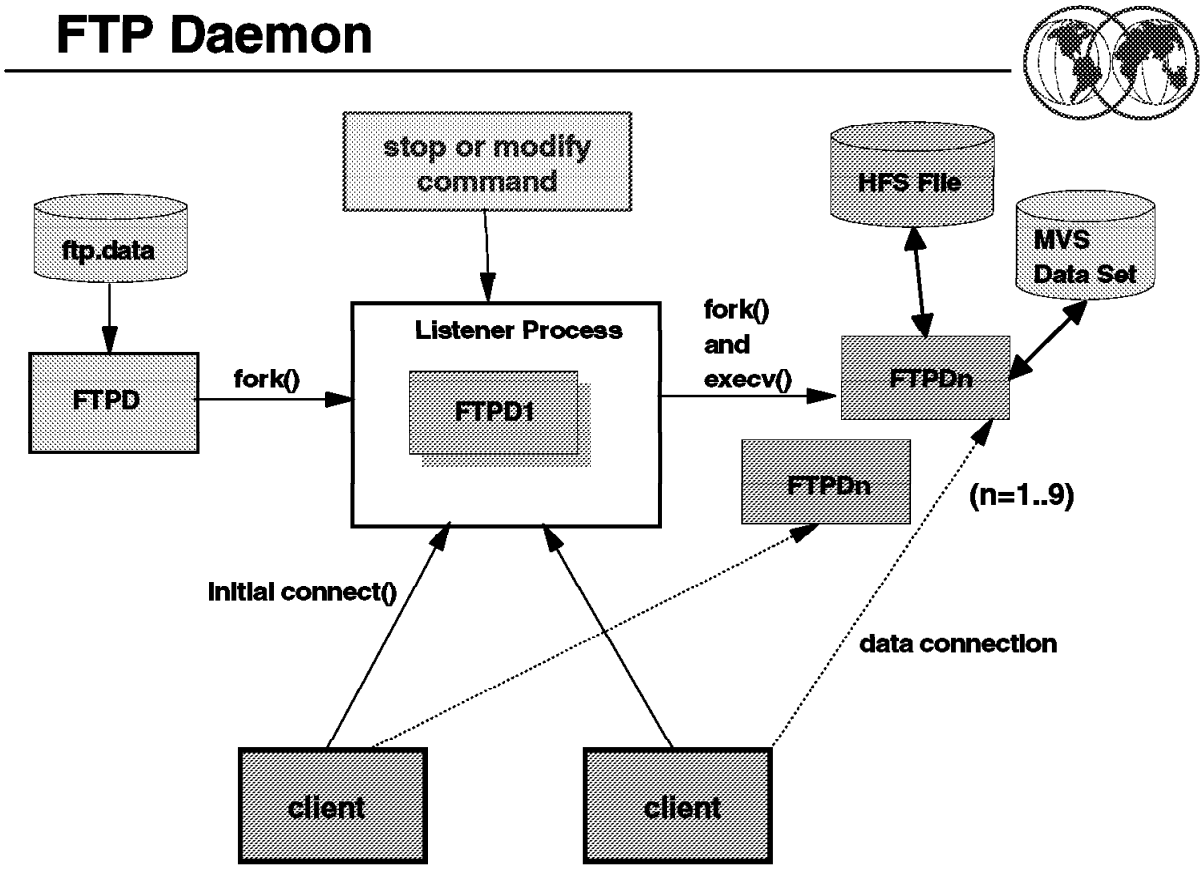


Figure 72. FTP daemon

1.28.4 FTP daemon

File Transfer Protocol (FTP) is used to transfer files between TCP/IP hosts. The FTP client is the TCP/IP host that initiates the FTP session, while the FTP server is the TCP/IP host to which the client connects.

The FTP server uses two different ports and manages two TCP connections:

- Port 21 is used to control the connection (user ID and password).
- Port 20 is used for actual data transfer based on the FTP client's requests.

The FTP server in OS/390 IP consists of the daemon (the listener) or *ftpd* and server address space (or processes). The daemon performs initialization, listens for new connections, and starts a separate server address space for each connection.

When a new client FTP connects to the FTPD daemon process, *ftpd* forks an FTP server process; thus, a new jobname is generated by OS/390 UNIX System Services.

The OS/390 eNetwork Communication Server IP FTP server and client exploit OS/390 UNIX System Services and provide access to both traditional MVS data sets and OS/390 UNIX hierarchical file system files.

The FTP server does not use inetd as a listener process, but has its own listener program, which is the FTP daemon address space.

When a client connects, the daemon forks a new address space to handle that client session.

To start the FTP daemon, you can use /etc/rc as shown on the next visual, or you can use the PROFILE.TCPIP data set.

Additional Information can be found in the following publications:

OS/390 UNIX System Services Planning, SC28-1890

OS/390 eNetwork Communication Server IP Configuration, SC31-8513

OS/390 TCP/IP OpenEdition Implementation Guide, SG24-2141

OS/390 eNetwork Communication Server TCP/IP Implementation Guide Volume 2, SG24-5228

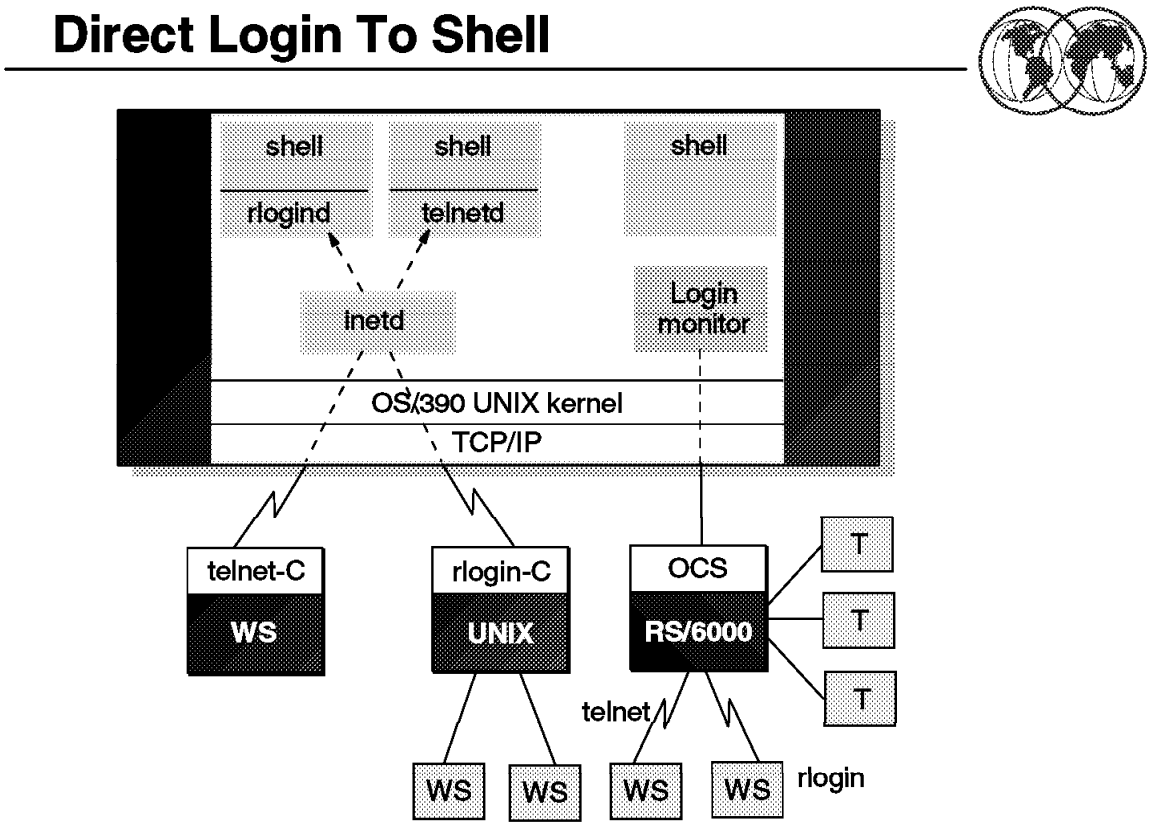


Figure 73. Logging in to OS/390 UNIX shell

1.28.5 Logging in to OS/390 UNIX shell

Typically, a user does not want to log in through an intermediate TSO stage, but would rather use standard UNIX techniques to log in directly to a shell ID in OS/390 UNIX.

The visual shows the many different ways available to start an OS/390 UNIX shell session. Let us review the different mechanisms to start an OS/390 UNIX shell session.

- Started via the TSO OMVS command:
  - A TSO user, logged in via SNA and VTAM, can issue the OMVS commands directly from the TSO command line.
  - A workstation user can use TCP/IP TN3270 protocol to logon to a TSO user ID via IP 3270(E) TELNET server. From TSO, use OMVS to access the shell.
- Started via rlogin done directly from workstation:
  - An UNIX/AIX user can use a standard “rlogin” client to do remote login to OS/390 UNIX. An OS/390 UNIX rlogin daemon initiates the shell.
- Started via telnet done directly from workstation:
  - Any platform can use a standard “telnet” client to do remote login to OS/390 UNIX. An OS/390 UNIX telnet daemon initiates the shell.
- Started via Communication Server (CS) LOGIN request:

  - An ASCII terminal directly attached to an OCS RS/6000 host can use local login to OS/390 UNIX to access the OS/390 UNIX shell.
  - A networked user attached to an RS/6000 CS host can route “rlogin” or “telnet” requests for OS/390 UNIX through a CS client on AIX. The CS login monitor (lm) server on OS/390 UNIX establishes shell sessions on behalf of these clients.
- Both direct “rlogin/telnet” and CS assisted logins support “raw” mode.

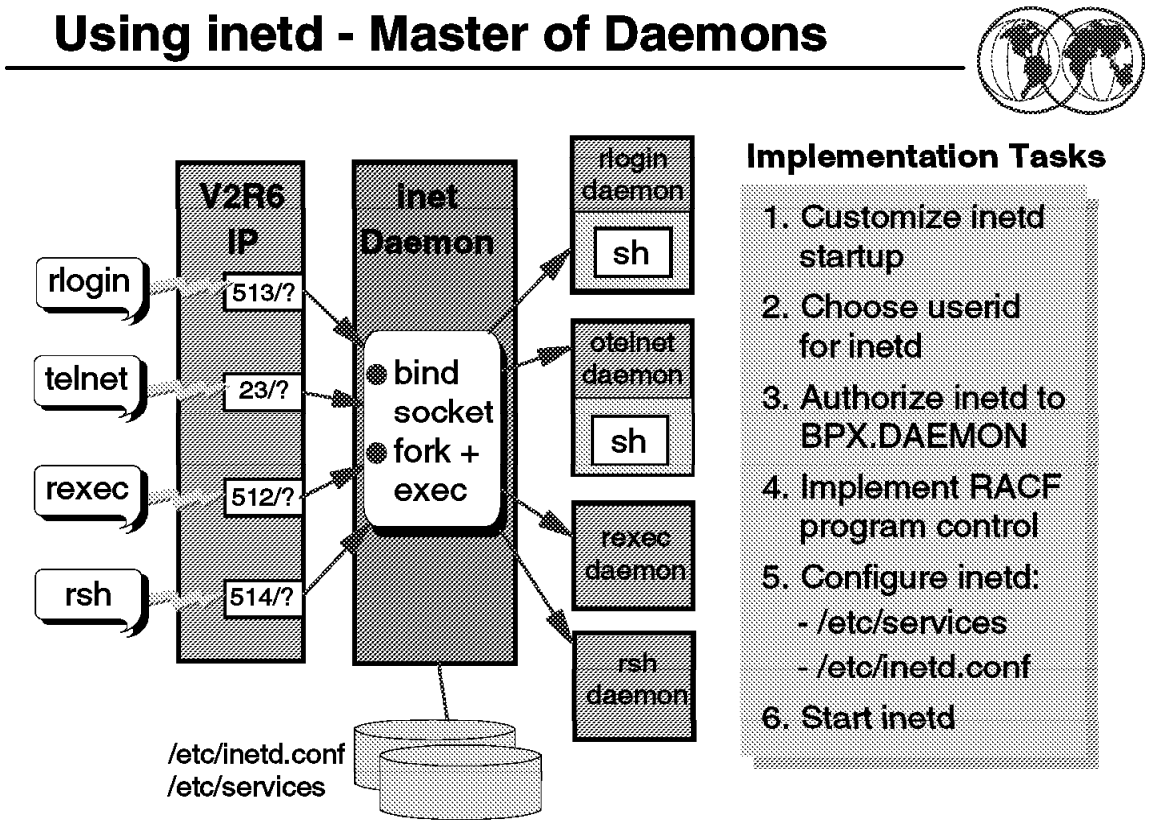


Figure 74. Using inetd - master of daemons

1.28.6 Using inetd - master of daemons

The inetd daemon, usually known as inetd or InetD, is a master of other daemons that execute in OS/390 UNIX. The function of inetd is to listen on certain “well-known” network ports for a request to run one of a number of daemons. When a request is received, inetd creates a new socket for remote connection, and then fork()s a new address space and uses exec() to start the requested daemon program.

The daemon started by inetd relates to the port where the request arrived. The correlation between port number and daemon is stored in configuration file /etc/inetd.conf.

The daemons started by inetd include:

- The rlogin daemon starts a shell session for a user rlogin request.
- The TELNET daemon starts a shell session for a user TELNET request.
- The rexec daemon executes a single command on OS/390 UNIX requested by a remote user entering an rexec command.
- The rsh daemon starts a shell session and runs a script generated by a remote user entering an rsh command.

Customization is needed to enable inetd to run on your system. You must decide how to start it, and what RACF ID it will execute under. If you have implemented enhanced daemon security with BPX.DAEMON, you must define inetd to the BPX.DAEMON and implement program control. Finally, you have to configure the relationship between the ports that inetd listens on and the daemons to be started.

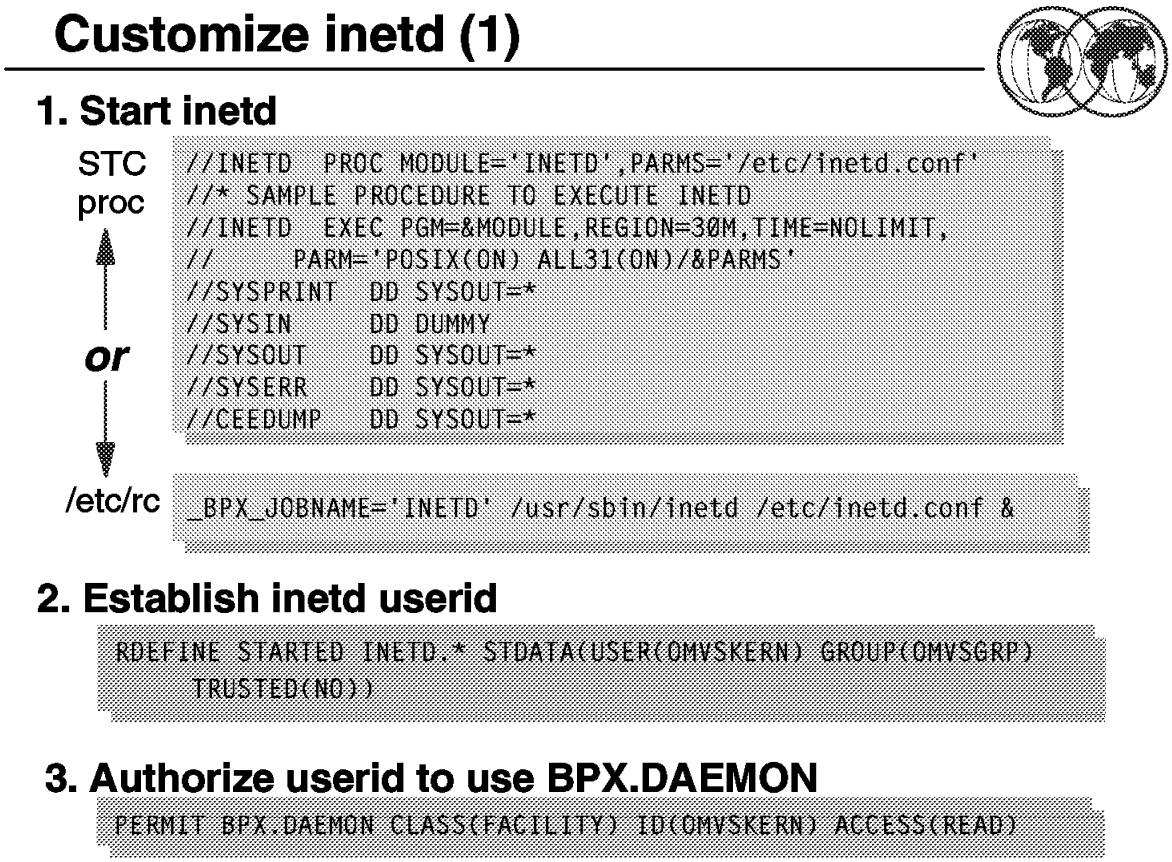


Figure 75. Customize inetd (part 1)

1.28.7 Customize inetd (part 1)

This visual and the next visual describe the steps necessary to customize the inetd daemon.

1. The inetd daemon program can be found in two places. In the HFS, the program file is /usr/sbin/inetd, but IBM has set the sticky bit on. A copy of this program is found in ′SYS1.LINKLIB(INETD)′, so this is the program that is used. You can use any of the methods to start a daemon, but the approved production ways are:
   - Start it from an OS/390 procedure like the one shown. The jobname comes from the label on proc = INETD. Parms supply the two variable values:
     - Program module name
     - Parameter passed to INETD program = name of inetd configuration file
   - Start from a line in the initialization script /etc/rc. In this case, use a command similar to the line shown.
2. The next step is to decide which user ID to associate with inetd. It needs to be a superuser (UID=0), and have minimum access to MVS data sets. How you do this depends on the start mode:
   - If started from /etc/rc, inetd inherits user ID OMVSKERN, which is superuser.
   - If started via STC, you need to associate the RACF UID with the STC name by updating the RACF STARTED facility class. You could use the kernel ID OMVSKERN, or you can create a new superuser ID and then use that one.

3. If you have activated the RACF BPX.DAEMON facility, then the INETD user ID must be authorized to this facility.

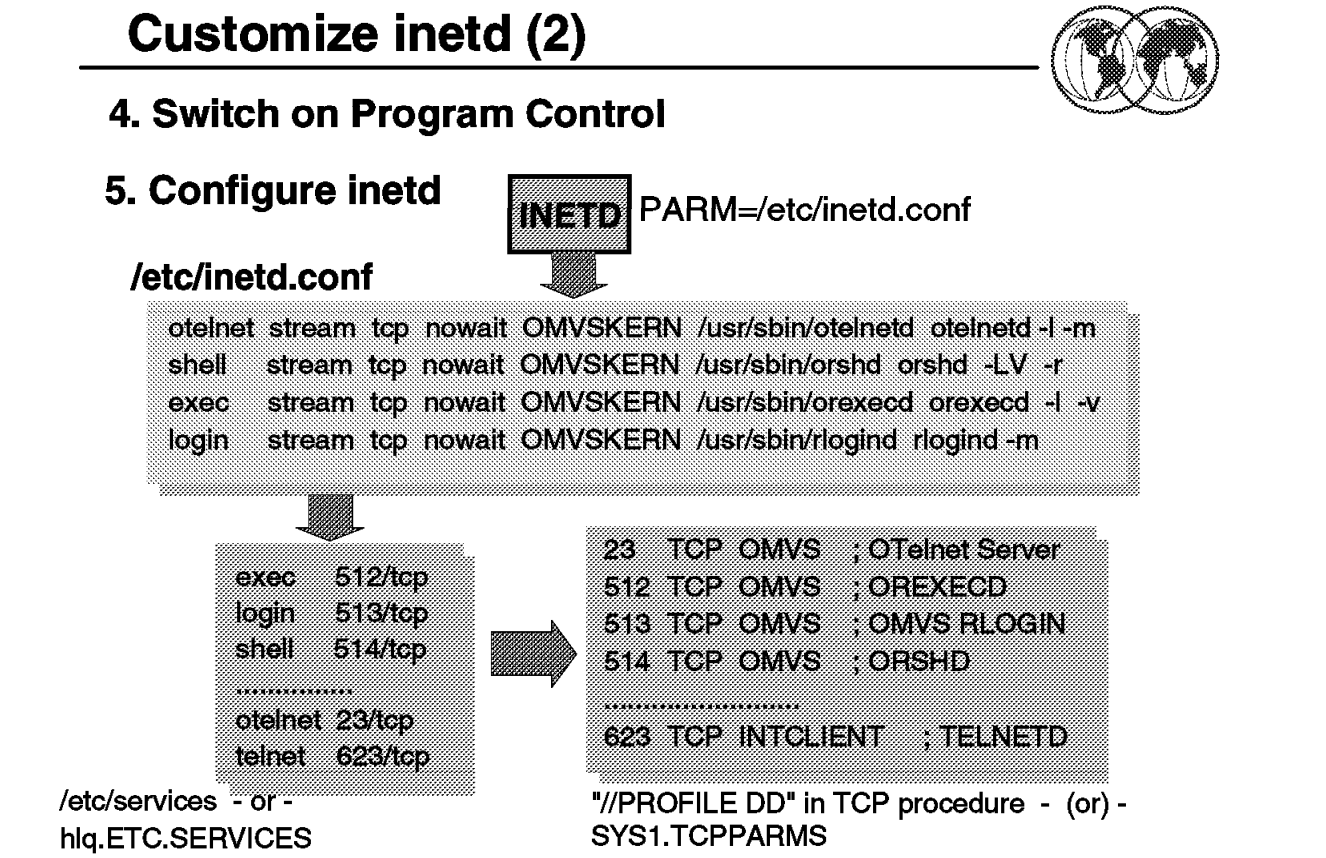


Figure 76. Customize inetd (part 2)

1.28.8 Customize inetd (part 2)

1. If you have set up the BPX.DAEMON, then you need to make sure that all programs are loaded into the inetd address space. At a minimum, you should protect the following programs:
   - SYS1.LINKLIB(INETD).
   - CEE.SCEERUN - LE/MVS run time - whole library
2. There are three configuration files that have to be updated for inetd support:
   - The primary file is /etc/inetd.conf, which is the inetd configuration file. There is one entry (line) in this file for each daemon controlled by inetd. The fields are interpreted as follows:
     - Field (1) - Service name - match daemon entry in /etc/services file
     - Field (2) - Daemon socket type - stream or dgram
     - Field (3) - Daemon socket protocol - TCP or UDP
     - Field (4) - Wait_flag - can be wait (single thread server - one request at a time) or nowait (multiple requests queued)
     - Field (5) - Login_name - RACF user ID under which daemon will run
     - Field (6) - Server_program - name of daemon program in HFS
     - Field (7) - Server-arguments - first string is jobname for daemon address space, and the rest is the parm string to pass to daemon
   - There is a corresponding entry in /etc/services for each daemon in inetd.conf. The entry lists the port where inetd listens for daemon requests.

- The TCP/IP Profile configuration must list the same ports in the PORT section. This entry identifies the job name authorized to open the socket to this port and the type of socket allowed.
- The two TCP/IP files usually exist already—you must make sure that inetd.conf corresponds with the values listed. You may want to change the port number for a daemon.

3. After all configuration is complete, start inted.

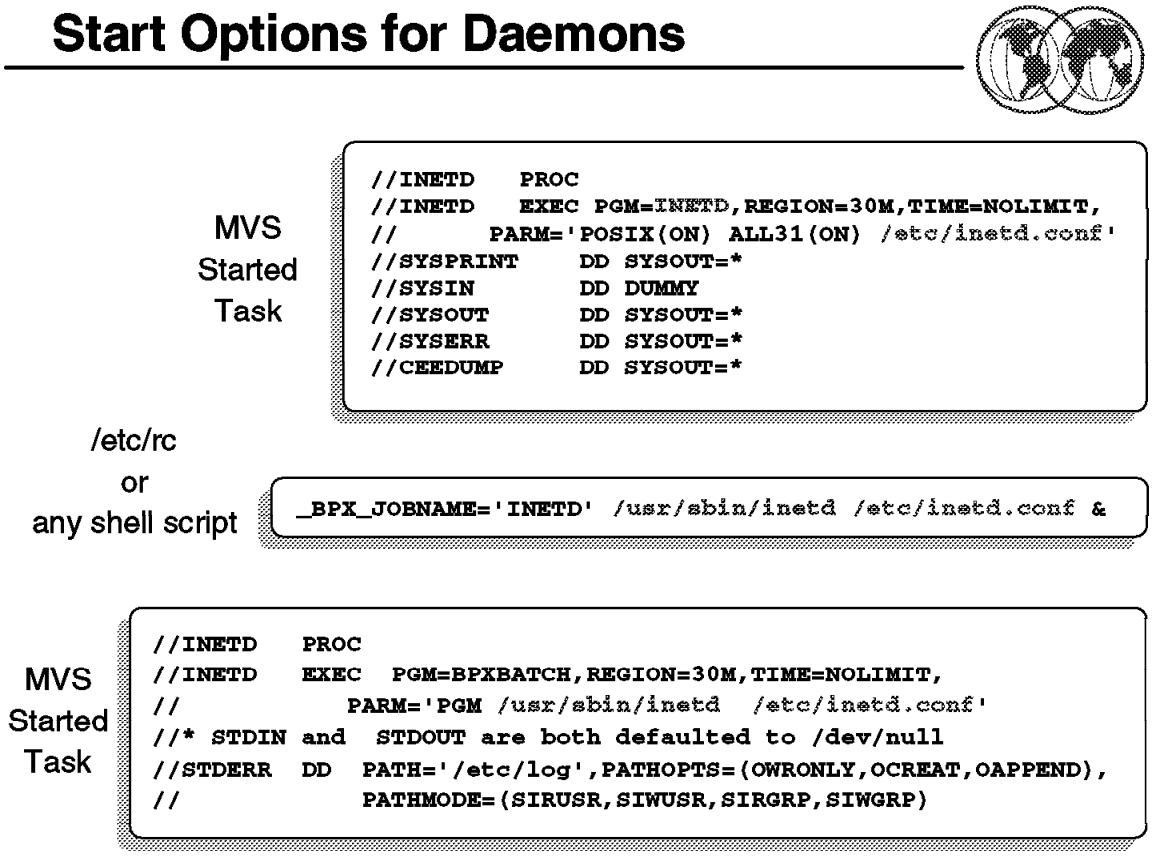


Figure 77. Start options for daemons

1.28.9 Start options for daemons

In an OS/390 system, there are several ways of starting and restarting daemons. The method used depends on the level of control the installation has chosen for daemons. The daemon programs are installed in /usr/sbin with the sticky bit on and in SYS1.LINKLIB. Daemons can be started using the following methods:

- As a cataloged procedure (started task). Create a cataloged procedure. In order for the procedure to get control with the correct authority, an entry must be added to the RACF started procedures table (ICHRIN03) or a profile added to the STARTED class. The daemons need superuser authority. Start the daemon by using the operator command S daemon_proc_name (for example, 'S INETD').
- To be started automatically when the kernel is started, place the start options in the HFS file called /etc/rc. The initialization of OS/390 UNIX includes running the commands in /etc/rc. The _BPX_JOBNAME environment variable assigns a job name to the daemon.
- As a cataloged procedure using the BPXBATCH program to invoke the daemon program.

If daemons need to be stopped, the kill command is typically used. Some daemons may have their own specific method of shutdown.

You should have appropriate procedures and directions in place to restart these daemons in case of failure. Started procedures are one way to do this, which may be more attractive depending on your automation strategy.

In an OS/390 system, most started tasks which are continuously executing are started from a cataloged procedure, for example located in SYS1.PROCLIB. The visual shows an example with a proc called INETD in SYS1.PROCLIB. It will execute the program INETD from SYS1.LINKLIB. To associate the INETD procedure with the correct user ID, an entry must be created in the RACF started procedures table (ICHRIN03) or a profile added to the STARTED class. This is done in the same way as we did for the OMVS procedure. The HFS file /etc/inetd.conf is a configuration file for the inetd daemon. It contains start options for the daemons controlled by inetd.

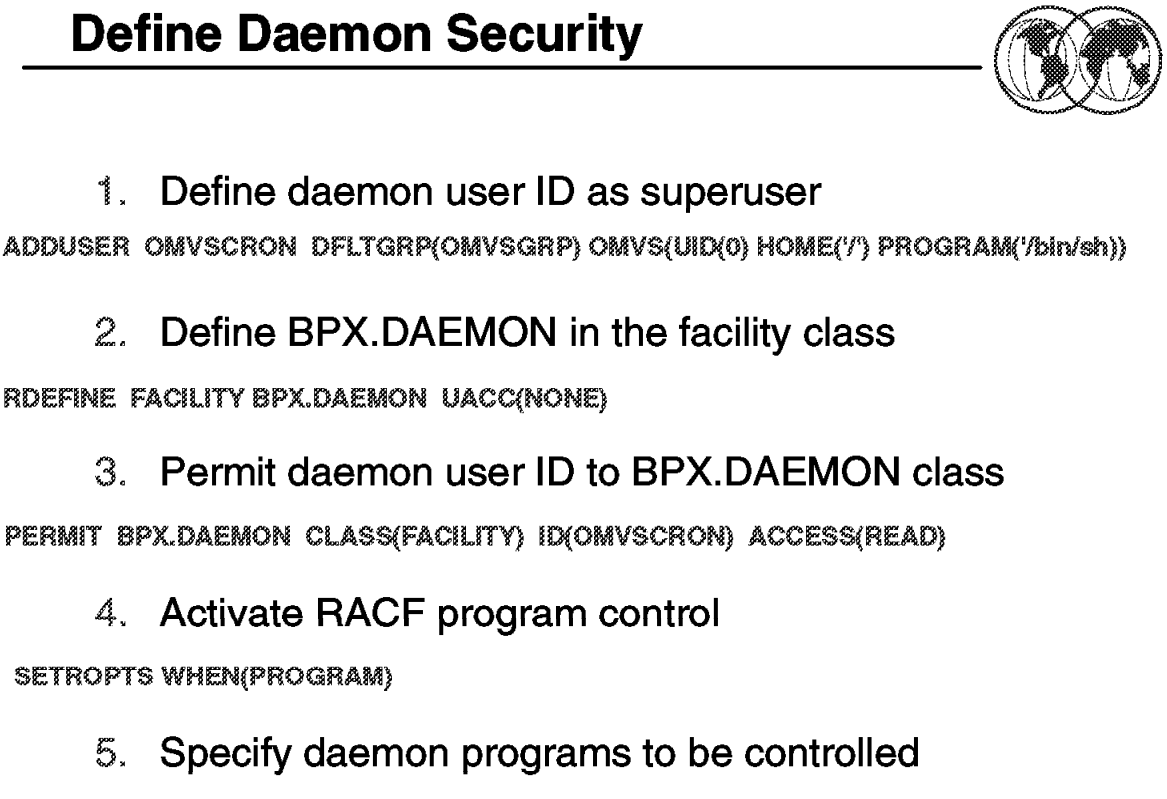


Figure 78. Define daemon security

1.28.10 Define daemon security

In many cases a daemon program is started from the kernel and will inherit the kernel user ID, OMVSKERN. This example shows that it can have a separate user ID as long as the user ID is defined as a superuser. This superuser must be defined with a UID=0 in RACF, which means that this user cannot become a superuser by using the su command.

Note: This user ID should not have a TSO/E segment defined; only the OMVS segment is needed.

The following steps describe how to define security for a daemon:

1. Define a user ID for the daemon which is a superuser with UID=0, for example OMVSCRON:

   ```
   ADDUSER OMVSCRON DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh
   ```

2. Define the BPX.DAEMON FACILITY class in RACF:

   ```
   RDEFINE FACILITY BPX.DAEMON UACC(NONE)
   ```

   The name BPX.DAEMON must be used. No substitutions for this name are allowed. UACC(NONE) is recommended.

   If this is the first RACF FACILITY class defined in RACF, the SETROPTS command must be used to activate the class.

   ```
   SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
   SETROPTS RACLIST(FACILITY)
   ```

3. Permit the daemon user ID to the BPX.DAEMON class:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSCRON) ACCESS(READ)
```

4. Activate RACF program control:

   ```
   SETROPTS WHEN(PROGRAM)
   ```

   You can choose to protect a whole program library or individual load modules (members) in a library. The daemon program must reside in a program-controlled MVS partitioned data set, or in an HFS file with the extended attribute turned on via the `extattr +p` command. Both the program library for the daemons (for example, SYS1.LINKLIB) and the C runtime library must be protected.

5. Protect the program libraries that need to be protected from unauthorized updates:

   ```
   ADDSD 'SYS1.LINKLIB' UACC(READ)
   ADDSD 'SYS1.SCEERUN' UACC(READ)
   ADDSD 'SYS1.SEZALINK' UACC(READ)
   ADDSD 'SYS1.SEZATCP' UACC(READ)
   ADDSD 'SYS1.SIMWMOD1' UACC(READ)
   ```

   The ADDSD command creates data set profiles for the data sets. You should protect against unauthorized updates so that nobody can replace a daemon program with a fake daemon program. If these profiles are already defined, this step can be skipped.

   The installation has a choice of either protecting all programs in a program library or individual programs. To protect all members in a data set, specify `PROGRAM *`.

   Note: Libraries in the LNKLIST concatenation are opened during IPL, and the programs in them are available to anyone unless the program name is defined as a controlled program. Mark the data sets as controlled libraries:

   ```
   RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'//NOPADCHK +
                            'SYS1.SCEERUN'//NOPADCHK +
                            'SYS1.SEZALINK'//NOPADCHK +
                            'SYS1.SEZATCP'//NOPADCHK +
                            'SYS1.SIMWMOD1'//NOPADCHK) UACC(READ)
   ```

   Or, mark the daemon program as controlled instead of the whole library:

   ```
   RDEFINE PROGRAM CRON ADDMEM('SYS1.LINKLIB'//NOPADCHK)
   UACC(READ) AUDIT(ALL)
   ```

   Place the PROGRAM profile in storage:

   ```
   SETROPTS WHEN(PROGRAM) REFRESH
   ```

1.29 OSA/SF

Open Systems Adapter/Support Facility (OSA/SF) is comprised of a suite of programs and utilities that are used to re-configure an Open Systems Adapter (OSA). The modes of OSA operation include:

- TCP/IP Passthrough

  Providing communications between the server TCP/IP applications and TCP/IP clients on the LAN
- SNA

  Systems Network Architecture
- HPDT ATM Native

  High Performance Data Transfer Asynchronous Transfer Mode
- ATM IP Forwarding
- HPDT MPC

  High Performance Data Transfer Multi-Path Channel
- LANRES/MVS

  LAN Resource Extension and Services/MVS

The OSA-2 can be configured to allow sharing among logical partitions (LPAR support) using OSA/SF. When the S/390 server is running in LPAR mode, TCP/IP and SNA/APPN applications can share access to an OSA-2 and can access the same LAN port. This is accomplished by configuring the OSA Address Table (OAT), and loading it onto the OSA. The OAT is used to define the possible modes of data transfer through the OSA. Each OSA has its own OAT, and the OAT is stored in the OSA in non-volatile storage, meaning that it can survive power outages and system reloads.

The OSA/SF utilities are shipped with OS/390 both as a downloadable GUI and as TSO REXX EXECs.

OSA/SF and OSA configuration is described in detail in *OS/390 V2R7.0 OSA/SF User's Guide*, SC28-1855.

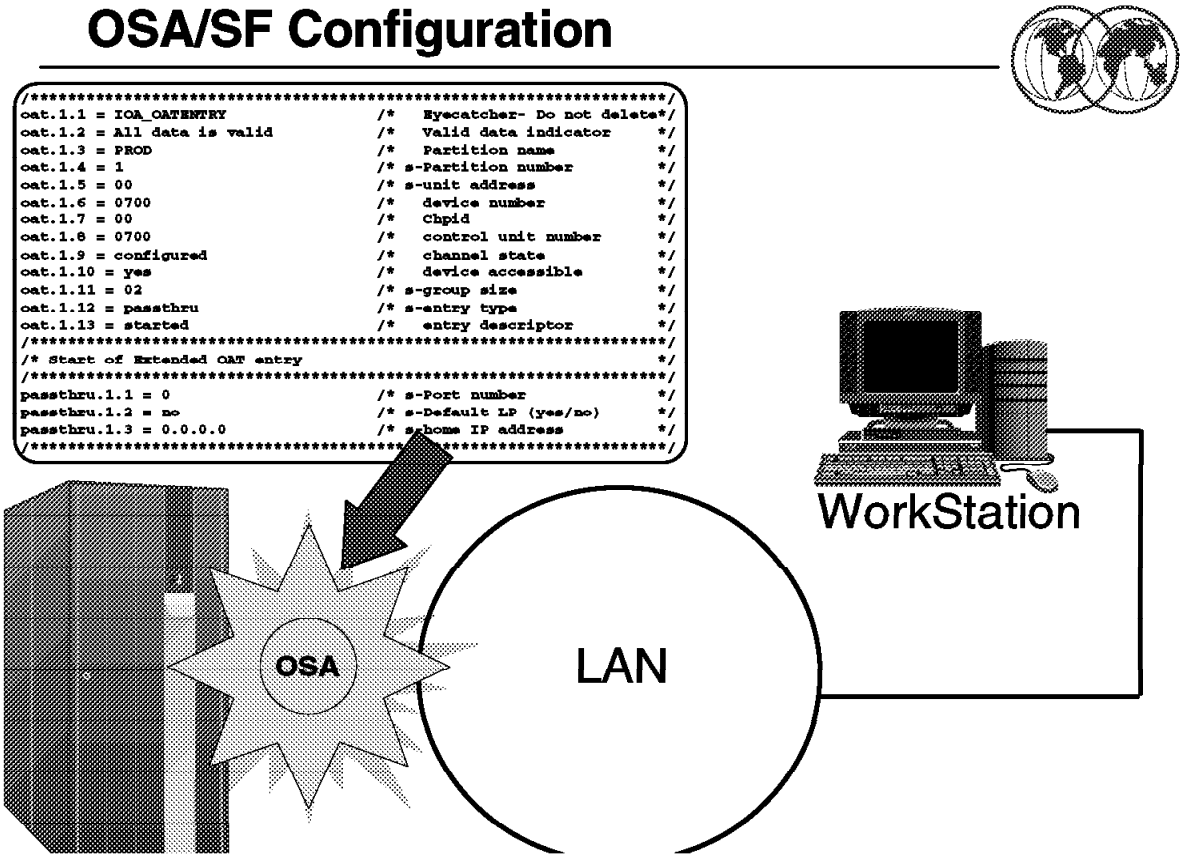


Figure 79. OSA/SF configuration

1.30 OSA/SF configuration

The most widely used type of OSA card is the ENTR that supports both TCP/IP and SNA traffic and can connect to both token-ring and Ethernet LANs.

The default mode of operation for an ENTR is TCP/IP Passthrough. The OSA card is capable of sensing whether it is connected to an Ethernet or a token-ring LAN and will therefore be ready to communicate using TCP/IP in a limited fashion without further modification.

The OSA is configured by using the supplied OSA/SF utilities. Re-configuration is performed by downloading a copy of the currently active OAT, modifying it, and reloading it to the OSA.

An example of a complete OAT is included in Appendix A.5, “Sample OAT” on page 345. Customization and setup of an OAT is described in 1.30.5, “OSA Address Table” on page 161.

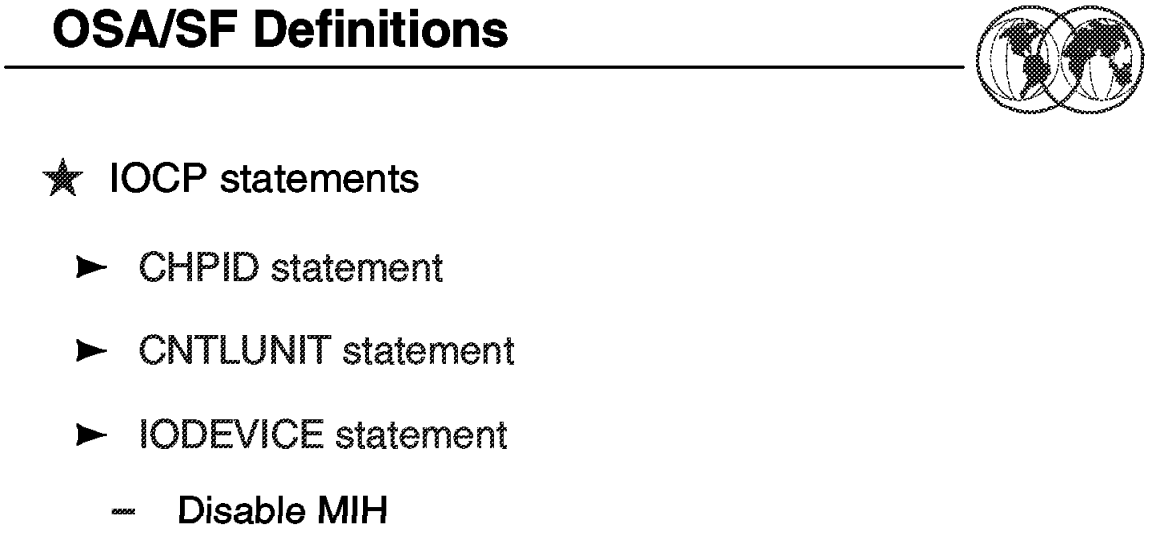


Figure 80. OSA/SF definitions

1.30.1 OSA/SF definitions

The OSA card must be defined to the CPU via IOCP definition statements.

CHPID The CHPID statement defines the OSA Channel Path ID (CHPID) as type OSA. Type OSA is a special kind of internal channel type used only for OSAs:

```
CHPID PATH=(5C),TYPE=OSA
```

This example defines the OSA on channel 5C.

CNTLUNIT The CNTLUNIT statement defines a control unit for the OSA:

```
CNTLUNIT CUNUMBR=0D00,PATH=(1C),UNIT=OSA
```

IODEVICE The IODEVICE statements are as follows:

```
1 IODEVICE CUNUMBR=D00,UNIT=OSA,ADDRESS=(D00,14),              x
        STADET=Y,UNITADD=00
2 IODEVICE CUNUMBR=D00,UNIT=OSAD,ADDRESS=(D0F),                x
        STADET=Y,UNITADD=FE
```

Note: There are two separate IODEVICE statements.

The first (1) defines the device addresses available for use by VTAM and TCP/IP as communications devices. In this example they are D00 through D0D inclusive.

Which of these device addresses are available to VTAM as SNA devices and which are available to TCP/IP as LCS devices will be determined by the OSA Address Table (OAT) described later in 1.30.5, “OSA Address Table” on page 161.

The second (2) defines the device address used internally by OSA/SF to communicate with the OSA itself. This is always the device address *FE*, and is defined as unit type *OSAD*.

*

1.30.1.1 Disabling the Missing Interrupt Handler

To avoid channel and device-end error conditions when running TCP/IP Passthrough, you should disable the Missing Interrupt Handler (MIH). In the previous examples we have defined an OSA device address range of 0D00 through 0D0D. Assume that two of these devices, 0D04 and 0D05, are to be used for TCP/IP communication, the OS/390 Missing Interrupt Handler needs to be disabled for these devices.

Edit the appropriate IECIOSxx member in SYS1.PARMLIB and add the following:

```
MIH TIME=00:00,DEV=(0D04-0D05)
```

Activate this definition using the following operator command:

```
SET IOS=xx
```

Where *xx* is the suffix of the IECIOSxx member you just edited.

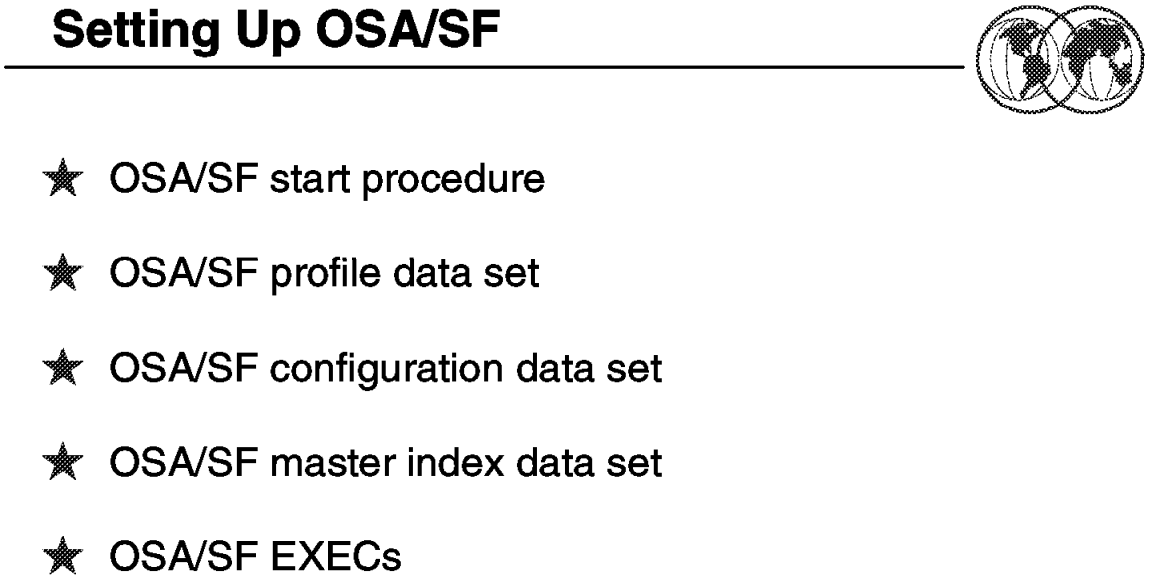


Figure 81. Setting up OSA/SF

1.30.2 Setting up OSA/SF

Detailed instructions on setting up OSA/SF can be found in *OS/390 Open Systems Adapter Support Facility User's Guide*, SC28-1855.

The following steps are a brief summary of the necessary actions required to set up OSA/SF.

1.30.2.1 Set up OSA/SF start procedure

The following is an example of a SYS1.PROCLIB(OSASF) startup procedure.

```
//*
//*   START OSA SUPPORT FACILITY
//*
//OSASF   EXEC PGM=IOAMAIN,TIME=1440,REGION=4M,DYNAMNBR=5
//IOALIB   DD  DSN=SYS1.SIOALMOD,DISP=SHR
//IOAPROF  DD  DSN=SYS1.OSASF.PROFILE,DISP=SHR
//SYSPRINT DD  SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=121)
//SYSUDUMP DD  SYSOUT=*
```

Note the IOALIB DD which points to the OSA/SF load library and the IOAPROF DD which points to the OSA/SF profile.

1.30.2.2 OSA/SF profile data set

The startup profile defines the data set names of the OSA/SF configuration and master index data set as well as the high level qualifier (HLQ) of the OSA/SF working data sets.

```
SET ALIAS SYSNAME SC50
SET ALIAS CECNAME SCZP401
*
SET NAME IOACFG
    Data set 'OSASF.&CECNAME.OSAS.CONFIG'
    UNIT  SYSALLDA
*
SET NAME IOAINX
    Data set 'OSASF.&CECNAME.MASTER.INDEX'
    UNIT  SYSALLDA
*
SET NAME IOAMSG
    Data set 'OSASF.SC68MESSAGE.LOG'
    UNIT  SYSALLDA
*
SET NAME IOADSN
    Data set 'OSASF.&CECNAME.OSASF'
    UNIT  SYSALLDA
```

A sample member is included in IOA.SIOASAMP(IOASPROF) and can be tailored as shown in the above example.

IOACFG The OSA configuration file. It defines what code should be installed for each OSA.

IOAINX The OSA master index file. It defines all the available OSA mode code.

IOAMSG Used by OSA/SF to store system information and status.

IOADSN The HLQ of all data sets that are created for use by OSA/SF.

1.30.2.3 OSA/SF configuration data set

Allocate the configuration data set with the following attributes:

```
TRACKS(2,2)
RECFM=FB
LRECL=80
BLKSIZE=5120
```

Copy the contents of the sample member IOA.SIOASAMP(IOACFG).

1.30.2.4 OSA/SF master index data set

Allocate the master index data set with the following attributes:

```
TRACKS(2,2)
RECFM=FB
LRECL=80
BLKSIZE=5120
```

Copy the contents of the sample member IOA.SIOASAMP(IOAINX), and edit it to change the HLQ of the host destination names that start in column 1 to match yours.

1.30.2.5 Create OSA/SF EXECs

IOACMD is the TSO command used to enter OSA/SF commands.

Allocate a data set named IOACMD.EXEC with the following attributes:

```
TRACKS(14,35)
RECFM=FB
LRECL=80
BLKSIZE=5120
```

Copy the contents of IOA.SIOASAMP(IOACMD) to this data set.

IOAINSNA is the TSO command used to load the SNA microcode onto the OSA.

Allocate a data set named IOAINSNA.EXEC with the following attributes:

```
TRACKS(14,35)
RECFM=FB
LRECL=80
BLKSIZE=5120
```

Copy the contents of IOA.SIOASAMP(IOAINSNA) to this data set.

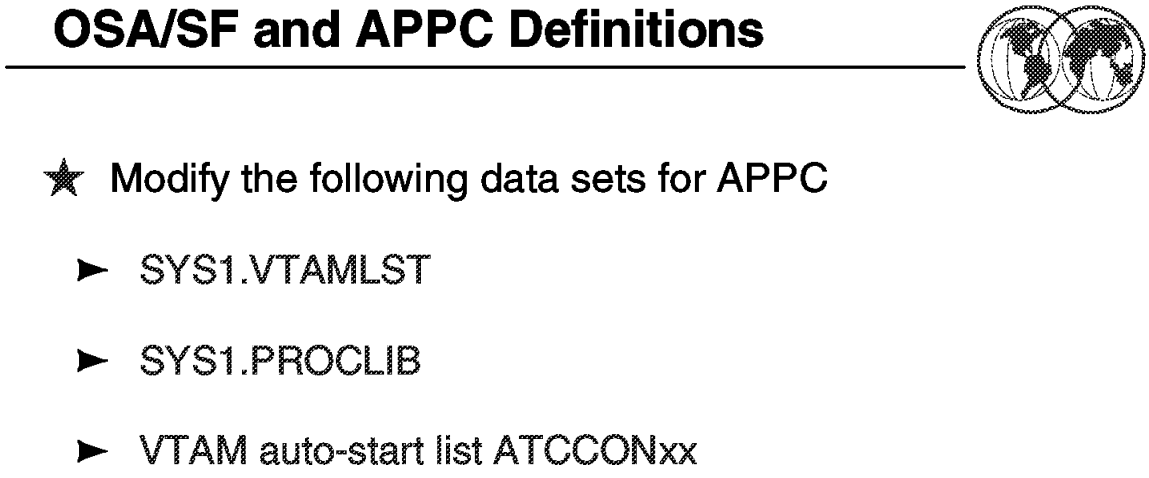


Figure 82. OSA/SF and APPC definitions

1.30.3 OSA/SF and APPC definitions

If you have not already set up APPC, refer to *OS/390 V2R6.0 MVS Planning: APPC/MVS Management*, GC28-1807, and complete the setup of APPC prior to continuing as follows:

1. Modify your current SYS1.VTAMLST(APPCPMxx) member and add the following:

```
IOASERV APPL   ACBNAME=IOASERV,
               APPC=YES,
               AUTOSES=0,
               DDRAINL=NALLOW,
               DMINWNL=5,
               DMINWNR=5,
               DRESPL=NALLOW,
               DSESLIM=10,
               LMDENT=19,
               MODETAB=MTAPPC,
               DLOGMOD=APPCHOST,
               PARSESS=YES,
               SECACPT=CONV,
               SRBEXIT=YES,
               VPACING=1
```

2. Modify your existing SYS1.PROCLIB(APPCPMxx) member and add the following:

```
LUADD ACBNAME(IOASERV)
      NOSCHED
      TPDATA(SYS2.APPCTP)
      TPLEVEL(SYSTEM)
```

3. APPC must be stopped and restarted to activate the above changes.

   Stop APPC by entering the following operator command:

```
C APPC
```

4. Restart APPC by entering the following operator command:

```
S APPC,SUB=MSTR,APPC=xx
```

5. Verify that APPC is running:

```
D A,L or D A,APPC
```

6. Verify that OSA/SF APPC LU is active:

```
D APPC,LU,ALL
```

1.30.3.1 Set up OSA/SF VTAM definitions

1. Copy IOA.SIOASAMP(IOAAPPL) into SYS1.VTAMLST, rename it to APPCOSA, and add an entry for it into VTAM's auto-start list ATCCONxx.
2. Activate APPCOSA:

```
V NET,ACT,ID=APPCOSA
```

3. Build a logmode table and place it into the SYS1.VTAMLIB data set.

   You can use the supplied sample job SYS1.SAMPLIB(ATBLJOB) to build a logmode table and place it into SYS1.VTAMLIB.

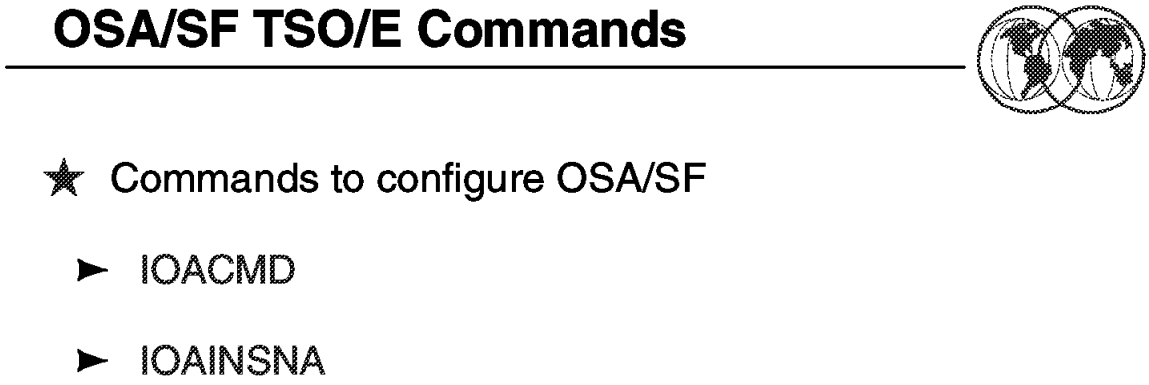


Figure 83. OSA/SF TSO/E commands

1.30.4 OSA/SF TSO/E commands

After you have completed the OSA/SF setup steps in 1.30.2, “Setting up OSA/SF” on page 154 you will be able to use the OSA/SF commands IOACMD and IOAINSNA to configure the OSA for the mode of operation you require.

1.30.4.1 IOACMD

IOACMD has many options and parameters, but until you are completely familiar with their use and syntax you can just enter the TSO command IOACMD and OSA/SF will prompt you for any additional parameters required.

```
IOACMD: Enter the command to be issued

IOACMD:  0 - End IOACMD
IOACMD:  1 - Clear Debug
IOACMD:  2 - Delete File
IOACMD:  3 - Get Console
IOACMD:  4 - Get Debug
IOACMD:  5 - Get File
IOACMD:  6 - Get OSA Address Table
IOACMD:  7 - Install
IOACMD:  8 - List File
IOACMD:  9 - Put File
IOACMD: 10 - Put OSA Address Table
IOACMD: 11 - Query
IOACMD: 12 - Remove Directory
IOACMD: 13 - Send Command
IOACMD: 14 - Set parameter
IOACMD: 15 - Start managing
IOACMD: 16 - Stop managing
IOACMD: 17 - Synchronize
IOACMD: 18 - Get Configuration File
IOACMD: 19 - Configure OSA CHPID
```

Important

The PTF for APAR OW33393 merged the IOAINSNA EXEC into the IOACMD EXEC.

This appears above as the option titled `IOACMD: 19 - Configure OSA CHPID`.

If this option is not displayed by IOACMD use the IOAINSNA command.

1.30.4.2 IOAINSNA

`IOAINSNA` IOAINSNA is the OSA/SF TSO command used to load the SNA microcode onto the OSA if you intend to run SNA traffic over the OSA.

Again, if you just enter the TSO command `IOAINSNA` you will be prompted for any required parameters.

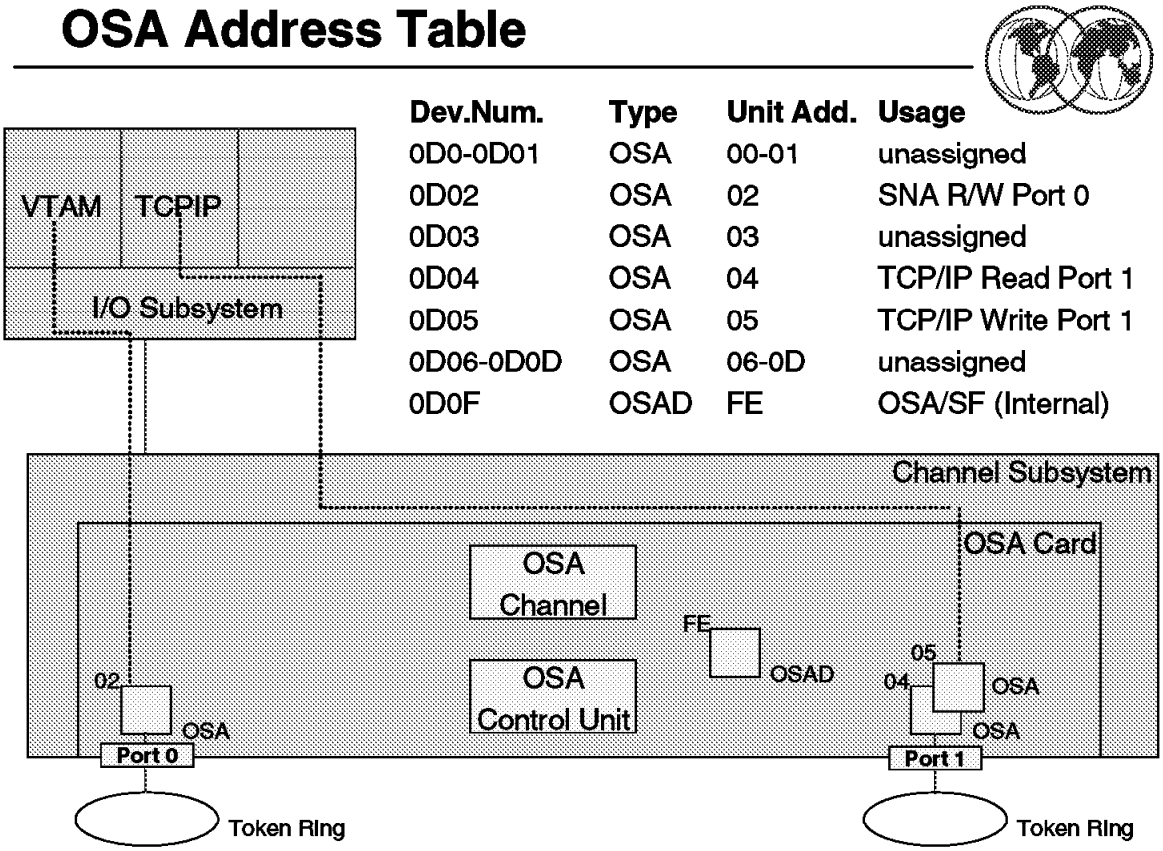


Figure 84. OSA Address Table

1.30.5 OSA Address Table

The OSA Address Table (OAT) is a parameter file that determines how traffic with pass through the OSA. It contains many individual definitions known as *OAT entries* that define the usage of the ports on the OSA. On a default OAT, there are 64 entries: two per port multiplied by 16 possible LPARs. Various examples of sample OATs are included in the IOS.SIOASAMP data set and there is also one in Appendix A.5, “Sample OAT” on page 345.

The format of an OAT is:

- The OAT Header *followed by*
- OAT Entries *and*
- OAT Extended Entries

1.30.5.1 OAT header

An OAT only contains *one* OAT header, and it is the first entry at the top of the OAT. The following is an example of an OAT header:

```
oathdr.1 = IOA_OAT_HDR                /*   Eyecatcher-Do not change */
oathdr.2 = 5C                         /*   Chpid                    */
oathdr.3 = 8                          /* s-Number of entries        */
```

The fields on the left of the entry define the OAT parameters. The fields on the right between the /* and */ pair are simply comments describing the parameter.

oathdr.1 OAT header line 1 is simply defining this entry as an OAT header.

oathdr.2 The OSAs chpid

oathdr.3 The number of port entries contained in this OAT. The default value is 64, specifying two entries per port repeated for the 16 possible LPARs. More than one entry can reference the same OSA device address, making definitions for access to the OSA from multiple LPARs.

1.30.5.2 OAT entry

An OAT entry is a collection of fields grouped together to identify parameters that define one data path through an OSA. There are usually more entries in the OAT than you will use because the OSA ships with default entries for all possible logical partitions when the S/390 is in LPAR mode. These can be modified so that you do not have to create new entries. Each entry in the OAT consists of 13 fields and, where necessary, an additional extended entry of one, two, or three fields. The 13 OAT entry fields are labeled in the format OAT.*x.y*, where *x* is the OAT entry number and *y* simply increments from 1 to 13 for each of the 13 fields.

For example, in A.5, “Sample OAT” on page 345, the OAT fields are as follows:

- The 13 fields for the first OAT entry are labeled oat.1.1 through oat.1.13.
- The 13 fields for the tenth entry will be oat.10.1 to oat.10.13

The OAT entries and their corresponding extended entries are explained in 1.30.5.3, “OAT extended entries.” Entries 1 and 2 define a TCP/IP transmit and receive pair of devices. The third entry defines an SNA device.

1.30.5.3 OAT extended entries

OAT extended entries are labeled using the same numerical suffix format as the OAT entry, but the prefix will be different depending on the type of port the corresponding OAT entry is defining. The possible extended entry types for the ENTR OSA are:

PASSTHRU In the previous example 1.30.5.2, “OAT entry,” OAT entries 1 and 2 define the TCP/IP Passthrough transmit and receive pair as device addresses D04 and D05. The corresponding extended entry for entry 1 is:

```
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
passthru.1.1 = 1                    /* s-Port number             */
passthru.1.2 = no                   /* s-Default LP (yes/no)     */
passthru.1.3 = 195.183.66.11        /* s-home IP address         */
/*********************************************************************/
```

SNA Again, referring to the example shown in 1.30.5.2, “OAT entry,” entry 3 defines an SNA device. The corresponding extended entry is:

```
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
sna.3.1 = 00                        /* s-Port number             */
/*********************************************************************/
```

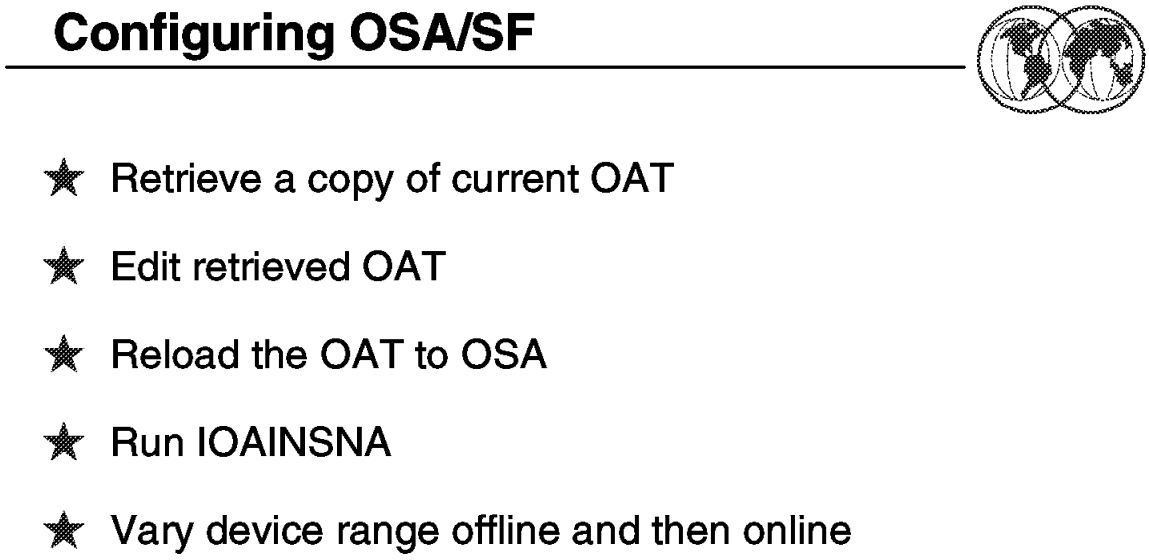


Figure 85. Configuring OSA/SF

1.30.6 Configuring OSA/SF

Complete instructions for OSA configuration and reconfiguration are documented in *OS/390 Open Systems Adapter Support Facility User's Guide*, SC28-1855.

The following is intended only as a brief outline of the necessary tasks:

1. Retrieve a copy of the current OAT.
   a. Issue the TSO command `IOACMD`
   b. Select the option **Get OSA Address Table**
   c. Respond to the prompts, specifying:
      - OSA chpid address
      - Data set name to save the retrieved OAT into
2. Edit the retrieved OAT, making the required changes.
3. Reload the OAT to the OSA.
   a. Issue the TSO command `IOACMD`
   b. Select the option **Put OSA Address Table**
   c. Respond to the prompts, specifying:
      - OSA chpid address
      - Data set name of the OAT to be loaded (as in *1c*).
4. If the OAT includes a definition for one or more SNA devices you need to run the *IOAINSNA* step to download the SNA support code onto the OSA.
5. Vary the entire device range for this OSA *OFFLINE* then back *ONLINE* to activate the changes.

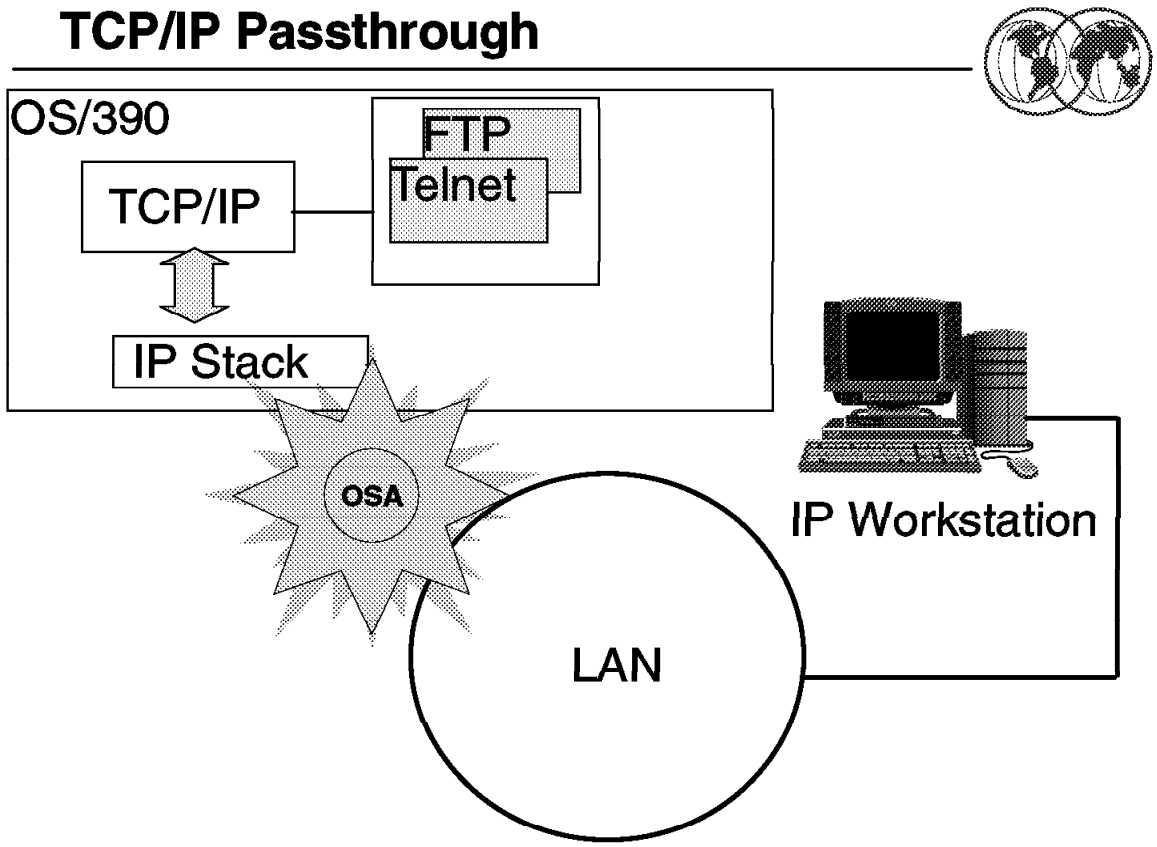


Figure 86. TCP/IP Passthrough

1.30.7 TCP/IP Passthrough

As the description implies, an OSA works as a passthrough agent for TCP/IP. TCP/IP views the OSA as a LAN Channel Station (LCS) device. The default mode of operation for an OSA is TCP/IP Passthrough.

The following examples show the OAT entries and the extended entries that define the transmit and receive device pair D04 and D05.

```
/*********************************************************************/
/* Start of OAT entry 1  ----- D04                                   */
/*********************************************************************/
oat.1.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.1.2 = All data is valid           /*   Valid data indicator     */
oat.1.3 =                             /*   Partition name           */
oat.1.4 = 0                           /* s-Partition number         */
oat.1.5 = 04                          /* s-Unit address             */
oat.1.6 = 0D04                        /*   Device number            */
oat.1.7 = 5C                          /*   Chpid                    */
oat.1.8 = 0D00                        /*   Control unit number      */
oat.1.9 = configured                  /*   Channel state            */
oat.1.10 = yes                        /*   Device accessible        */
oat.1.11 = 02                         /*   Group size               */
oat.1.12 = passthru                   /* s-Entry type. One of:      */
                                      /*     Passthru               */
                                      /*     SNA                    */
                                      /*     Unassigned             */
oat.1.13 = started and in use         /*   Entry descriptor         */
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
passthru.1.1 = 1                      /* s-Port number              */
passthru.1.2 = no                     /* s-Default LP (yes/no)      */
passthru.1.3 = 195.183.66.11          /* s-home IP address          */
```

```
/*********************************************************************/
/* Start of OAT entry 2   --- D05                                    */
/*********************************************************************/
oat.2.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.2.2 = All data is valid           /*   Valid data indicator     */
oat.2.3 =                             /*   Partition name           */
oat.2.4 = 0                           /* s-Partition number         */
oat.2.5 = 05                          /* s-Unit address             */
oat.2.6 = 0D05                        /*   Device number            */
oat.2.7 = 5C                          /*   Chpid                    */
oat.2.8 = 0D00                        /*   Control unit number      */
oat.2.9 = configured                  /*   Channel state            */
oat.2.10 = yes                        /*   Device accessible        */
oat.2.11 = 02                         /*   Group size               */
oat.2.12 = passthru                   /* s-Entry type. One of:      */
                                      /*     Passthru               */
                                      /*     SNA                    */
                                      /*     Unassigned             */
oat.2.13 = started and in use         /*   Entry descriptor         */
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
passthru.2.1 = 1                      /* s-Port number              */
passthru.2.2 = no                     /* s-Default LP (yes/no)      */
passthru.2.3 = 195.183.66.11          /* s-home IP address          */
/*********************************************************************/
```

The following table identifies the relationship between the above OAT parameters and the corresponding TCP/IP setup parameters (see 1.27.1, “Configuring TCP/IP - Profile data set” on page 119):

Table 5. Correspondence between OAT parameters and TCP/IP parameters

OAT parameter	TCP/IP parameter
oat.1.6 = *cuu*	DEVICE *devname* LCS *cuu* LINK *linkname* IBMTR 1 *devicename*
passthru.1.3 = *ip address*	HOME *ip address*

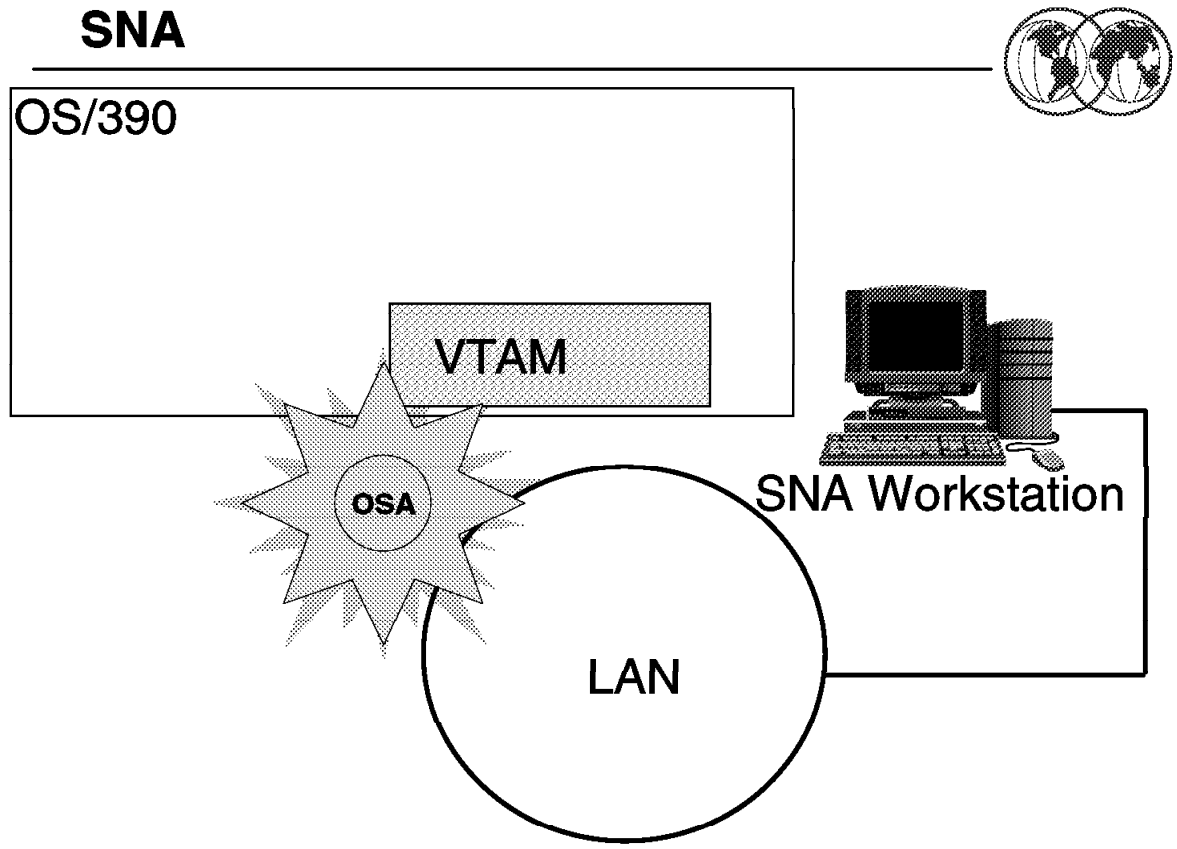


Figure 87. SNA

1.30.7.1 SNA

In order to enable SNA traffic through the OSA you must include an OAT entry and an SNA extended entry for the SNA device, as shown in the following example:

```
/*********************************************************************/
/* Start of OAT entry 3                                              */
/*********************************************************************/
oat.3.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.3.2 = All data is valid           /*   Valid data indicator     */
oat.3.3 =                             /*   Partition name           */
oat.3.4 = 0                           /* s-Partition number         */
oat.3.5 = 02                          /* s-Unit address             */
oat.3.6 = 0D02                        /*   Device number            */
oat.3.7 = 5C                          /*   Chpid                    */
oat.3.8 = 0D00                        /*   Control unit number      */
oat.3.9 = configured                  /*   Channel state            */
oat.3.10 = yes                        /*   Device accessible        */
oat.3.11 = 01                         /*   Group size               */
oat.3.12 = SNA                        /* s-Entry type. One of:      */
                                      /*     Passthru               */
                                      /*     SNA                    */
                                      /*     Unassigned             */
oat.3.13 = started and in use         /*   Entry descriptor         */
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
sna.3.1 = 00                          /* s-Port number              */
/*********************************************************************/
```

The following table identifies the relationship between the above OAT parameters and the corresponding VTAM XCA parameters (see A.2, “XCA Major Node” on page 342):

Table 6. Correspondence between OAT parameters and VTAM parameters

OAT parameter	**VTAM parameter**
oat.3.6 = *cuu*	CUADDR=*cuu* in XCA

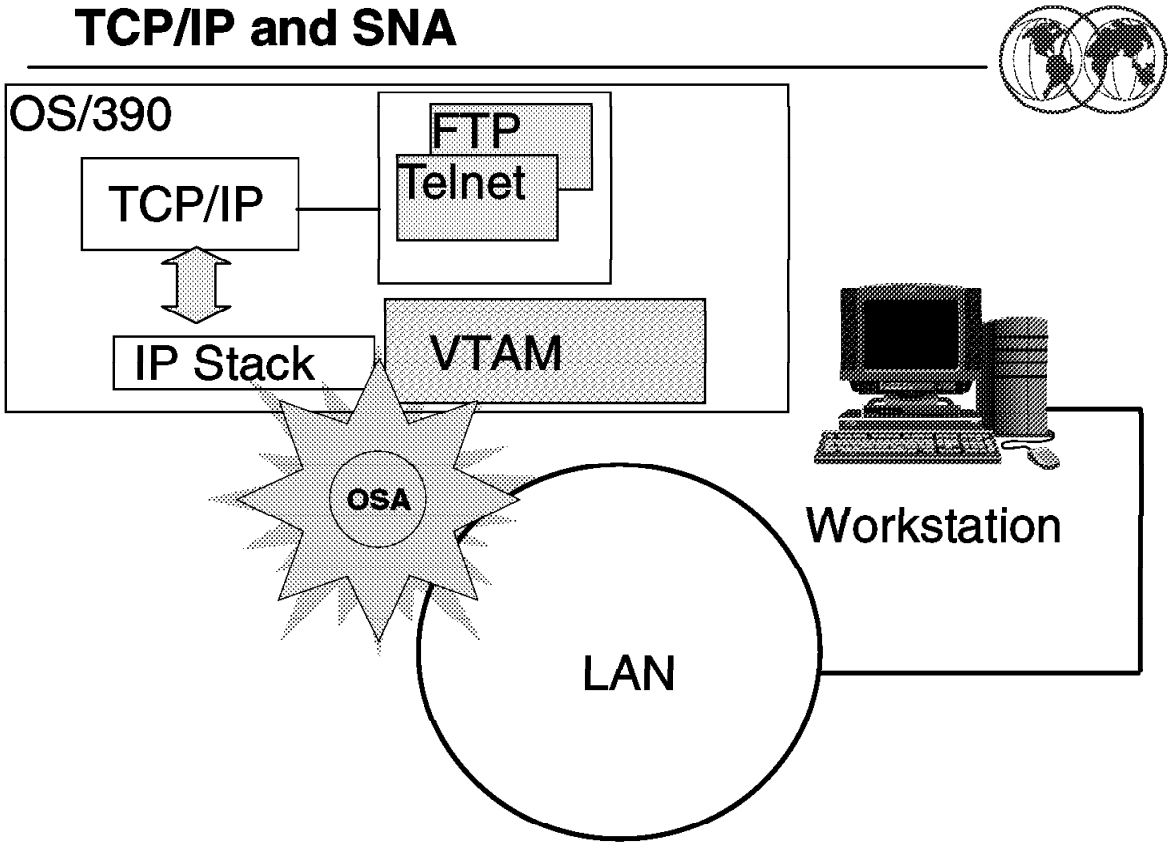


Figure 88. TCP/IP Passthrough and SNA port sharing

1.30.7.2 TCP/IP Passthrough and SNA port sharing

It is possible to configure the OSA for both SNA traffic and TCP/IP traffic through the same physical OSA port.

This is accomplished by setting the *s_Port Number* value in the extended entries for both the SNA and Passthrough devices that you want to use the same port.

Chapter 2. Security and RACF

If you require security for part or all of your installation′s database, you can use RACF to define and protect these parts. If you want to limit the users who can access certain data, and make RACF “invisible” to some users, you can define these restrictions with user attributes, group structures, and access authorities within group structures. RACF provides a very flexible approach for defining which users can use which data. The key factor is to understand what RACF functions you want to use in order to achieve your security goals.

As the number of users and the ease of use of data systems increase, the need for data security takes on new importance. An installation can no longer ignore security simply because few people know how to access the data. Installations must actively pursue and demonstrate security and use a security mechanism to control any form of access to critical data. RACF helps meet the needs for security by providing the ability to:

- Identify and verify users
- Authorize users to access the protected resources
- Control the means of access to resources
- Log and report attempts to access protected resources
- Administer security to meet an installation′s security goals

RACF provides these functions when the installation defines the users and the resources to be protected.

A specific RACF user, called the security administrator, has the responsibility to define users and resources to RACF. The security administrator sets down the guidelines that RACF uses to decide the user-resource interaction within the installation.

The responsibility to implement the guidelines falls to the system programmer, who provides technical support for RACF. The system programmer installs RACF on the system and maintains the RACF database. This person oversees the programming aspects of system protection and provides technical input on the feasibility of the implementation plan. In addition, the technical support person writes, installs, and tests RACF installation exit routines.

RACF retains information about the users, resources, and access authorities in profiles on the RACF database and refers to the profiles when deciding which users should be permitted access to protected system resources.

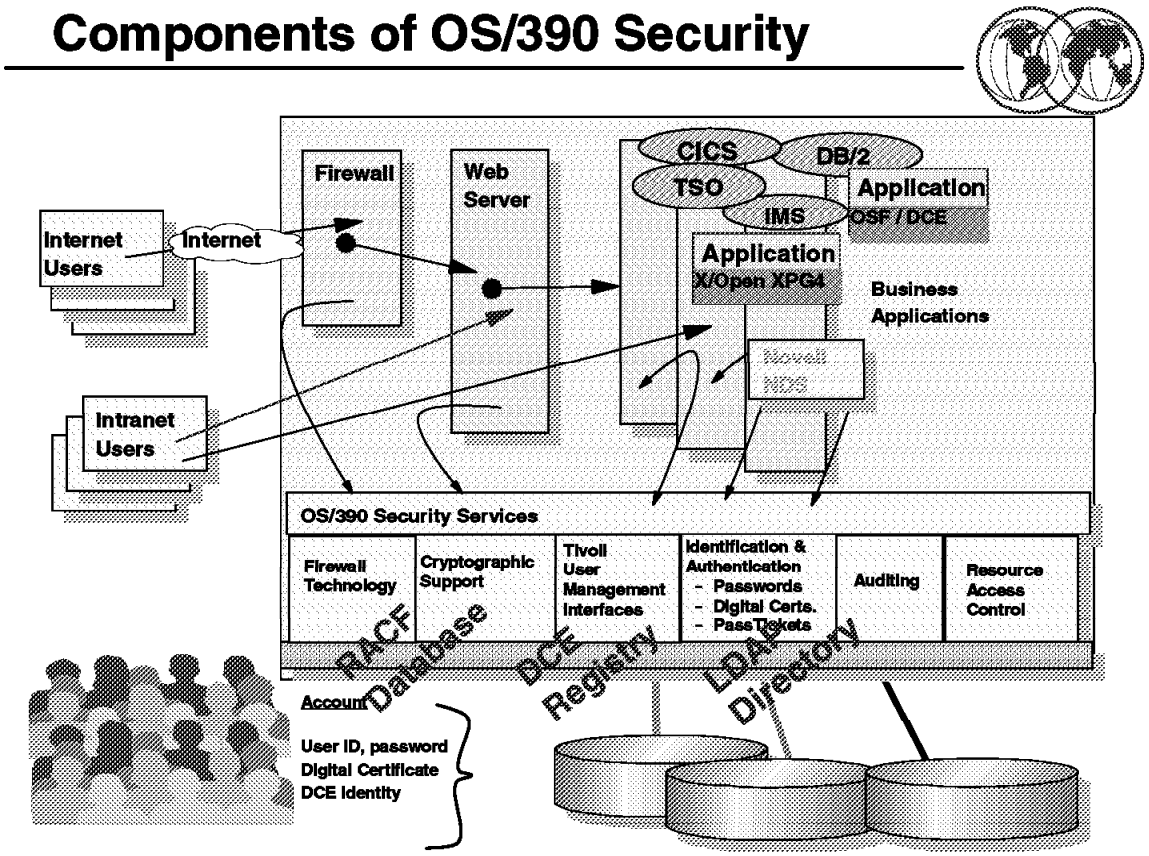


Figure 89. Components of OS/390 security

2.1 Components of OS/390 security

The OS/390 security server consists of these components:

- OS/390 Distributed Computing Environment (DCE) security server

  DCE Security Server provides user and server authentication for applications using the client-server communications technology contained in the Distributed Computing Environment for OS/390. Beginning with OS/390 Security Server Version 2 Release 5, the DCE Security Server can also interoperate with users and servers that make use of the Kerberos V5 technology developed at the Massachusetts Institute of Technology and can provide authentication based on Kerberos tickets. Through integration with RACF, OS/390 DCE support allows RACF-authenticated OS/390 users to access DCE-based resources and application servers without having to further authenticate themselves to DCE. In addition, DCE application servers can, if needed, convert a DCE-authenticated user identity into an RACF identity and then access OS/390 resources on behalf of that user, with full RACF access control.

- OS/390 Firewall Technologies

  Implemented partly in the Security Server and partly in the SecureWay Communications Server for OS/390, OS/390 Firewall Technologies provide basic firewall capabilities on the OS/390 platform to reduce or eliminate the need for non-OS/390 platform firewalls in many customer installations. The Communications Server provides the firewall functions of IP packet filtering, IP security (VPN or tunnels), and Network Address Translation (NAT). The Security Server provides the firewall functions of FTP proxy support SOCKS daemon support, logging, configuration, and administration.

- OS/390 Lightweight Directory Access Protocol (LDAP) server

  LDAP Server provides secure access from applications and systems on the network to directory information held on OS/390 using the Lightweight Directory Access Protocol.

- Resource Access Control Facility (RACF)

  The primary component of the SecureWay Security Server for OS/390 is the Resource Access Control Facility, which works closely with OS/390 to protect its vital resources. Building from a strong security base provided by the RACF component, the Security Server is able to incorporate additional components that aid in securing your system as you make your business data and applications accessible by your intranet, extranets, or the Internet.

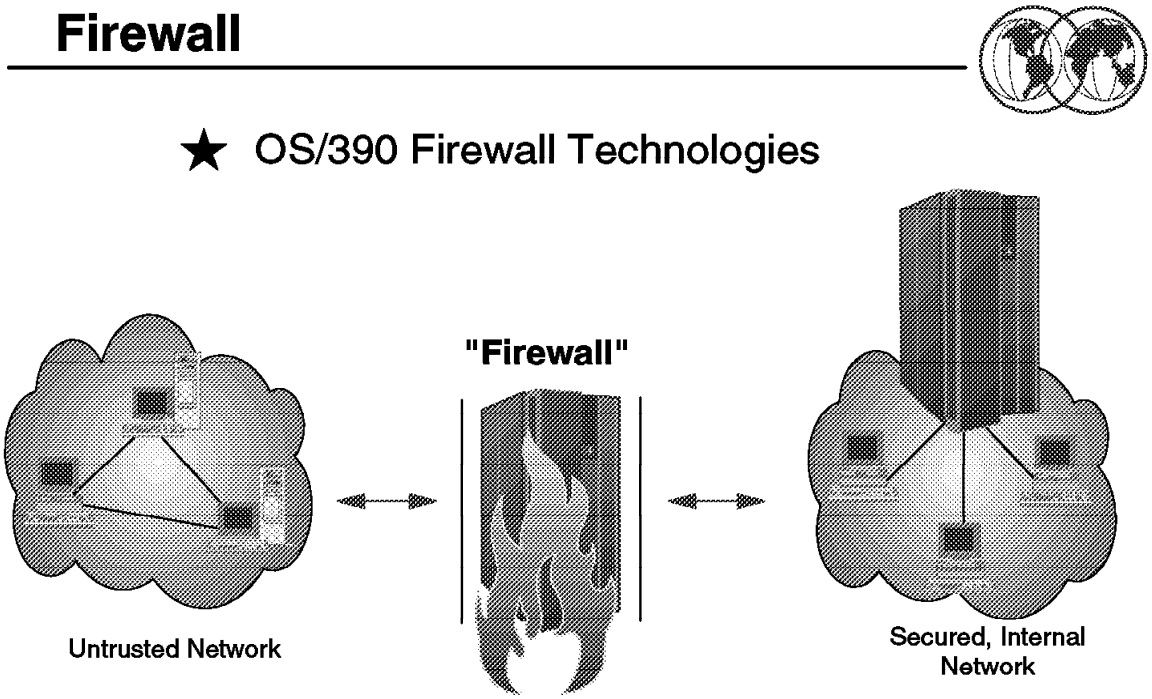


Figure 90. Firewall

2.2 OS/390 Firewall Technologies

The OS/390 Firewall Technologies functions are provided to a customer in both the eNetwork Communication Server for OS/390 and the OS/390 Security Server. The OS/390 Firewall Technologies provides a “firewall” program that does just that.

It isolates your internal network from other networks in the Internet, while at the same time letting typical TCP/IP applications (such as FTP) access hosts outside in the Internet without allowing intruders access to the secure network.

The OS/390 Firewall Technologies acts as a barrier between your network and the rest of the Internet.

Functions that are included with the eNetwork Communication Server for OS/390 are:

- IP Packet Filtering
- IP Security (tunnels)
- Network Address Translation (NAT)

These functions are included as part of the IP Security feature of OS/390 and are always available.

Functions that are included as part of the OS/390 Security Server are:

- FTP proxy support
- Socks server (daemon) support
- Enhanced Logging

- Command Line Configuration/Administration
- GUI Configuration/Administration

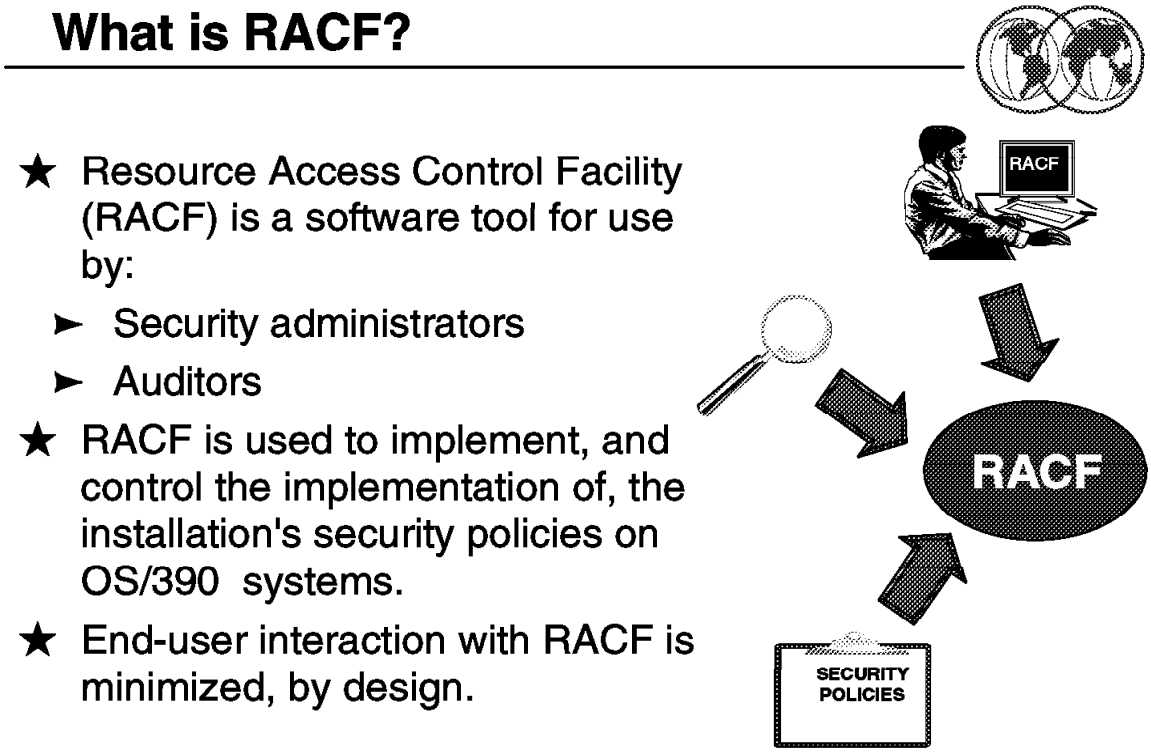


Figure 91. What is RACF?

2.3 What is RACF

RACF is a add-on software product that provides the basic security to an OS/390 system. There are other security software packages available such as Computer Associates, ACF2 or Top Secret. However, RACF is included as part of the base OS/390 system but requires a separate licence and fee.

RACF allows you to implement/reflect on your system the security policies you choose. That means that your system will not be secure by simply installing RACF; the quality of the system protection depends on the way you use the RACF functions.

With RACF each defined user belongs to at least one group, known as the *default group*. A *group* is a collection of RACF users who share common access requirements to protected resources or who have similar attributes within the system.

RACF records information about the groups in the group profile, which reside in the RACF database.

RACF allows users to be members of more than one group. A RACF user who is associated with a group is, in RACF terminology, “connected” to that group.

A group owner, usually the user who defined the group to RACF, can define control of the other users connected to the group. The group owner can also delegate various group administrative responsibilities and authorities to various users connected to the group.

Each RACF-defined resource has a profile, though an installation can optionally use a single profile to protect multiple resources.

As you can see, the fundamental structure of RACF consists of *users*, *groups*, and *resources*. RACF not only authorizes the users who can access the system, but also controls whether the user can access resources which depend on the user's purpose, (for example, reading or updating).

The "active" users of RACF are security administrators and auditors. End users of a system are "passive" users of RACF.

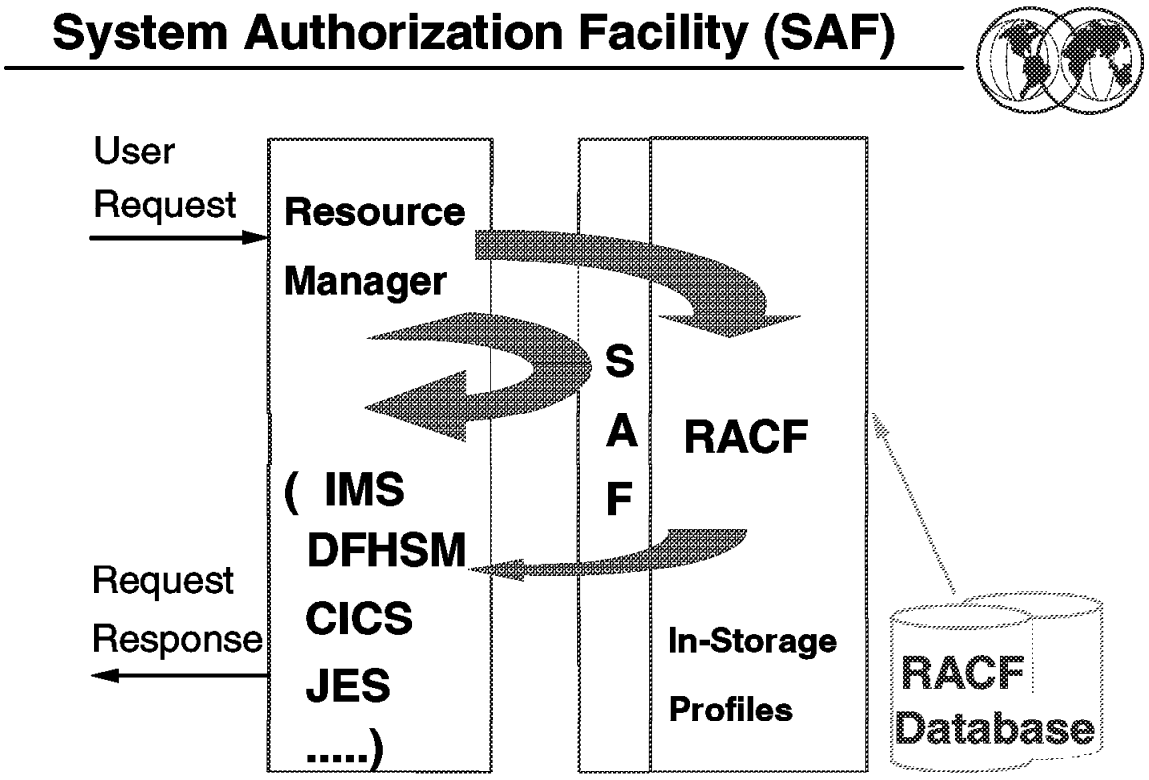


Figure 92. System Authorization Facility (SAF)

2.4 System Authorization Facility (SAF)

System Authorization Facility (SAF) is part of the operating sytem. SAF is present on all OS/390 systems and is active even when RACF is not.

SAF establishes default security functions when RACF is not active. To enable this, SAF is initialized early in the NIP process.

Resource managers are responsible for calling SAF to determine whether a user is allowed access to the system or resource. The resource manager is responsible for enforcing the decision made by SAF or RACF.

The figure illustrates the SAF function. Based on the original user's request, the resource manager formulates a request to SAF. Depending on the request, SAF may respond directly or pass the request to RACF. In either case the user receives the response from the resource manager.

Access to the RACF database may not be required if the response can be formulated from profiles that are held in storage.

2.4.1 Resource managers

Examples of resource managers are:

- OpenEdition
- DFSMS
- IMS
- CICS
- TSO
- DB2
- JES
- Console Services
- VTAM

2.4.2 Token support

SAF also creates and maintains security tokens.

A security token is an 80-(decimal) byte packet of security information that is associated to a unit of work. These tokens provide a means by which all work, including input and output, can be identified as it flows around the system.

Information contained in the token includes :

- Port of entry
- Submitting node
- User ID
- Group ID

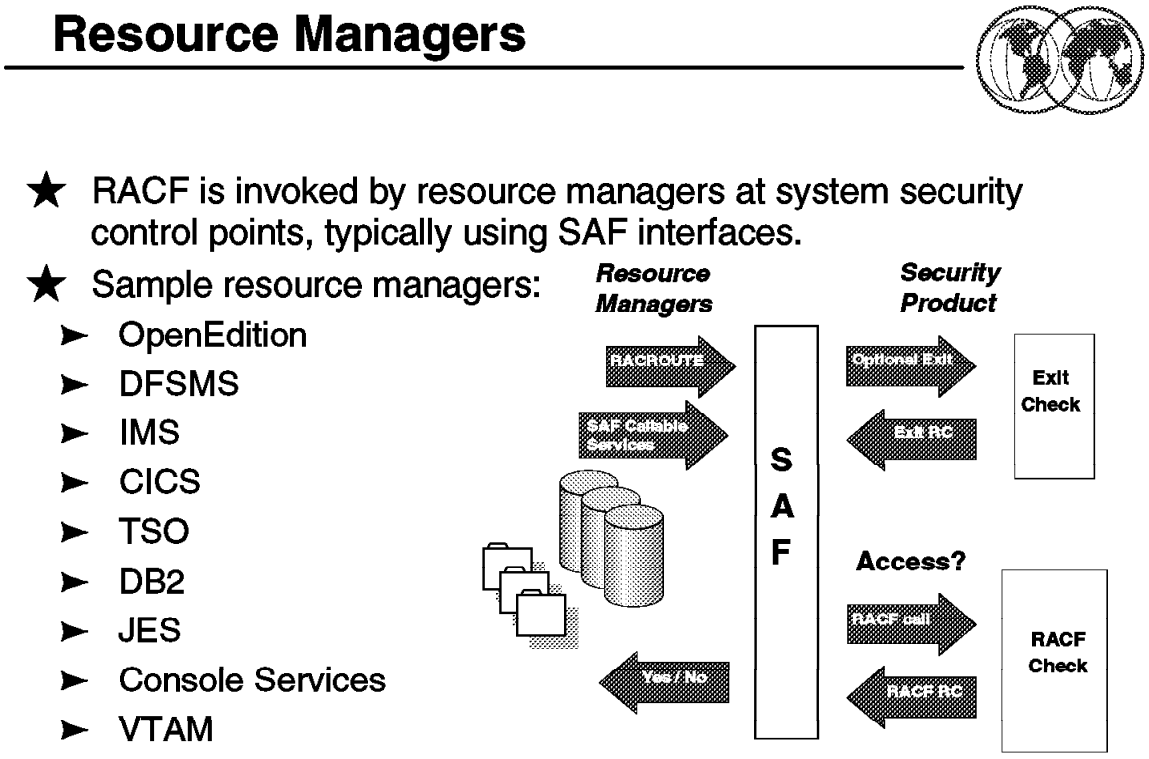


Figure 93. Resource managers

2.4.3 Resource validation overview

RACF identifies and authenticates users accessing the system when the various system resource managers (such as job initiation) request it. RACF determines:

- If the user is defined to RACF
- If the user has supplied a valid password, PassTicket, or operator identification card (OIDCARD), and a valid group name
- If the user′s UID and GID are valid on OS/390 UNIX
- If the user ID is in REVOKE status, which prevents a RACF-defined user from entering the system at all or entering the system with certain groups
- If the user can use the system on this day and at this time of the day (an installation can impose restrictions)
- If the user is authorized to access the terminal (which can also include day and time restrictions for accessing that terminal)
- If the user is authorized to access the application

After it has authenticated the user′s identity, RACF specifies the scope of the user′s authorization for the current terminal session or batch job.

After identifying and authenticating the user, RACF controls interaction between the user and the system resources. RACF must authorize:

- Which users may access resources
- How the user may access them, such as for reading or for updating

RACF can also authorize when a user can access resources, by either time or day as follows:

- A user is identified and verified to the RACF-protected system.
- A user wants to modify an existing RACF-protected resource.
- The user issues a command to the system to access the resource.
- The system resource manager (such as data management) processes the request.
- The resource manager “asks” RACF whether the user can access the resource.
- RACF checks one profile to verify that the user can access the resource and to determine whether the user has the required authorization to modify the contents.
- RACF returns the results of its check to the resource manager.
- The resource manager, based on what RACF indicates, either grants or denies the request.

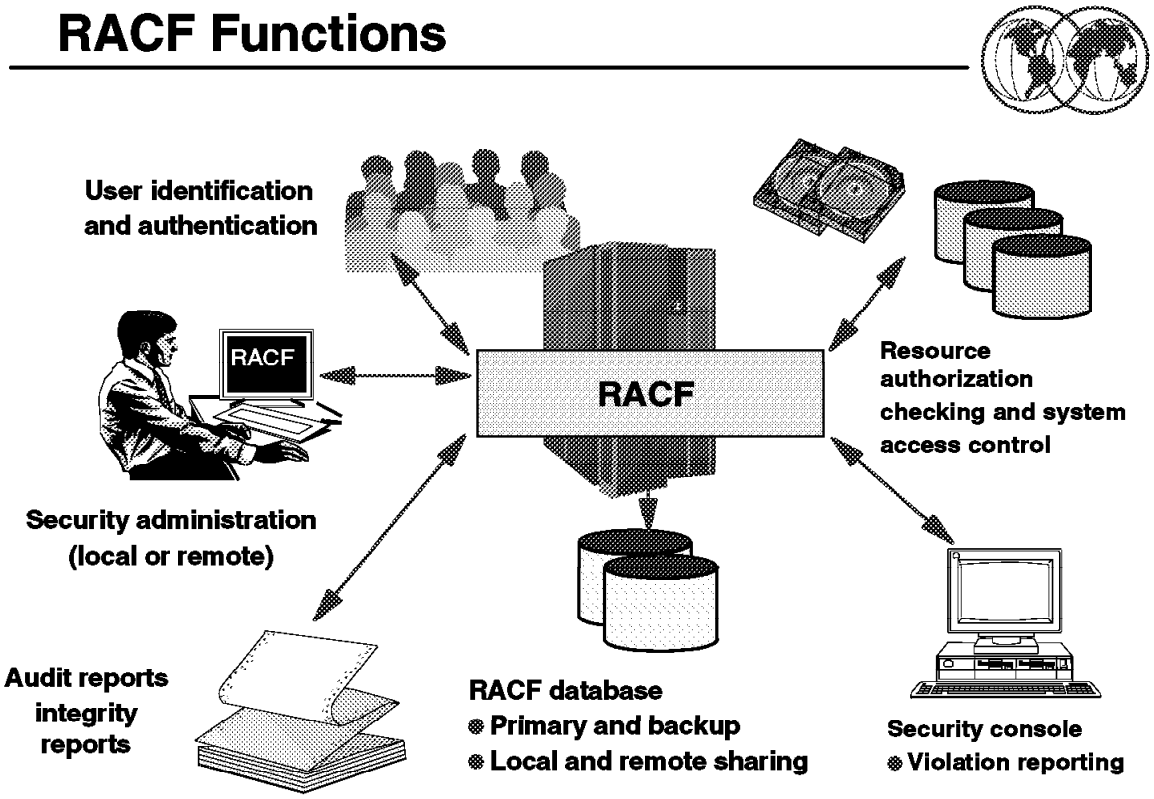


Figure 94. RACF functions

2.5 RACF functions

RACF protects resources by granting access only to authorized users of the protected resources. To accomplish this, RACF gives you the ability to:

- Identify and authenticate users

  User authentication is validation of the user requesting access. The standard approach to RACF user identification is achieved by the use of user ID and password. Other options are available, such as digital certificate and smart card.

- Security administration

  RACF can be administered either in a centralized or decentralized approach. In a centralized approach, the RACF administrator (a user who has the SPECIAL attribute) would control the access to all users.

  In a decentralized approach, RACF administration is delegated to an administrator at a group level. This administrator would have the group-SPECIAL attribute, which enables them to control access to their group only.

- Resource authorization

  Having identified and verified the user, RACF then controls interaction to the system resources. RACF must authorize not only the users who may access resources, but also the way the user may access them, which depends on the user′s purpose (for example, reading or updating). RACF can also authorize when a user can access resources, by either time or day.

- Log and report various attempts of unauthorized access to protected resources

  RACF provides two types of reporting functions. It gives immediate or “real time” notification of security events, as well as providing a data logging function.

- Control the means of access to resources

  RACF retains information about the users, resources, and access authorities in profiles on the RACF database and refers to the profiles when deciding which users should be permitted access to protected system resources. RACF allows applications to use RACF macros.

- RACF database

  The RACF database holds all RACF access control information. RACF processing uses the information from the database each time a RACF-defined user enters a system and each time a user wants to access a RACF-protected resource

  You maintain your RACF database through commands, macros, and utilities.

  The RACF database is a non-VSAM single extent data set that is made up of 4 KB blocks and must be cataloged.

  RACF allows you to provide a backup database to which you can switch without a re-IPL should your primary RACF database fail. A backup RACF database reflects the contents of the primary database. Once the installation has created the backup database, RACF can maintain it automatically.

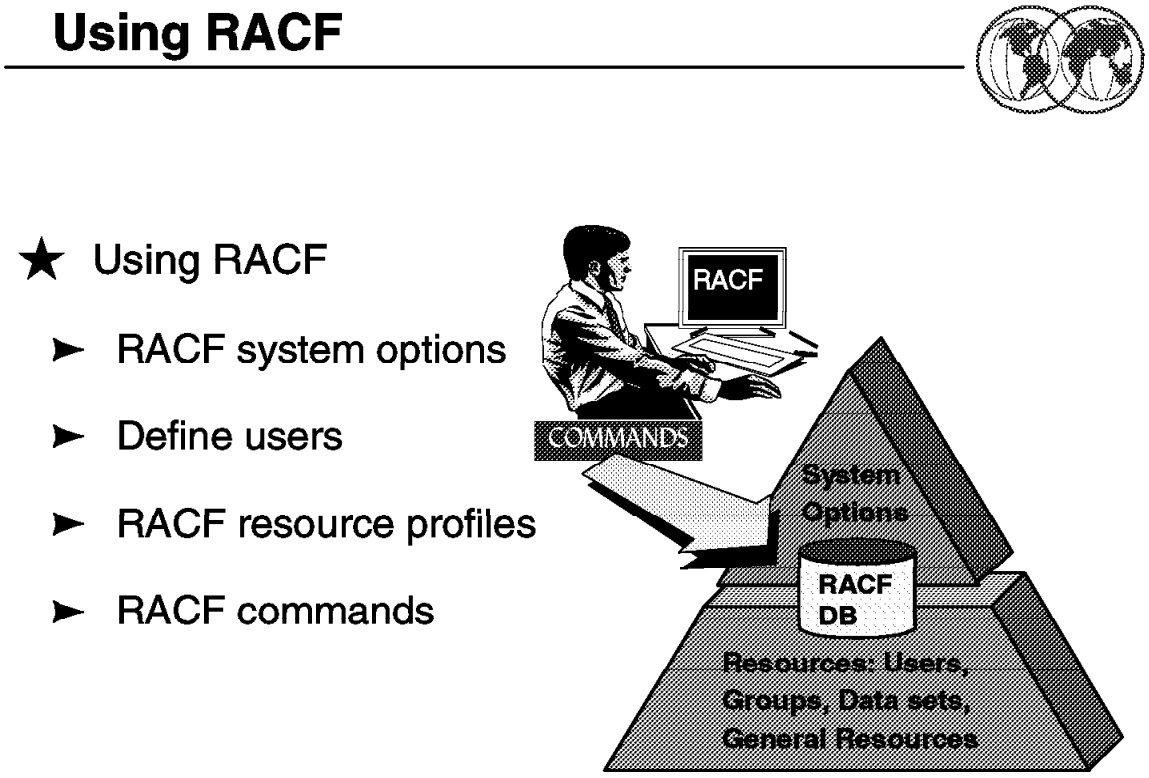


Figure 95. Using RACF

2.6 Using RACF

RACF gives the user defined with the SPECIAL attribute (the security administrator) many responsibilities both at the system level and at the group level. As the security administrator, you are the focal point for planning security at your installation. You need to:

- Determine which RACF functions to use
- Identify the level of RACF protection
- Identify which data RACF is to protect
- Identify administrative structures

A RACF system administrator should be assigned to:

- Identify users and assign attributes

  Identifying and defining user and group relationships make it simpler and more efficient to protect resources that those users and groups create, share, or use. In instances where some groups require exceptional access controls, you might subdivide your organization to minimize occasions when data needs to be passed between these groups and the rest of the organization. If the users in a group share common access requirements, as is often the case, the administrative task of authorizing users is greatly simplified.

- Define RACF system options

  The key factor is to understand what RACF functions you want to use in order to achieve your security goals. The following list shows some RACF functions that you might use and relates these

functions to the security they provide; Data Set Protection, Resource Protection, Naming Conventions, Organization, Group Names, Transparency, RACF Tailoring, Recovery, Violation Detection, Subsystems, Networks, and Data Sharing.

- Define RACF resource profiles

  RACF maintains information entries called *profiles* in the RACF database. It uses them to protect DASD and tape data sets and general resources, such as tape volumes and terminals.

  - Data set profiles contain security information about DASD and tape data sets.
  - General resource profiles contain security information about general resources.

  Each RACF-defined resource has a profile, though you can optionally use a single profile to protect multiple resources.

- Understand RACF commands

  The RACF operator commands allow you to perform functions in the RACF subsystem and can be entered from an operator console. These commands allow an MVS operator to perform certain RACF operations in the RACF subsystem. The RACF subsystem prefix in front of the command identifies the RACF subsystem as the processing environment. Many RACF commands can be entered via TSO/E.

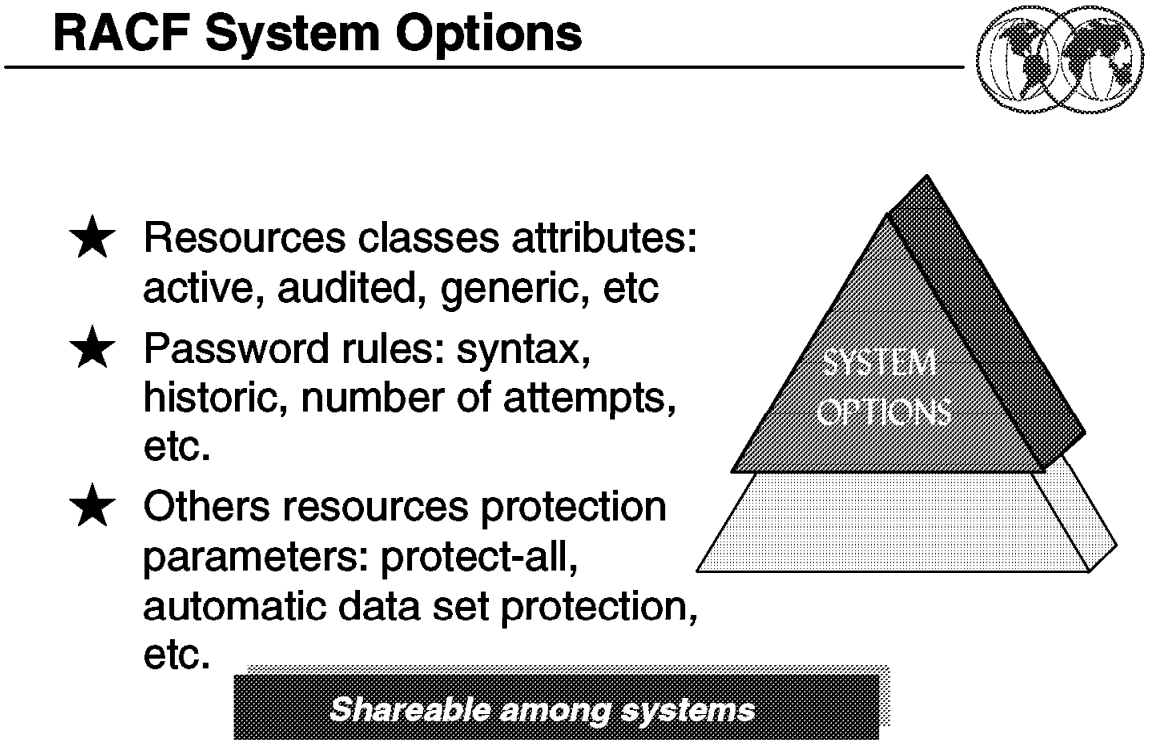


Figure 96. System options

2.6.1 System options

The SETROPTS command is used to set or change system-wide settings within RACF. This command allows settings to be updated dynamically by use of the refresh option.

If your installation has activated SETROPTS GENLIST processing for a particular resource class, you must refresh in-storage profiles for this processing when you make changes to one of these profiles in the database. Refreshing profiles for SETROPTS GENLIST processing ensures that the most current copy of a profile resides in common storage and is available for RACF authorization checking. To refresh profiles for this processing, issue the SETROPTS command with the GENERIC and REFRESH operands and specify the appropriate resource classes.

Some examples of system-wide settings are:

- Establishing Password Syntax Rules (PASSWORD Option)
- Setting the Maximum Password Change Interval (PASSWORD Option)
- Extending Password and User ID Processing (PASSWORD Option)
- Revoking Unused User IDs (INACTIVE Option)
- Activating List-of-Groups Checking (GRPLIST Option)
- Setting the RVARY Passwords (RVARYPW Option)
- Restricting the Creation of General Resource Profiles (GENERICOWNER Option)
- Activating General Resource Classes (CLASSACT Option)

- Activating Generic Profile Checking and Generic Command Processing
- Activating Statistics Collection (STATISTICS Option)
- Activating Global Access Checking (GLOBAL Option)
- RACF-Protecting All Data Sets (PROTECTALL Option)
- Activating JES2 or JES3 RACF Support
- Preventing Access to Uncataloged Data Sets (CATDSNS Option)
- Activating Enhanced Generic Naming for the DATASET Class (EGN Option)
- Controlling Data Set Modeling (MODEL Option)
- Bypassing Automatic Data Set Protection (NOADSP Option)
- Displaying and Logging Real Data Set Names (REALDSN Option)
- Protecting Data Sets with Single-Qualifier Names (PREFIX Option)
- Activating Tape Data Set Protection (TAPEDSN Option)
- Activating Tape Volume Protection (CLASSACT(TAPEVOL) Option)
- Establishing a Security Retention Period for Tape Data Sets (RETPD Option)
- Erasing Scratched or Released DASD Data (ERASE Option)
- Establishing National Language Defaults (LANGUAGE Option)

RACF System Options

★ SETROPTS LIST example:

```
ATTRIBUTES = INITSTATS WHEN(PROGRAM) SAUDIT CMDVIOL OPERAUDIT
STATISTICS = NONE
AUDIT CLASSES = DATASET USER GROUP DASDVOL GDASDVOL TAPEVOL DSNR
ACTIVE CLASSES = DATASET USER GROUP DASDVOL GDASDVOL TAPEVOL TIMS GIMS
                 AIMS DSNR TCICSTRN GCICSTRN PCICSPSB QCICSPSB FACILITY
GENERIC PROFILE CLASSES =  DATASET DASDVOL TAPEVOL DSNR FACILITY OPERCMDS
AUTOMATIC DATASET PROTECTION IS NOT IN EFFECT
PROTECT-ALL IS ACTIVE, CURRENT OPTIONS:
   PROTECT-ALL WARNING OPTION IS IN EFFECT
TAPE DATA SET PROTECTION IS ACTIVE
INACTIVE USERIDS ARE NOT BEING AUTOMATICALLY REVOKED.
PASSWORD PROCESSING OPTIONS:
  PASSWORD CHANGE INTERVAL IS  62 DAYS.
   6 GENERATIONS OF PREVIOUS PASSWORDS BEING MAINTAINED.
  AFTER   4 CONSECUTIVE UNSUCCESSFUL PASSWORD ATTEMPTS,
      A USERID WILL BE REVOKED.
  PASSWORD EXPIRATION WARNING LEVEL IS   7 DAYS.
  INSTALLATION PASSWORD SYNTAX RULES:
    RULE 1  LENGTH(6:8)   ********
   LEGEND:
    A-ALPHA C-CONSONANT L-ALPHANUM N-NUMERIC V-VOWEL W-NOVOWEL *-ANYTHING
```

Figure 97. Display RACF system options

2.6.2 SETROPTS LIST command

This command specifies that the current RACF options are to be displayed. If you specify operands in addition to LIST on the SETROPTS command, RACF processes the other operands before it displays the current set of options.

If RACF is enabled for sysplex communication and the system is in read-only mode, users on that system can issue the SETROPTS LIST command. All other operands will be ignored.

You must have the SPECIAL, AUDITOR, group-SPECIAL, or group-AUDITOR attribute to enter the LIST operand.

If you have the SPECIAL or group-SPECIAL attribute, RACF displays all operands except these auditing operands:

The visual shows sample output from the following SETROPTS command:

```
SETROPTS LIST
```

Another example on how to use the SETROPTS command is:

```
SETROPTS PASSWORD(INTERVAL(60))
```

The INTERVAL suboperand specifies the system default for the number of days that the user's password is to remain valid. The example specifies that each user's password remain valid for 60 days.

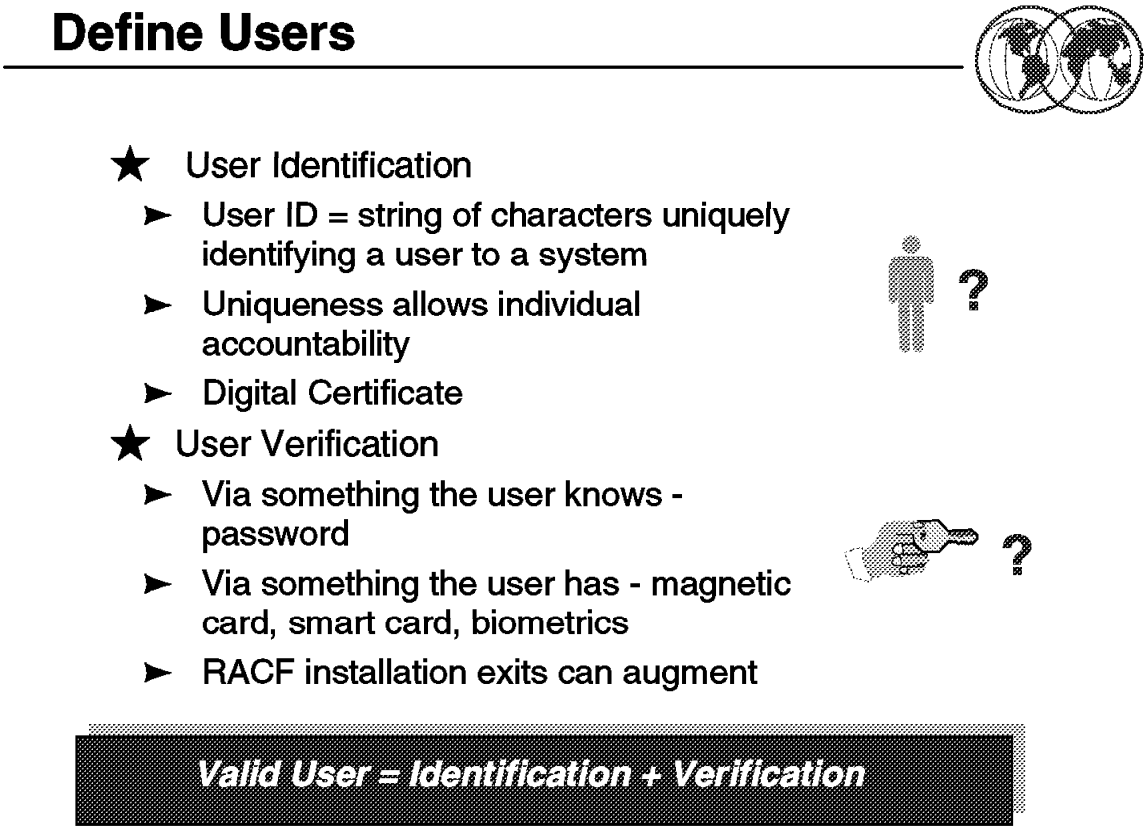


Figure 98. Define users

2.6.3 Define users

As a general objective, all users should be defined to RACF. Users who are not defined to RACF can use the system virtually without verification, unless, of course, they attempt to access data to which they are unauthorized.

You should consider defining the following users to RACF:

- Interactive users of CICS, IMS, TSO/E, NetView, or other products that support logging on at a terminal.
- Users who submit batch jobs
- MVS or JES system operators
- Started procedures
- Node names in an NJE network
- RJP or RJE remote workstations or nodes

In a client-server network environment, entities identify themselves using digital certificates. The combination of a serial number and the name of the certificate authority (or issuer's distinguished name) uniquely identifies a client's digital certificate.

RACF uses a user ID and a system-encrypted password to perform its user identification and verification. When you define a user to RACF, you assign a user ID and temporary password. The user ID identifies the person to the system as a RACF user. The password verifies the user's identity.

Verification can be done with the use of a card with a magnetic stripe encoded with unique characters and used to verify the identity of a user to RACF on an OS/390 system.

You define users to RACF by issuing RACF commands that include various user attributes, as well as other control information that RACF uses. Following are some of the commands you might use in your user-definition tasks:

ADDUSER Add a user profile to RACF.

ALTUSER Change a user's RACF profile.

CONNECT Connect a user to a group.

DELUSER Delete a user profile from RACF and remove connection to a group.

REMOVE Remove a user from a group and assign a new owner for group data sets owned by the removed user.

LISTUSER Display the contents of a user's profile.

PERMIT Permit a user to access a resource (or deny access to a resource).

PASSWORD Change a user's password.

In addition to defining individual users, you can define groups of users. Group members can share common access authorities to a protected resource.

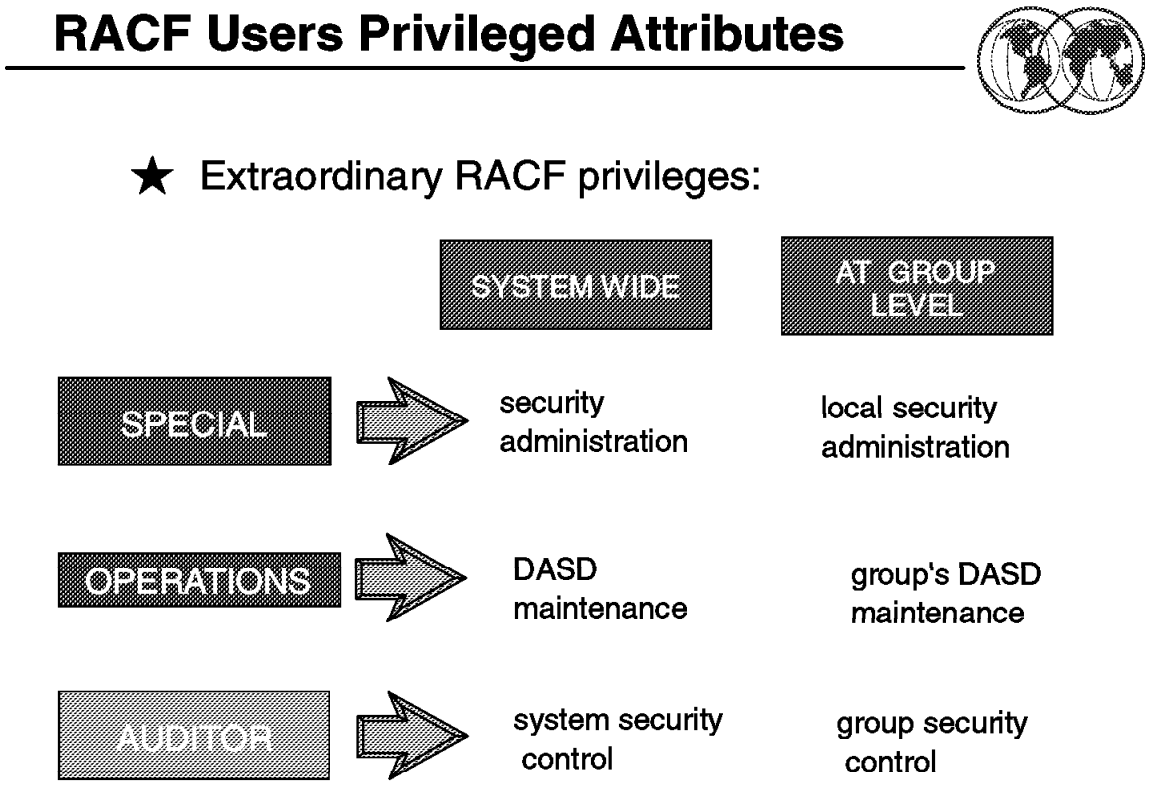


Figure 99. RACF user privileged attributes

2.6.4 User attributes

User attributes are extraordinary capabilities, restrictions, or environments that can be assigned to a user, either all of the time or when the user is connected to a specific group or groups. When an attribute is to apply all of the time, it is specified at the system level and is called a user attribute. When an attribute is to apply only to a specific group or groups, it is specified at the group level and is called a group-related user attribute. For example, user attributes that you specify in an ADDUSER or ALTUSER command are stored in the user′s profile and are in effect regardless of the group to which the user is connected.

The user attributes are as follows:

SPECIAL A user who has the SPECIAL attribute at the system level can issue all RACF commands. The SPECIAL attribute gives the user full control over all of the RACF profiles in the RACF database.

You can assign the SPECIAL attribute at the group level. When you do, the group-SPECIAL user has full control over all of the profiles within the scope of the group.

AUDITOR AUDITOR attribute is given only to users who are responsible for auditing RACF security controls and functions. To provide a check and balance on RACF security measures, you should give the AUDITOR attribute to security or group administrators other than those who have the SPECIAL attribute.

OPERATIONS A user who has the system-OPERATIONS attribute has full access authorization to all RACF-protected resources in the DATASET, DASDVOL, GDASDVOL, PSFMPL, TAPEVOL, VMBATCH, VMCMD, VMMDISK, VMNODE, and VMRDR classes.

CLAUTH Users receive the CLAUTH attribute on a class-by-class basis. You cannot assign the CLAUTH attribute at the user or group level.

If a user has the CLAUTH attribute in a class, RACF allows the user to define profiles in that class.

REVOKE You can prevent a RACF user from entering the system by assigning the REVOKE attribute. This attribute is useful when you want to prevent a user from entering the system, but you cannot use the DELUSER command because the user still owns RACF resource profiles.

GRPACC If a user has the GRPACC (Group Access) attribute, any group data set profiles that the user defines to RACF (through either the ADSP attribute, the PROTECT parameter on the DD statement, or the ADDSD command) are automatically made accessible to other users in the group if the user defining the profile is a member of that group.

ADSP When a user has the Automatic Data Set Protection (ADSP) Attribute, RACF always automatically creates a discrete profile every time the user defines a permanent DASD or tape data set.

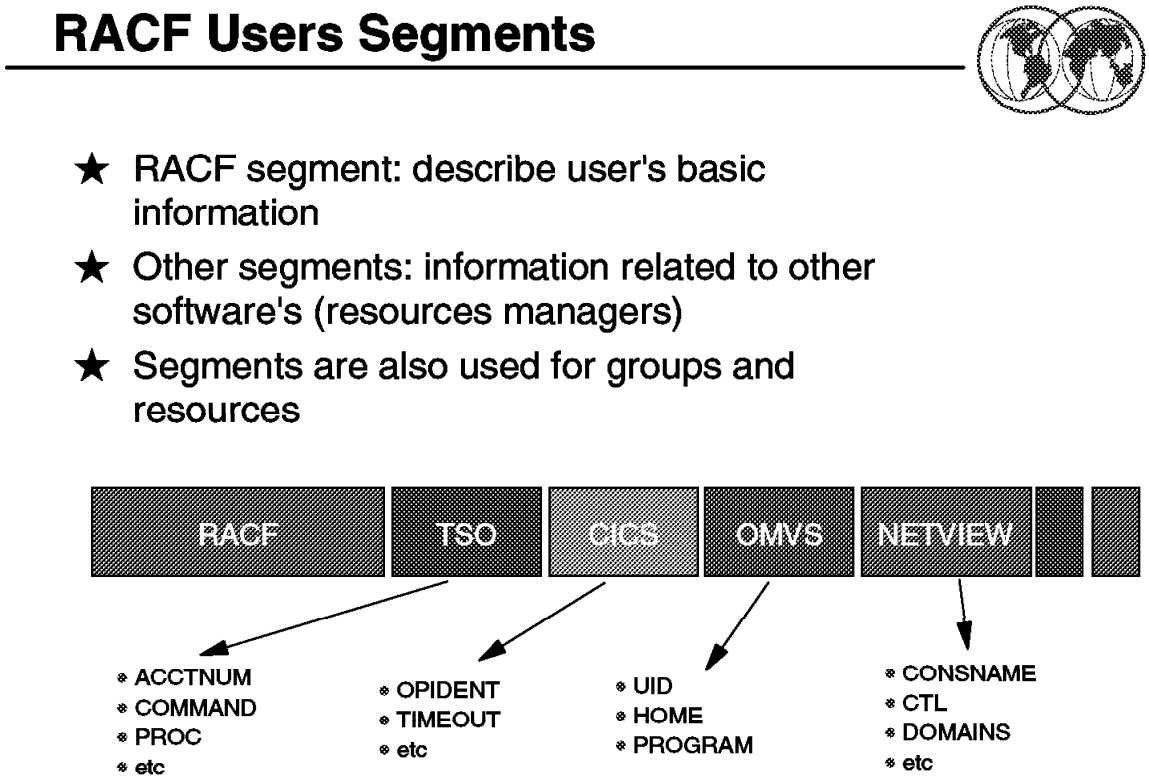


Figure 100. RACF user segments

2.6.5 RACF user segments

When you define a user to RACF, you create a user profile in the RACF database. A user profile consists of a RACF segment and, optionally, any of the following segments: CICS, DCE, DFP, LANGUAGE, LNOTES, NDS, NETVIEW, OMVS, OPERPARM, OVM, TSO, and WORKATTR.

The RACF segment is the portion of a RACF profile that contains the fundamental information about a user, group, or resource and is common to several applications. It is also called the base segment.

The other segments allow resource managers to store related information (not always/only security-related information).

The number of resource managers using RACF for their protection is continuously growing.

You can specify the following information in the RACF segment of the user profile:

USERID User′s identification

NAME User′s name

OWNER Owner of the user′s profile

DFLTGRP User′s default group

AUTHORITY User′s authority in the default group

PASSWORD User′s password

REVOKE	Date on which RACF prevents the user from having access to the system
RESUME	Date on which RACF lets the user have access to the system again
UACC	Default universal access authority for resources that the user defines
WHEN	Days of the week and hours of the day during which the user has access to the system
ADDCATEGORY	User's installation-defined security category
SECLEVEL	User's installation-defined security level
CLAUTH	Classes in which the user can define profiles
SPECIAL	Gives the user the system-wide SPECIAL attribute
AUDITOR	Gives the user the system-wide AUDITOR attribute
OPERATIONS	Gives the user the system-wide OPERATIONS attribute
DATA	Installation-defined data
ADSP	Indicates that all permanent data sets the user creates are to be RACF-protected with discrete profiles
GRPACC	Indicates that other group members can have access to any group data set the user protects with a data set profile
MODEL	Name of the data set model profile to be used when creating new data set profiles, either generic or discrete
OIDCARD	Indicates that the user must supply an operation ID card when logging on to the system
SECLABEL	User's default security label
CERTNAME	The names of the profiles in the DIGTCERT class that are related this RACF user ID
CERTLABL	The certificate labels associated with the profiles in the DIGTCERT class that are related to this RACF user ID

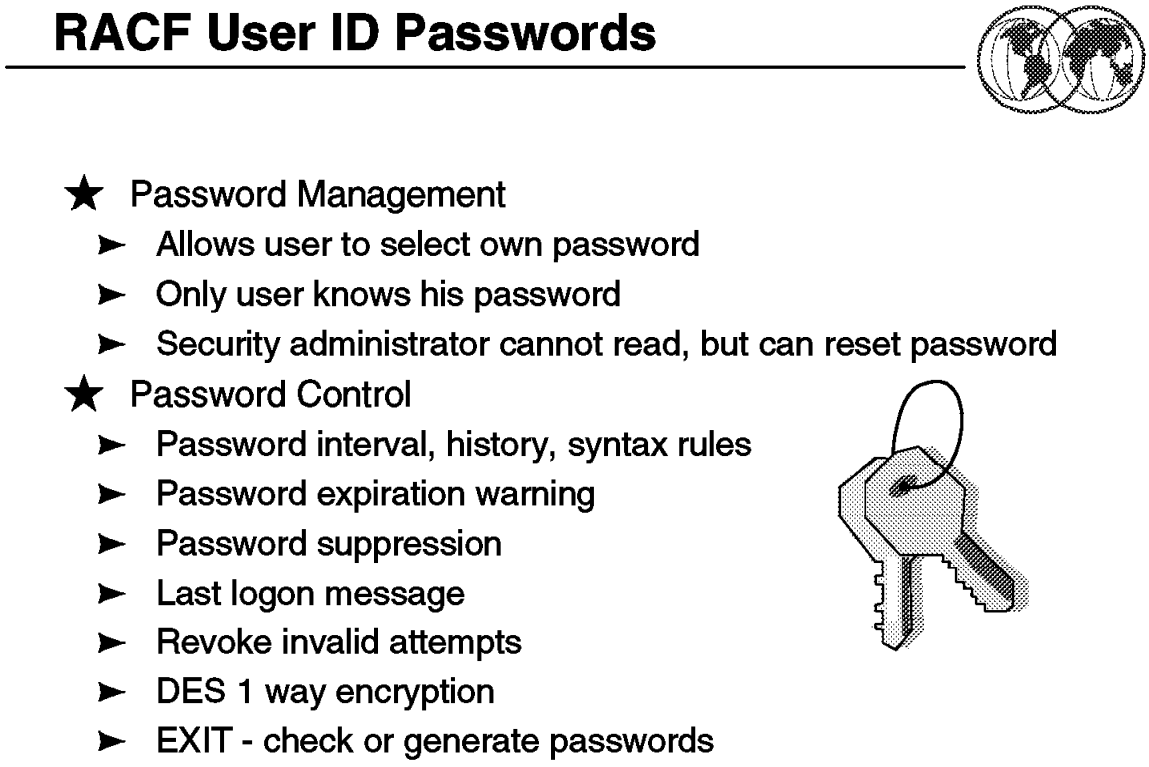


Figure 101. User RACF user ID passwords

2.6.6 RACF user ID passwords

User identification is achieved via the user ID. This is a string of characters that uniquely identify a user to a system.

In RACF the user selects his own password and only the user knows their own password. If a password needs to be reset, the security administrator will reset the password. This new password will be in an expired state, thus forcing the user to enter a new password on the first logon.

You can set a variety of rules for forming valid passwords, and this is done via the SETROPS command (for system-wide setings) or with the password command (to affect only one user). You can change such things as the number of days a password is valid for; how long to maintain password history to prevent the user from reusing the same password again; and so on.

When a user changes a password, RACF treats the new, user-supplied password as an encryption key to transform the RACF user ID into an encoded form using the DES algorithm that it stores on the database. The password is not stored.

2.6.6.1 Alternatives to password verification

1. RACF allows workstations and client machines in a client-server environment to use a PassTicket in place of a password. A PassTicket can be generated by RACF or by another authorized function, and can be used only once on a given computer system, within ten minutes of generation.
2. RACF allows the use of an operator identification card (OIDCARD) in place of, or in addition to, the password during terminal processing. By requiring that a person not only know a password but also furnish an OIDCARD, an installation has increased assurance that the user ID has been entered by the proper user.
3. OS/390 UNIX users are also identified with numeric user identifiers (UIDs), and OS/390 UNIX groups are identified with numeric group identifiers (GIDs). Unlike user names or group names, these numeric IDs can be shared by more than than one user but is not recommended.
4. In a client/server environment, RACF can identify a RACF user ID by extracting information from the digital certificate. A digital certificate or digital ID, issued by a certifying authority, contains information that uniquely identifies the client.
5. The Lotus Domino Go Webserver authenticates a client using the client's certificate and the Secure Sockets Layer (SSL) protocol. Domino Go Webserver passes the client's digital certificate to OS/390 UNIX for validation. OS/390 UNIX passes the certificate to RACF. This means that the RACF user ID and password of each client do not need to be supplied when accessing secure Web pages.

RACF ISPF Panel

```
                          RACF - SERVICES OPTION MENU
OPTION ===> 1

SELECT ONE OF THE FOLLOWING:

   1  DATA SET PROFILES

   2  GENERAL RESOURCE PROFILES

   3  GROUP PROFILES AND USER-TO-GROUP CONNECTIONS

   4  USER PROFILES AND YOUR OWN PASSWORD

   5  SYSTEM OPTIONS

   6  REMOTE SHARING FACILITY

   7  DIGITAL CERTIFICATES AND KEY RINGS
  99  EXIT
```

Figure 102. How to use RACF ISPF panels

2.7 How to use RACF ISPF panels

If your installation has installed the RACF panels, you can use them to perform security tasks.

To get to the RACF panels, enter the command: ISPF

The Interactive System Productivity Facility (ISPF) primary menu appears; choose option **R** for RACF.

Note: Although this is the usual way to access RACF panels, your installation may have this implemented via a different path.

The RACF panel interface is similar in use to all other ISPF panel options, therefore we do not go into detail here on to how to use it.

You can access help information for the RACF panels. Help panels exist for each individual panel. If you have a question about the information you should provide on the panel,either press the PF1 key or type HELP on the command line. The help panels give more information about the terms on the panel and the information you should enter.

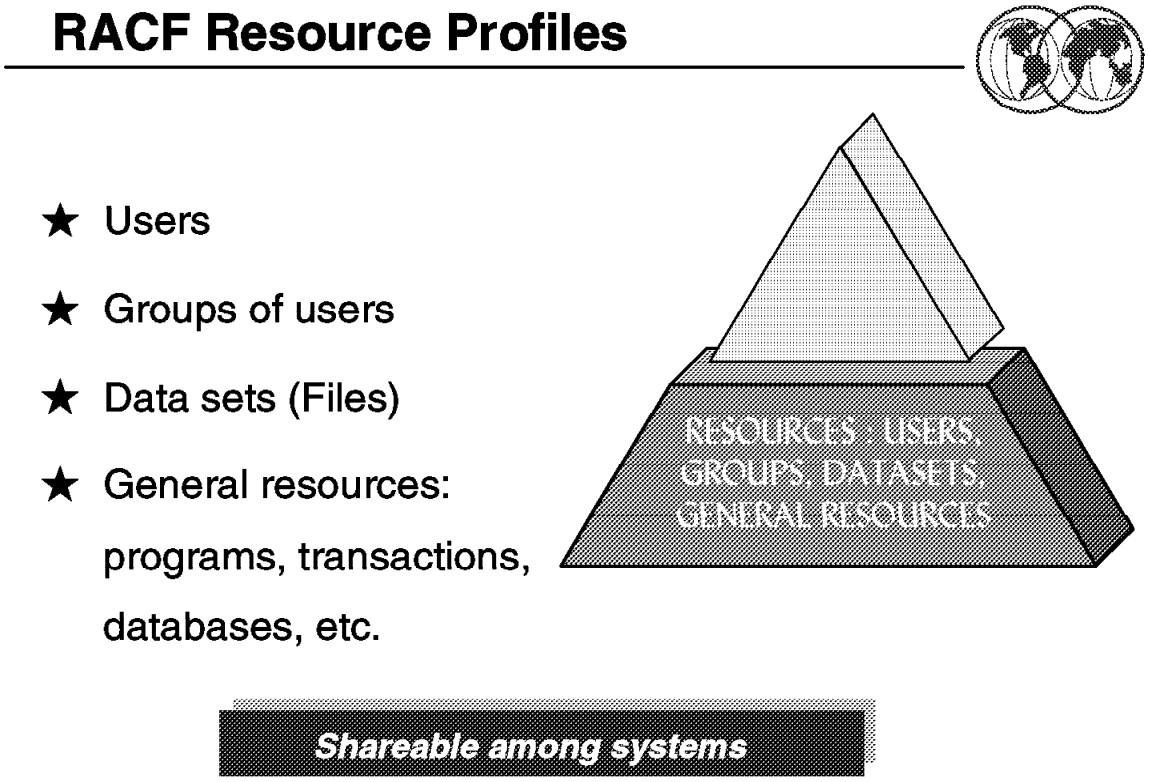


Figure 103. RACF resource profiles

2.7.1 RACF resource profiles

RACF-protected resources can be divided into two categories: data sets and general resources. General resources are all of the resources that are defined in the class descriptor table. For example, general resources include DASD and tape volumes, load modules (programs), terminals, and others.

RACF maintains information entries called profiles in the RACF database. It uses them to protect DASD and tape data sets and general resources, such as tape volumes and terminals.

- Data set profiles contain security information about DASD and tape data sets.
- General resource profiles contain security information about general resources.

Each RACF-defined resource has a profile, though you can optionally use single profile to protect multiple resources.

RACF commands or the RACF ISPF panels can be used to create and modify general resource profiles.

RACF provides discrete, generic, and grouped resource profiles for both data sets and general resources, as follows:

Discrete Discrete profiles have a one-for-one relationship with a resource; one profile for each resource. Discrete profiles provide very specific levels of control and should be used for sensitive resources. They protect only the one identified data set that is on the specified volume or that spans specific volumes. For example, a single data set can be defined with a discrete profile to allow access by one user.

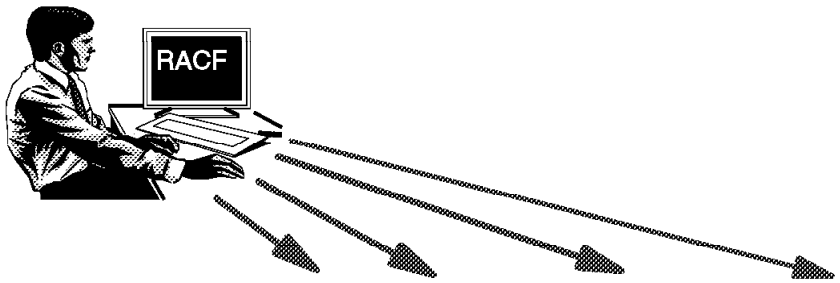
Generic Generic profiles have a one-for-many relationship. One profile controls access to one or more resources whose names contain patterns or character strings that RACF uses to associate them with each other. They contain a list of the authorized users and the access authority of each user. A single generic profile can protect many data sets that have a similar naming structure. For example, all data sets that have a high-level qualifier of SMITH and the characters DATA as a second-level qualifier can be controlled with one generic profile.

Grouped Another type of RACF profile is the grouped profile. There may be no way to associate the resources with a common access list based on patterns in the resource names. In this case, the many resource names can be associated with a single RACF profile through the use of a grouping profile that contains the names of the associated resources.

Some subsystems with high performance requirements, such as IMS/ESA, have the profiles resident in the subsystem address space. These subsystems can save main storage by using grouped profiles.

RACF Commands

★ For resources administration:



FUNCTION	USER	GROUP	DATASET	GENERAL RESOURCE
DEFINE	ADDUSER	ADDGROUP	ADDSD	RDEFINE
ALTER	ALTUSER	ALTGROUP	ALTDSD	RALTER
LIST	LISTUSER	LISTGROUP	LISTDSD	RLIST
DELETE	DELUSER	DELGROUP	DELDSD	RDELETE

Figure 104. RACF commands

2.8 RACF commands

For each resource type, a set of commands is available to define, modify, list, and delete resources.

There are several ways to enter RACF commands:

- RACF TSO commands

  This is easy and appropriate for ad hoc displays and update of user profiles and data set profiles, for example:

  ```
  RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)

  PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(JANE) ACCESS(READ)
  ```

- RACF TSO commands in batch

  This is most appropriate for a set of displays that is run, unchanged, at regular intervals.

- RACF ISPF panels

  These may be most appropriate for display of some of the more complex RACF general resource profiles. They are also very useful if you do not know the syntax for a particular command.

In general, you must have authority for a RACF entry in order to display it. A normal TSO user can display only the RACF data relevant to himself. A user with SPECIAL authority can display almost anything.

Note: We say *almost* because RACF has another authority named AUDITOR who can uniquely display certain statistical data. A SPECIAL user can create AUDITOR authority, so the SPECIAL user remains the ultimate controller of RACF.

2.8.1.1 Using RACF commands with TSO/E

You can enter RACF TSO commands from the ready prompt or from Option 6 Command from the ISPF menu.

You can get online help for RACF commands. To get online help for a command, type:

```
HELP command-name
```

For example, to see on-line help for the PERMIT command, enter:

```
HELP PERMIT
```

To limit the information displayed, use the SYNTAX operand on the HELP command:

```
HELP command-name SYNTAX
```

For example, to see only the syntax of the PERMIT command, enter:

```
HELP PERMIT SYNTAX
```

Here is a list of general use RACF commands:

PASSWORD Change password/interval

CONNECT Associate user with group

REMOVE Disassociate user from group

PERMIT Modify resource profile access list

SEARCH Locate RACF information

SETROPTS Set/modify RACF system options

RVARY Switch RACF databases

Abbreviations can be used for commands and parameters :

- AU for ADDUSER
- LG for LISTGROUP
- CO for CONNECT
- ID for USERID
- AC for ACCESS
- INT for INTERVAL

Any TSO commands can be used in a batch job, using the JCL for executing the TSO monitor in batch; for example:

```
//P390S  JOB 1,P390,MSGCLASS=X
//TSOBAT01 EXEC PGM=IKJEFT01
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUADS  DD   DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD   DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN  DD   *
LD DA('MARTIN.*') AUTHUSER
LU MARTIN
/*
```

Where:

```
LD DA('MARTIN.*') AUTHUSER  -  would list generic profile MARTIN and
                               its access list
LU MARTIN   -  would display the basic RACF data for userid MARTIN
```

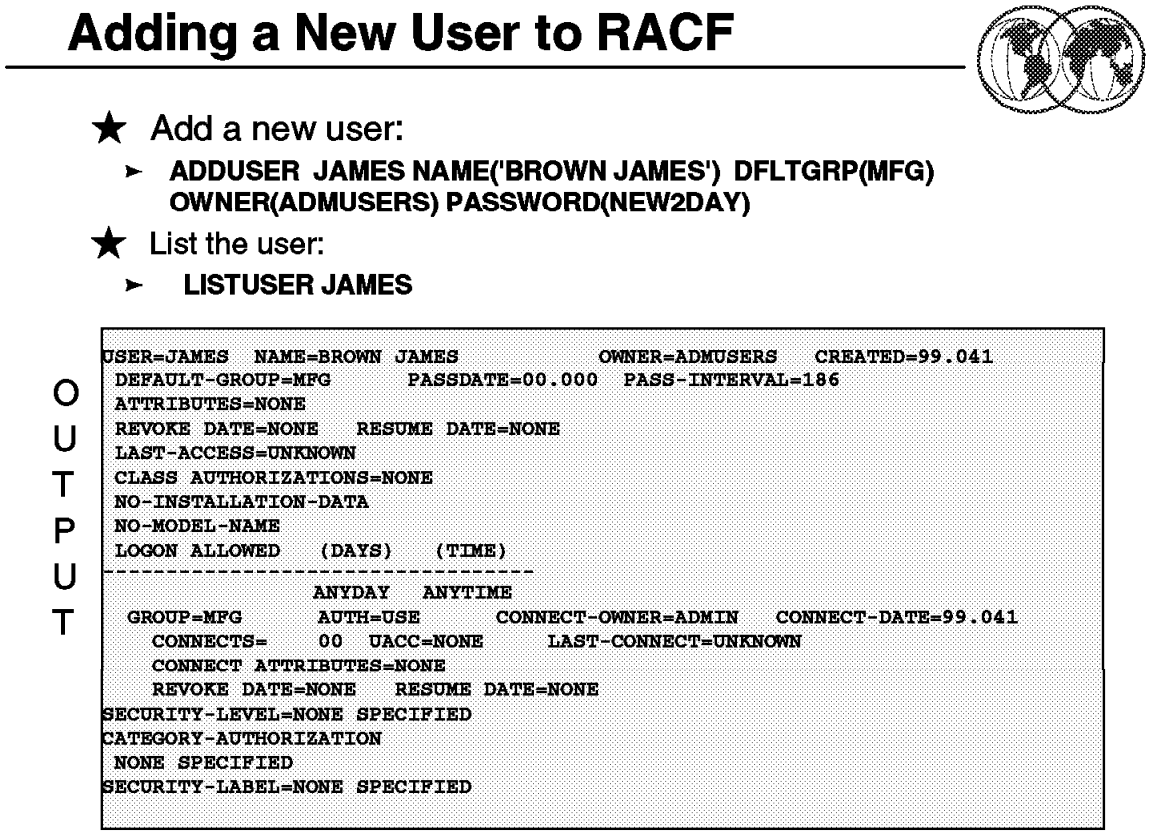


Figure 105. How to add a user

2.8.2 How to add a user

When you define a user's profile (using the ADDUSER command) or change a user's profile (using the ALTUSER command), you can specify the information contained in each field of each segment of the profile.

The command adds a profile for the new user to the RACF database and creates a connect profile that connects the user to whichever default group you specify.

The user profile consists of a RACF segment and, optionally, other segments such as a TSO segment, a DFP segment, or an OMVS segment. You can use this command to define information in any segment of the user's profile.

The visual shows sample output from the following ADDUSER command when the LISTUSER is issued:

```
ADDUSER JAMES NAME('BROWN JAMES') DFLTGRP(MFG)
OWNER(ADMUSERS) PASSWORD(NEW2DAY)
```

This command adds a new user ID JAMES into default group MFG.

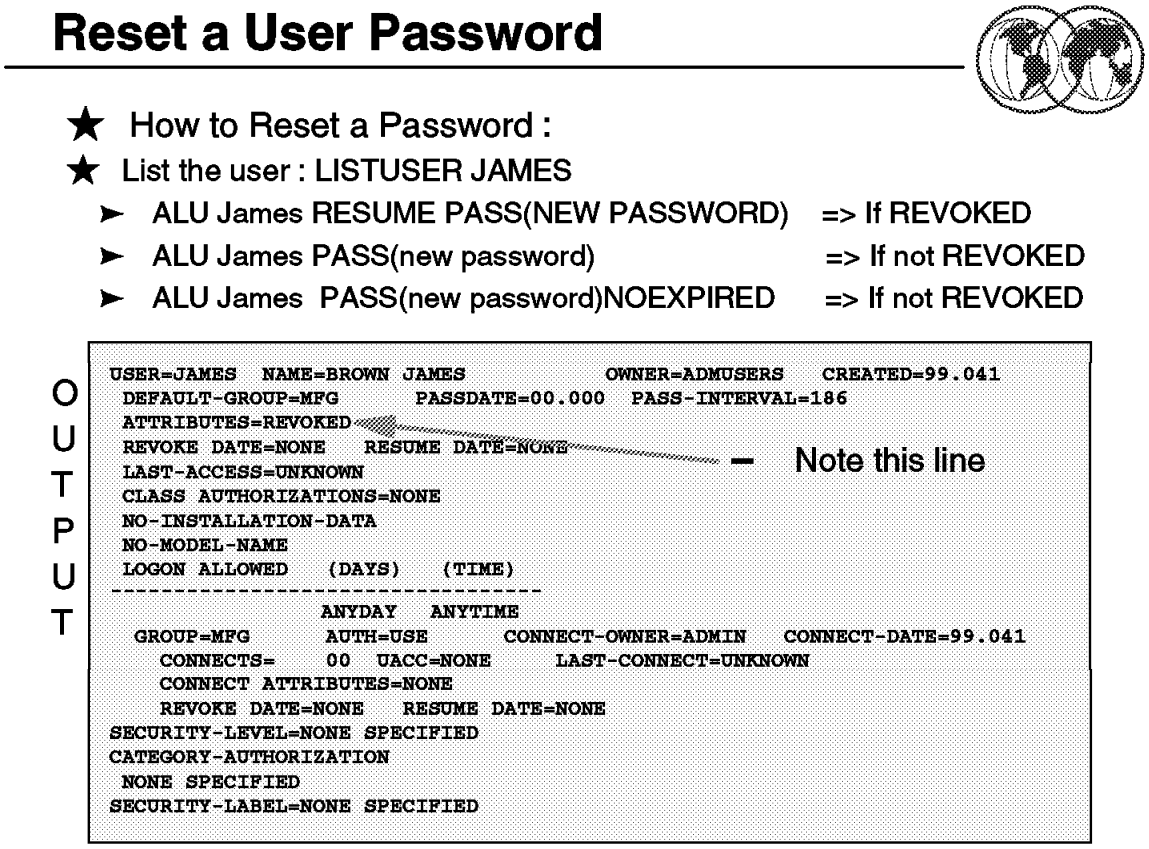


Figure 106. How to reset a password

2.8.3 How to reset a password

A system administrator is often asked to reset a user's password. There are two common reasons for this:

1. The user forgot his password (or made too many errors when attempting change it), or
2. The user ID has been REVOKED for some reason.

You can use the RACF ISPF panels to reset passwords, but it is easier to use direct RACF TSO commands. There are two relevant commands:

PASSWORD When used to reset another user's password, the only option is to set the password equal to the user's default group name. The default group name is often SYS1; if the PASSWORD command is used to reset a user's password, the password will probably be SYS1. This has obvious security consequences.

ALTUSER You can select the password to assign when you use this command. Furthermore, you can determine whether the password you set will be expired or not.

In both cases, the password is automatically marked as *expired*, by default. This means that the user will be forced to select a new password the next time he logs onto the system. With the ALU command, you can also set an unexpired password, one that the user can use until he changes it for some reason.

Before resetting a password, we suggest you always use the LISTUSER command to verify that the user definition exists, and determine if the user is REVOKED. For example, in this case we would probably use this command:

```
ALU martin RESUME PASS(newpwd)      <== if REVOKED
ALU martin PASS(newpwd)             <== if not REVOKED
ALU martin PASS(newpwd) NOEXPIRED   <== if not REVOKED

PASSWORD NOINTERVAL USER(martin)    <== if you want this
```

You would need to tell Martin the new password you assigned. He will need it to log on, but will be forced to change it immediately to a password of his own selection unless you used the NOEXPIRED option. The PASSWORD NOINTERVAL command will prevent this user's password from ever expiring.[1] You need SPECIAL authority to issue these commands.

2.8.3.1 How to reset a password with ISPF panels

You can also use the RACF ISPF panels to change or reset passwords. The end result is the same as if you used the direct commands shown above.

```
                        RACF - CHANGE USER JAMES
COMMAND ===>

ENTER THE DESIRED CHANGES:

   OWNER             ===>          Userid or group name

   USER NAME         ===>

   DEFAULT GROUP     ===>          Group name

   PASSWORD          ===>          User's password OR use default ===>
                     ===>          Re-enter password to verify
        EXPIRED      ===>          Mark new password as expired ?

   PASSWORD INTERVAL ===>          1 - 254 days, NO, or blank

   REVOKE            ===>          YES, mm/dd/yy (date) or blank

   RESUME            ===>          YES, mm/dd/yy (date) or blank
```

Figure 107. RACF Change User menu

The path to the appropriate RACF ISPF panels is:

```
ISPF Primary Option Menu
  RACF     (select RACF from the primary ISPF menu)
    RACF - Services Option Menu
      User Profiles and Your Own Password
        RACF - User Profile Services
          CHANGE (and enter target userid in the USER field)
```

This should produce the panel shown in the previous figure, and you would carry on from this point. Remember that whatever password you assign must be changed by the user when he logs onto the

[1] Using this is very poor security, but may be appropriate in smaller, closed systems.

system the next time. This same panel, and follow-on panels shown after you press ENTER, can be used to change the same elements as the ALTUSER command.

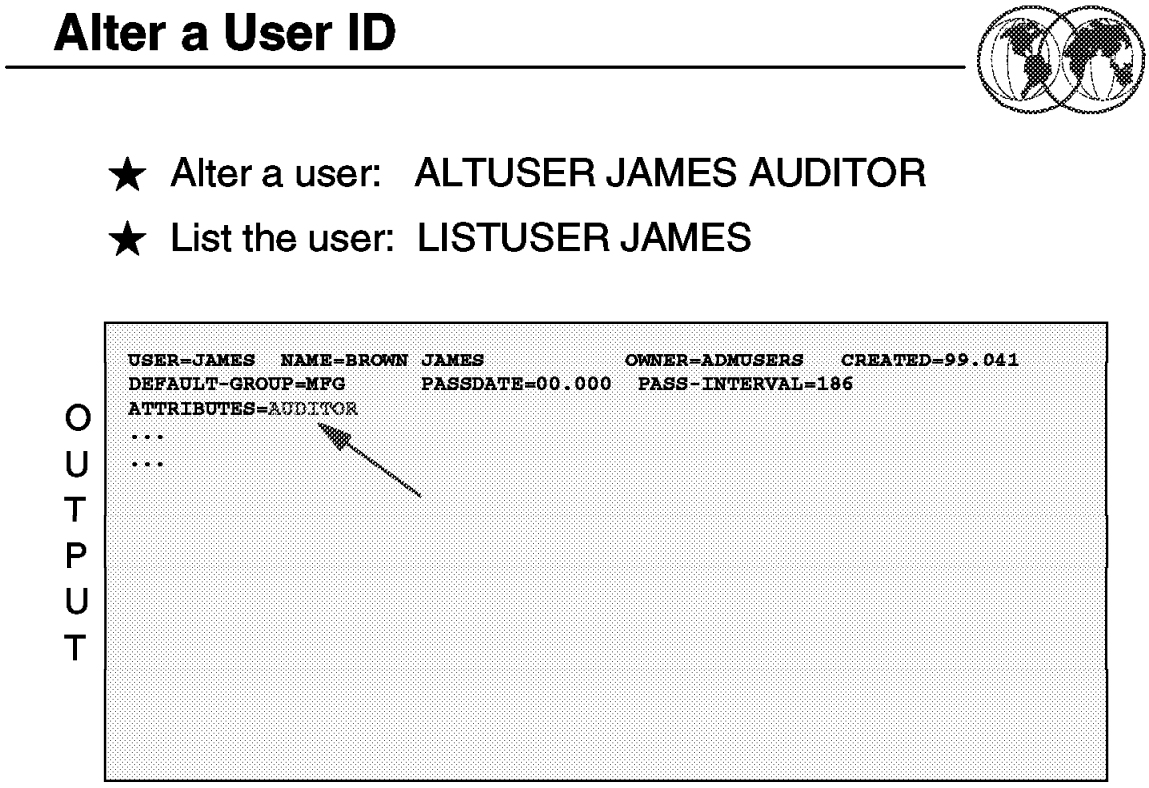


Figure 108. How to alter a user ID segment

2.8.4 How to alter a user ID segment

Use the ALTUSER command to change the information in a user's profile, including the user's system-wide attributes and authorities. The user profile consists of a RACF segment and, optionally, other segments such a TSO segment or a DFP segment. You can use this command to change information in any segment of the user's profile.

When you change a user's level of authority in a group (using the AUTHORITY operand), RACF updates the appropriate group profile. When you change a user's default universal access authority for a group (using the UACC operand), RACF changes the appropriate connect profile. For all other changes, RACF changes the user's profile.

The visual shows sample output from the ALTUSER command:

```
ALTUSER JAMES AUDITOR
```

This command adds the attribute of AUDITOR to the user ID JAMES.

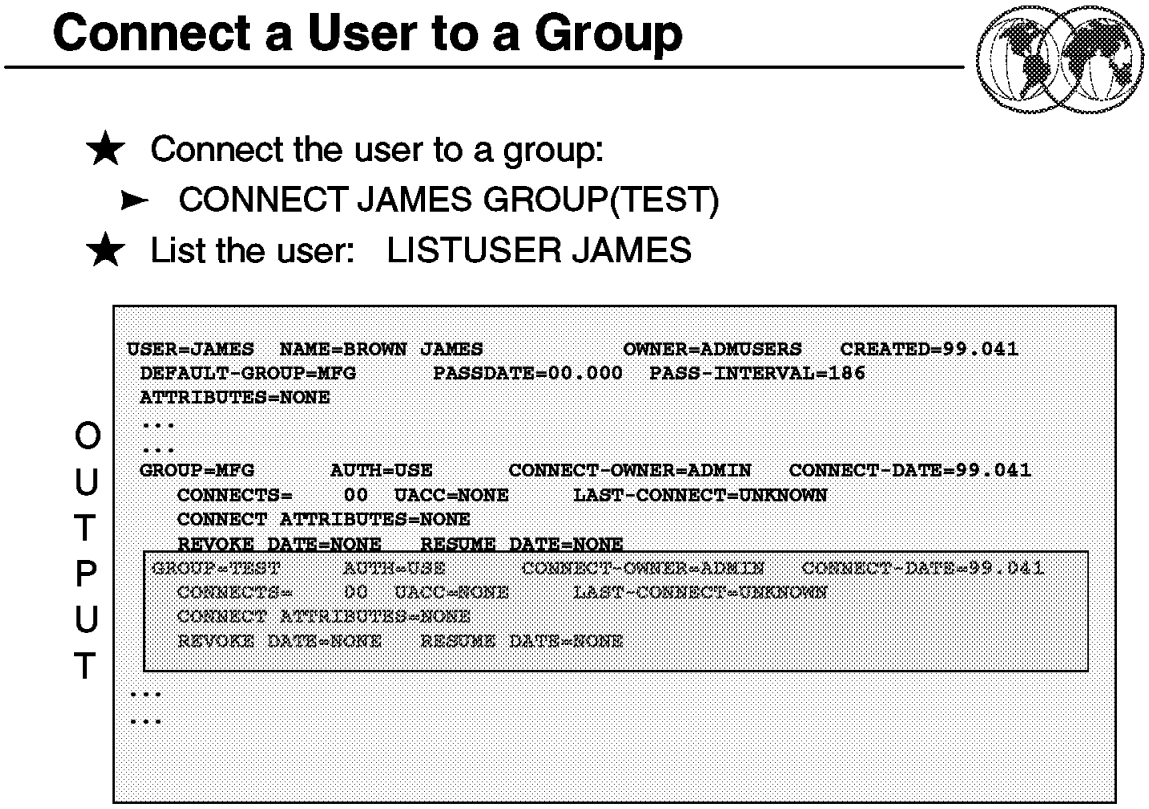


Figure 109. How to connect a user to a group

2.8.5 How to connect a user to a group

Use the CONNECT command to connect a user to a group, modify a user's connection to a group, or assign the group-related user attributes. If you are creating a connection, defaults are available as stated for each operand. If you are modifying an existing connection, no defaults apply.

To use the CONNECT command, you must have at least one of the following:

- The SPECIAL attribute
- The group-SPECIAL attribute in the group
- The ownership of the group
- JOIN or CONNECT authority in the group

The visual shows sample output from the CONNECT command:

CONNECT JAMES GROUP(TEST)

This command connects user JAMES to group TEST.

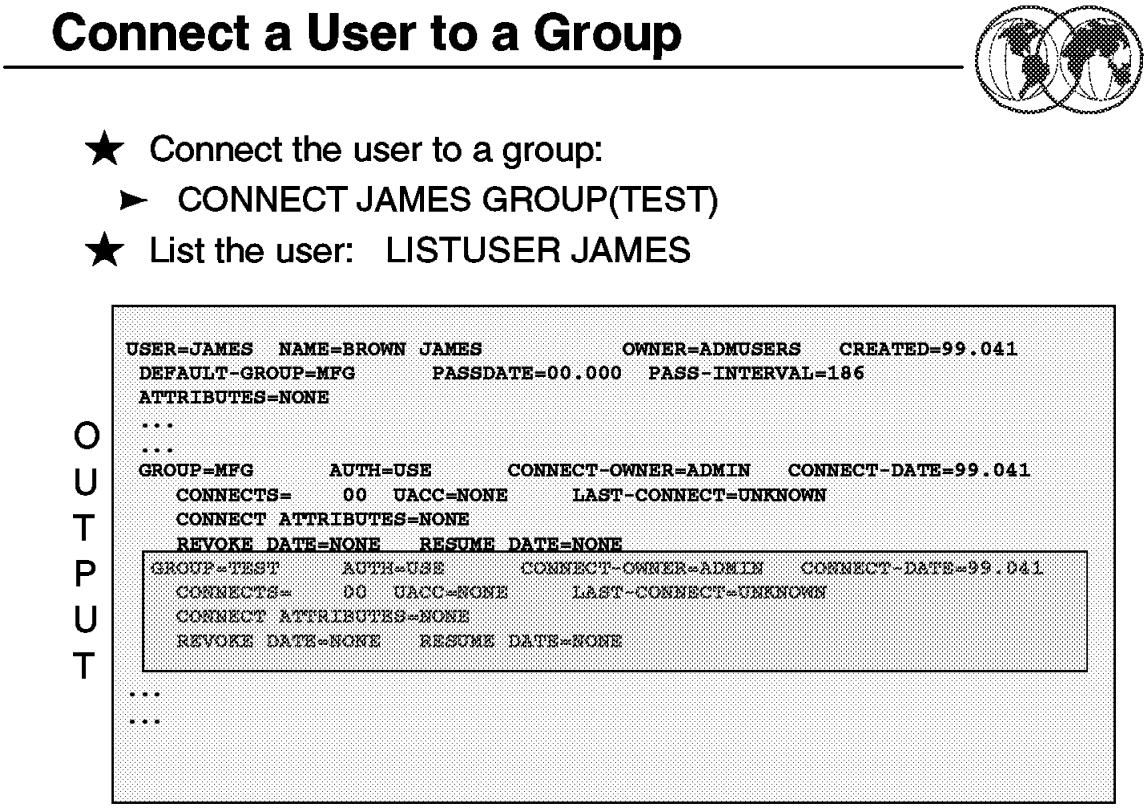


Figure 110. How to remove a user from a group

2.8.6 How to remove a user from a group

You can use the REMOVE command to remove a user from a group, and to assign a new owner to any group data set profiles the user owns on behalf of that group.

To use the REMOVE command, one of the following conditions must be true:

- You have the SPECIAL attribute.
- The group profile is within the scope of a group in which you have the group-SPECIAL attribute.
- You are the owner of the group.
- You have JOIN or CONNECT authority in the group.

The visual shows sample output from the REMOVE command:

```
REMOVE JAMES GROUP(TEST)
```

Change a User's Password Interval

★ Change password interval:
  ► PASSWORD USER(JAMES) INTERVAL(60)
★ List the user: LISTUSER JAMES

OUTPUT

```
USER=JAMES   NAME=BROWN JAMES            OWNER=ADMUSERS    CREATED=99.041
 DEFAULT-GROUP=MFG       PASSDATE=00.000  PASS-INTERVAL=60
 ATTRIBUTES=NONE
 ...
 ...
```

Figure 111. How to a change a user

2.8.7 How to a change a user′s password interval

The *interval* indicates the number of days during which a password remains valid; the range is from 1 through 254 days.

The value you specify here cannot exceed the value, if any, that your installation has specified using the INTERVAL operand on the SETROPTS command. The initial system default after RACF initialization is 30 days.

If you specify INTERVAL on the PASSWORD command without a change-interval value, RACF uses the installation-specified maximum.

The visual shows sample output from the PASSWORD command:

```
PASSWORD USER(JAMES) INTERVAL(60)
```

This command would set user ID James′ password expiry date to 60 days. Overriding any system default password expiring setting is set by the SETROPTS command.

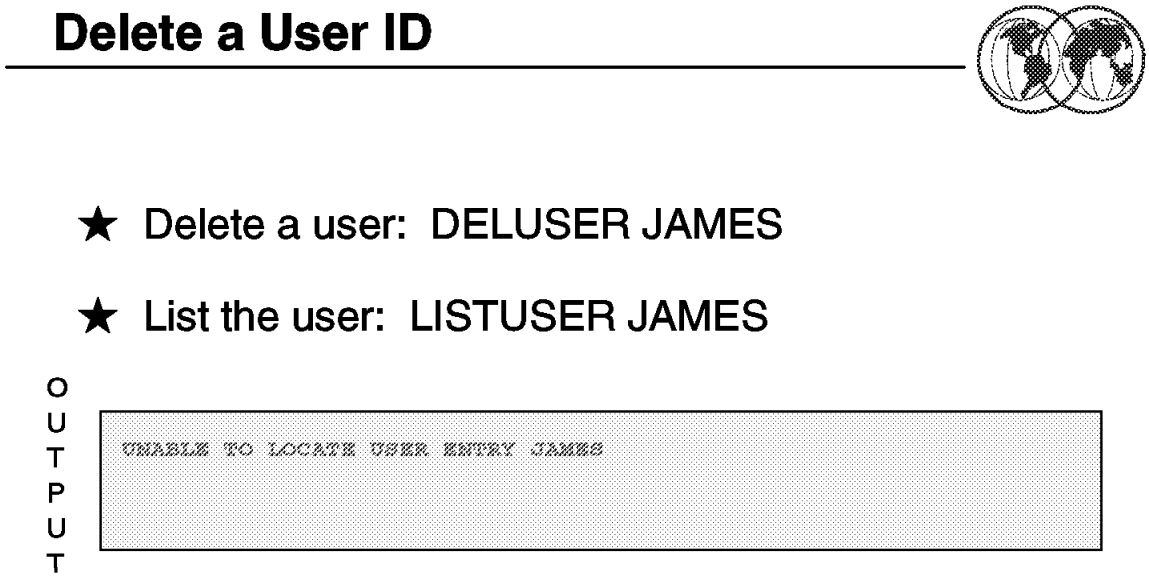


Figure 112. How to a delete a user

2.8.8 How to a delete a user

Use the DELUSER command to delete a user from RACF. This command removes the user′s profile and all user-to-group connections for the user. (The connect profiles define the user′s connections to various RACF groups.)

There are, however, other places in the RACF database where the user′s user ID might appear, and the DELUSER command does not delete the user ID from all these places. Specifically, the user could be the owner of a group, the owner of a user′s profile, the owner of a group data set, or in an access list for any resource. Before issuing DELUSER, you must first issue the REMOVE command to assign new owners for any group data sets the user owns in groups other than his default group. You can use the RACF Remove ID utility (IRRRID00) to remove all of the occurrences of a user ID. For information on using the RACF Remove ID utility, see *OS/390 Security Server (RACF) Security Administrator′s Guide*, SC28-1915.

To use the DELUSER command, at least one of the following must be true:

- You must have the SPECIAL attribute.
- The user profile to be deleted must be within the scope of a group in which you have the group-SPECIAL attribute.
- You must be the owner of the user′s profile.

The visual shows sample output from the DELUSER command:

```
DELUSER JAMES
```

This command deleted user ID James.

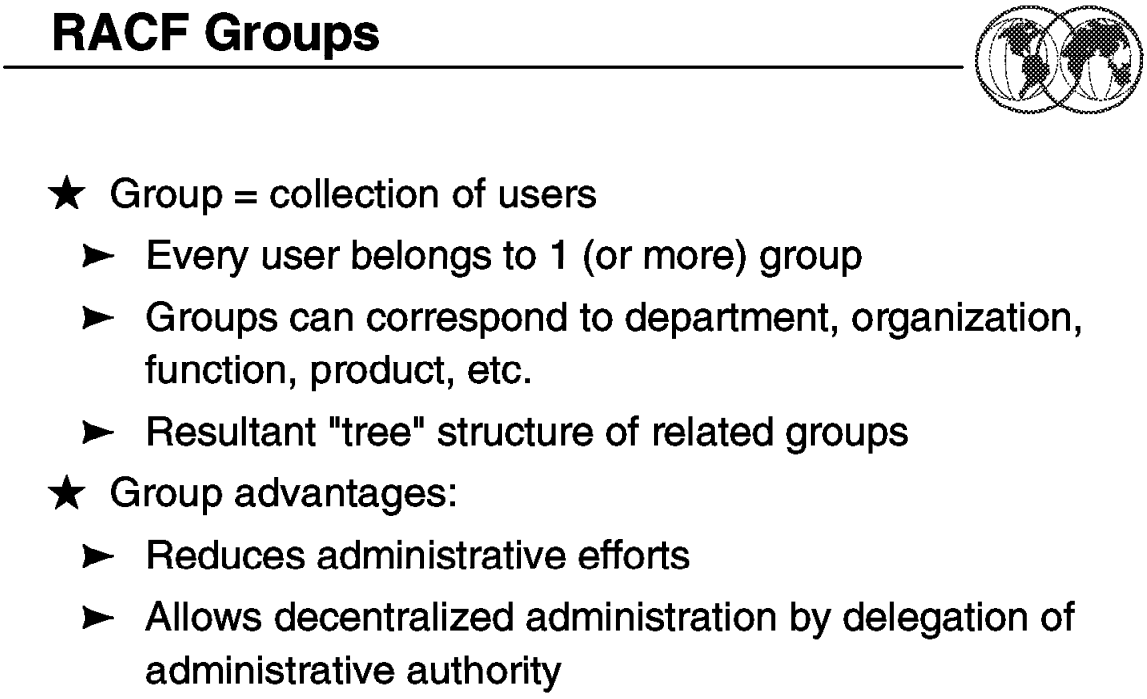


Figure 113. RACF groups

2.9 RACF groups

With RACF, all defined users belong to at least one group. You can think of the groups forming a hierarchical or “tree” structure, where each group is owned by a superior group. Groups can also own resources as well as users in another group.

RACF has the following types of groups:

Administrative You can create a group simply as an administrative convenience. For example, you might create a group to represent an organizational entity, such as a region or a division.

With RACF delegation, you can create this kind of group for each group administrator. Operating from such groups, the group administrators can then define other groups needed by their local users.

Holding This is a technique that retains user definition centrally, yet allows the effective use of group administrators to establish a holding group. You define all users centrally and initially connect them to a group named HOLD with the minimum of authorities. HOLD does not appear in any access lists, and therefore has no real significance to the user.

Group administrators, to whom you give CONNECT (but not JOIN) authority, can connect the appropriate users to the groups under their control and change the users′ default group name as appropriate. This technique allows the installation to assign correct account numbers and control other installation considerations while allowing flexibility in the grouping of the user population.

Data Control You can create a group to act as a control point for the protection of data. For example, by using the group SYS1, you can determine which users are permitted to protect the SYS1 data sets. Only users with CREATE authority or higher in this group can protect system data sets. At your location, you might consider defining one such group for every high level qualifier representing data that is to be protected.

Functional A group can represent a functional area of the installation for the purpose of data sharing. For example, a financial analyst might need to access a variety of resources across many groups, such as accounting, payroll, marketing, and others. Of course, the owners of each resource could permit the financial analyst to access their resources by placing the analyst′s user ID on an access list. But if a new financial analyst takes over the job, it is then necessary to add the new user ID to each RACF profile. Likewise, the RACF profiles must be updated when the analyst no longer has a need to access the data. This arrangement involves a great deal of unnecessary activity by the resource owners.

Instead, you can create a group that represents the financial analyst function and permits access to the data defined to the group. Access to the entire range of data can then be managed by controlling the user population in the defined group. For cases involving one-time access, owners of the needed data would simply PERMIT access by the defined group. Where appropriate, the group name could be included in profile access lists to ensure automatic availability of needed data to the financial analyst group. New financial analysts could be connected to the group, as required, to gain access to the entire range of data. Likewise, analysts could be removed from the group whenever necessary. By controlling the user population of such a functional group, resource profile changes on a day-to-day basis becomes unnecessary.

User You can define a group to serve as an anchor point for users who otherwise have no common access requirements. For example, engineers and scientists, as well as other problem-solving users, might have no need to access application-related data in the system. Their only interest may be in their own personal data. You can place this set of users in a single group that has no access to other data.

You can also define groups based on access level. For example, if PAY.DATA is a RACF-defined data set, two groups could be defined, PAYREAD and PAYUPDTE, both of which would appear in the PAY.DATA access list, but with READ and UPDATE access, respectively. Any users requiring access would be connected as appropriate, by the group administrator.

RACF Group Structure Example

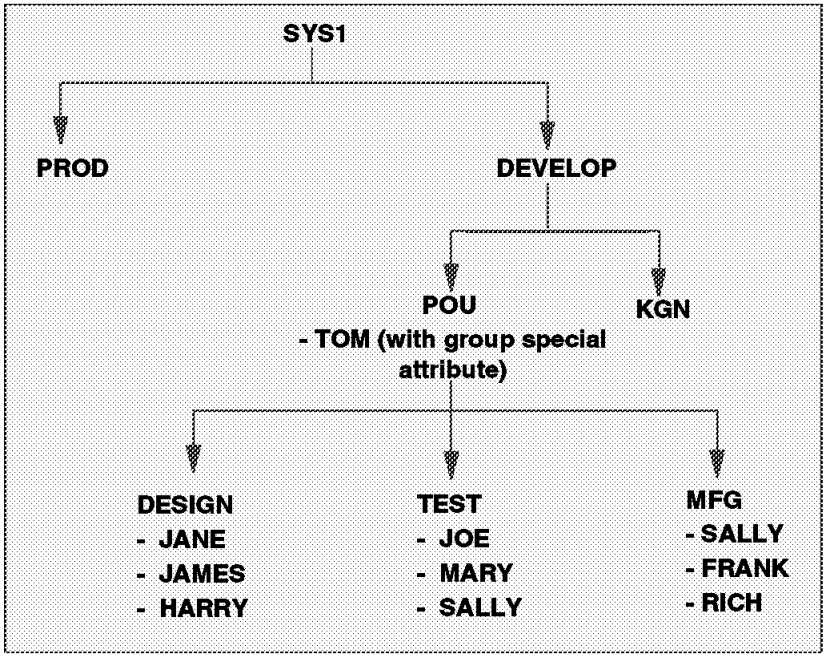


Figure 114. RACF group structure

2.9.1 RACF group structure

The group structure of RACF can be mapped to the organizational structure that exists at your installation. That is, RACF conforms naturally to a tree structure of groups, where each group (except SYS1, which is predefined as the highest group) has a superior, or owning, group. Groups can correspond directly to business entities such as divisions, departments, and projects. Users can be connected to one or more groups.

When you define a group, you should consider the basic purpose of the group. Is it an administrative group, a holding group, a data control group, a functional group, or a user group? When setting up RACF groups, keep in mind that the maximum number of users that you can connect to any one group is approximately 5900.

You should map your groups to your organization′s structure and arrange them hierarchically, with the IBM-supplied SYS1 group as the highest group, so that each group is a subgroup of another group.

A user may be connected in more than one group (in the example, SALLY is connected to MFG and TEST groups).

In this visual: GROUP, DESIGN, TEST, and MFG are all owned by group POU. Tom is connected to group POU as special. This gives Tom (who is the RACF administrator) control over all POU resources DESIGN, TEST, and MFG.

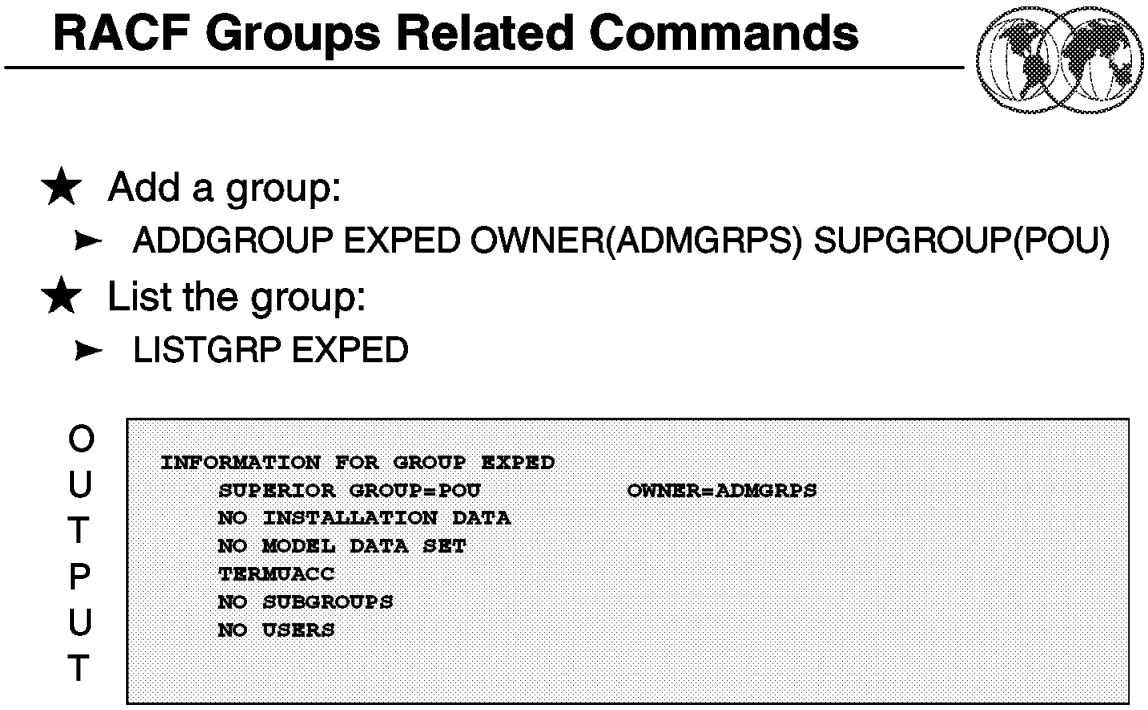


Figure 115. How to add a group

2.9.2 How to add a group

Use the ADDGROUP command to define a new group to RACF. The command adds a profile for the new group to the RACF database. It also establishes the relationship of the new group to the superior group you specify.

Group profiles consist of a RACF segment and, optionally, other segments such as DFP and OMVS. You can use this command to specify information in any segment of the profile.

To use the ADDGROUP command, you must meet at least one of the following conditions:

- Have the SPECIAL attribute
- Have the group-SPECIAL attribute and the superior group is within your group-SPECIAL scope
- Be the owner of the superior group
- Have JOIN authority in the superior group

The visual shows sample output from the ADDGROUP command:

```
ADDGROUP EXPED OWNER(ADMGRPS) SUPGROUP(POU)
```

This command added a new group named EXPED and it is a subgroup to group POU.

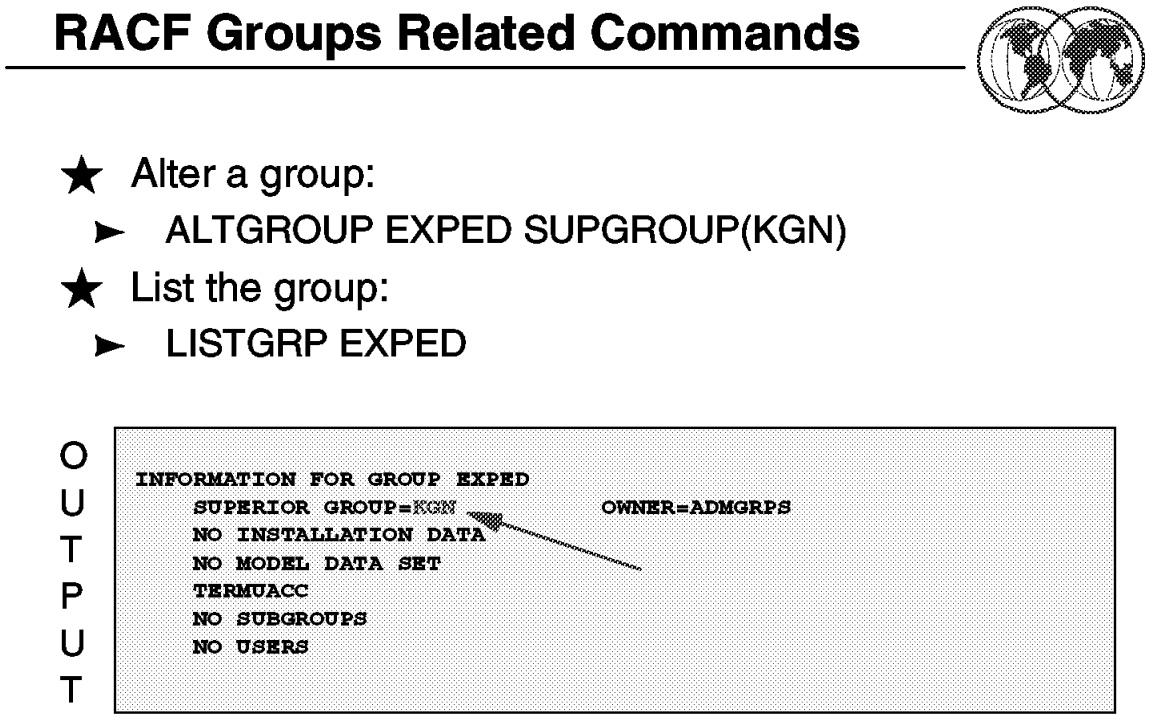


Figure 116. How to alter a group

2.9.3 How to alter a group

Use the ALTGROUP command to change:

- The superior group of a group
- The owner of a group
- The terminal indicator for a group
- A model profile name for a group
- The installation-defined data associated with a group
- The default segment information for a group (for example, DFP or OMVS).

To change the superior group of a group, you must meet at least one of the following conditions:

- You must have the SPECIAL attribute
- All the following group profiles must be within the scope of a group in which you have the group-SPECIAL attribute:
  - The group whose superior group you are changing
  - The current superior group
  - The new superior group
- You must be the owner of, or have JOIN authority in, both the current and the new superior groups.

Note: You can have JOIN authority in one group and be the owner of, or have the group-SPECIAL attribute in, the other group.

The visual shows sample output from the ALTGROUP command:

```
ALDGROUP EXPED SUPGROUP(KGN)
```

This command moved the group named EXPED from being a subgroup of group PGN to a subgroup to group KGN.

RACF Groups Related Commands

★ Connect a user to a group:

➤ CONNECT JAMES GROUP(EXPED)

★ List the group:

➤ LISTGRP EXPED

OUTPUT

```
INFORMATION FOR GROUP EXPED
  SUPERIOR GROUP=KGN       OWNER=ADMGRPS
  NO INSTALLATION DATA
  NO MODEL DATA SET
  TERMUACC
  NO SUBGROUPS

  USER(S)=   ACCESS=   ACCESS COUNT=   UNIVERSAL ACCESS=
   JAMES       USE          000000            NONE
  CONNECT ATTRIBUTES=NONE
  REVOKE DATE=NONE          RESUME DATE=NONE
```

Figure 117. How to connect a user to a group

2.9.4 How to connect a user to a group

Use the CONNECT command to connect a user to a group, modify a user's connection to a group, or assign the group-related user attributes. If you are creating a connection, defaults are available as stated for each operand. If you are modifying an existing connection, no defaults apply.

To use the CONNECT command, you must have at least one of the following:

- The SPECIAL attribute
- The group-SPECIAL attribute in the group
- The ownership of the group
- JOIN or CONNECT authority in the group

You cannot give a user a higher level of authority in the group than you have.

The visual shows sample output from the CONNECT command:

```
CONNECT JAMES GROUP(EXPED)
```

This command connects user James to the EXPED group.

RACF Groups Related Commands

★ Remove a user from a group:

► REMOVE JAMES GROUP(EXPED)

★ List the group:

► LISTGRP EXPED

OUTPUT

```
INFORMATION FOR GROUP EXPED
    SUPERIOR GROUP=KGN          OWNER=ADMGRPS
    NO INSTALLATION DATA
    NO MODEL DATA SET
    TERMUACC
    NO SUBGROUPS
    USER(S)      ACCESS      ACCESS COUNT      UNIVERSAL
ACCESS
      JAMES        USE           000000            NONE
      CONNECT ATTRIBUTES=NONE
      REVOKE DATE=NONE                  RESUME DATE=NONE
```

Figure 118. How to remove a user from a group

2.9.5 How to remove a user from a group

You can use the REMOVE command to remove a user from a group, and to assign a new owner to any group data set profiles the user owns on behalf that group.

The visual shows sample output from the REMOVE command:

```
REMOVE JAMES GROUP(EXPED)
```

This command removes user James from the EXPED group.

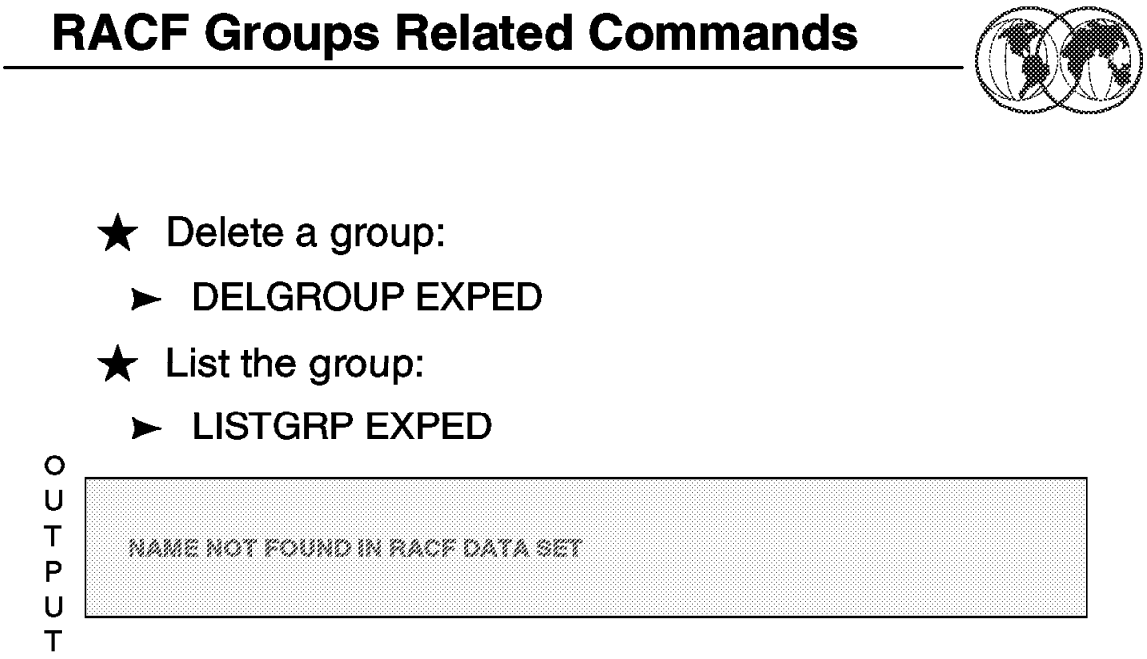


Figure 119. How to delete a group

2.9.6 How to delete a group

Use the DELGROUP command to delete a group and its relationship to its superior group from RACF.

There are, however, other places in the RACF database where the group name might appear, and DELGROUP processing does not delete these other occurrences of the group name. For example, the group name could be in the access list for any resource. You can use the RACF Remove ID utility (IRRRID00) to remove all occurrences of a group name. For information on using the RACF Remove ID utility, see *OS/390 Security Server (RACF) Security Administrator's Guide*, SC28-1915.

The visual shows sample output from the DELGROUP command:

```
DELGROUP EXPED
```

This command deletes the EXPED group.

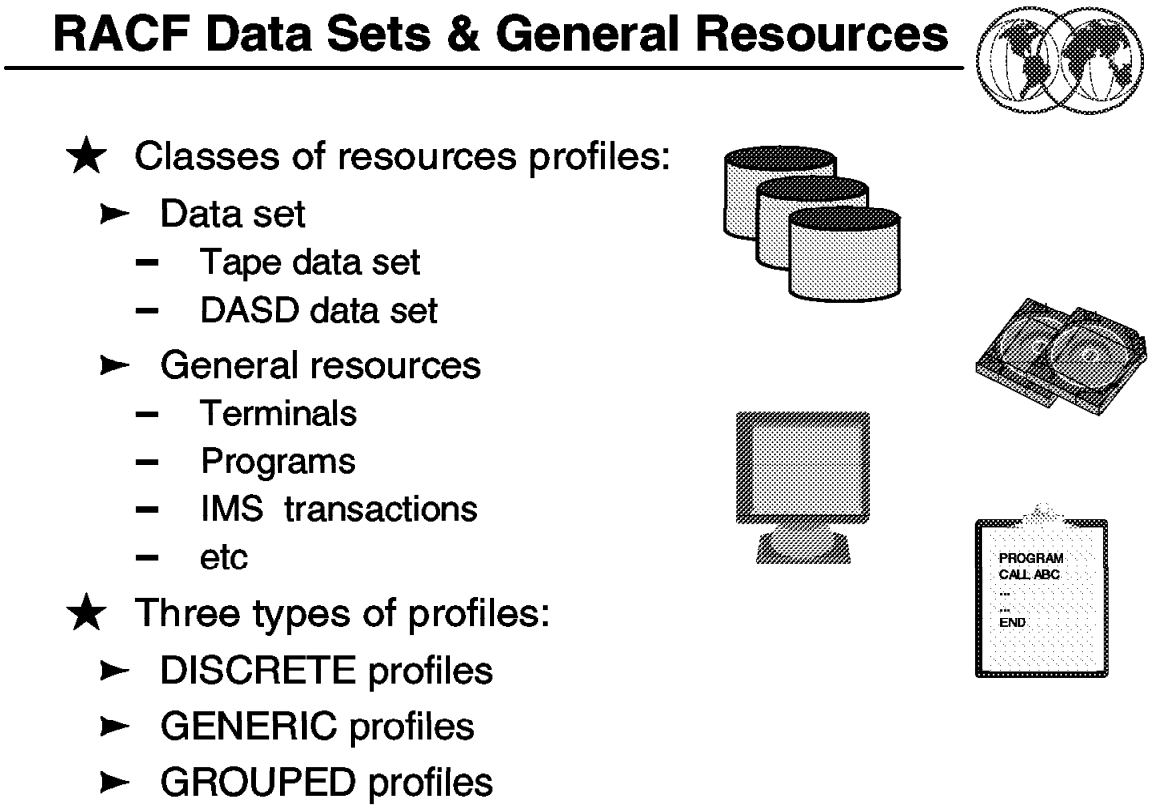


Figure 120. Controlling access to resources

2.9.7 Controlling access to resources

To protect a general resource, create a general resource profile using the RDEFINE command. When you create a general resource profile, you must specify a general resource class for the profile. IBM supplies a list of the general resource classes in the class descriptor table (CDT). The classes for OS/390 systems are relevant to the system on which you are running the OS/390 Security Server (RACF).

RACF-protected resources can be divided into two categories: data sets and general resources. General resources are all of the resources that are defined in the class descriptor table. For example, general resources include DASD and tape volumes, load modules (programs), terminals, and others.

RACF allows the installation to set its own rules for controlling the access to its resources by defining what is controlled at what level. The installation can tailor RACF to interact with its present operating environment and assign security responsibilities either on a system-wide or a group-wide basis.

The three types of profiles are described in 2.7.1, “RACF resource profiles” on page 196. RACF has its own algorithm to match resources to profiles. The basic steps are:

The installation establishes the controls; RACF enforces them.

Resource profiles contain:

- The owner of the profile
- The auditing parameters

- The Universal Access authority
- An access list with users and groups
- A “Warning” indicator
- A security classification
- A real-time notification information
- An erase-on-scratch indication for data sets
- A volume and a unit (if data set)
- A security retention period (if tape data set)
- Access statistics

Your installation can add new class descriptor table (CDT) entries or modify or delete existing entries that you have added in the installation-defined class descriptor table (ICHRRCDE). When you define a new resource class, you can optionally designate that class as either a resource group class or a resource member class. For a resource group class, each user or group of users that is permitted access to that resource group is permitted access to all members of the resource group. Note that for each resource group class you create, you must also create a second class that represents the members of the group.

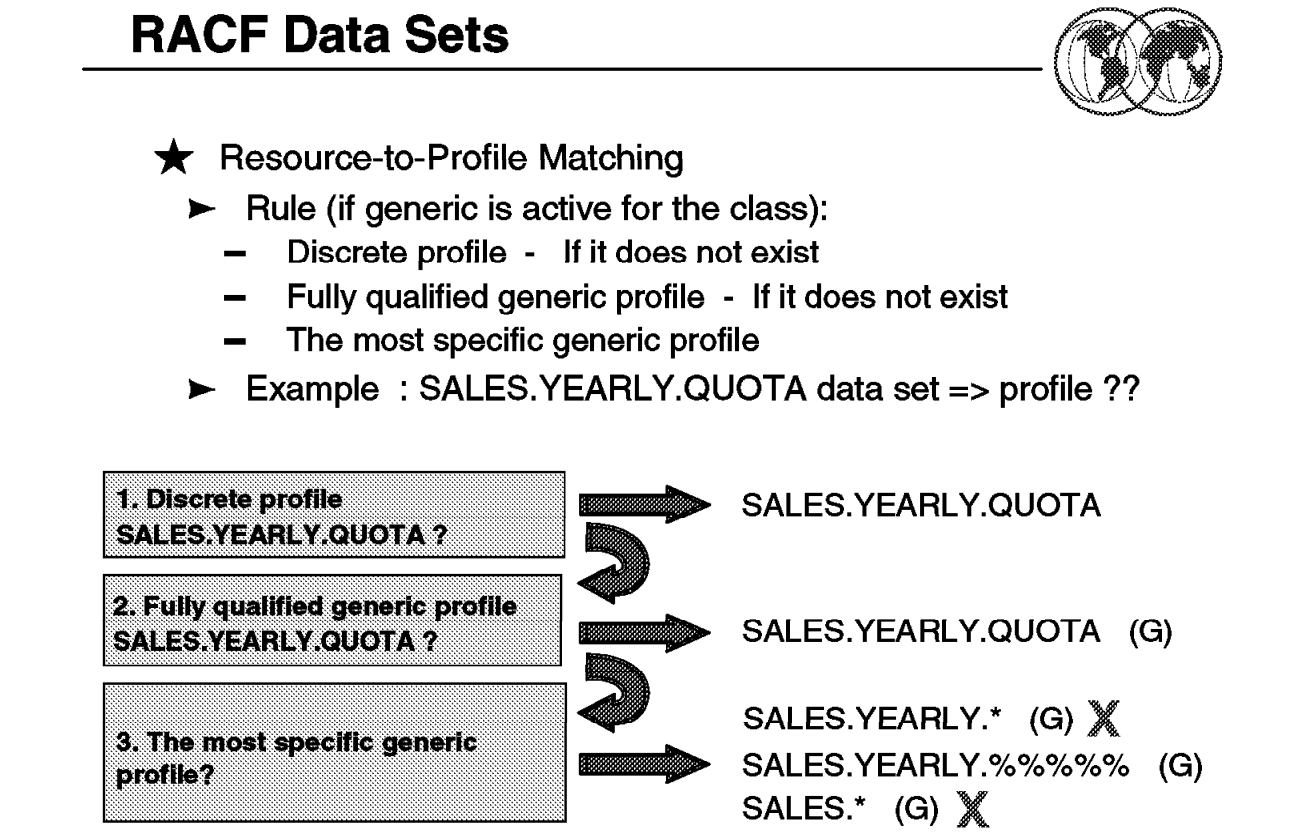


Figure 121. RACF data sets and general resources

2.9.8 RACF data sets and general resources

RACF does the following to locate a resource profile:

- RACF will look for a discrete profile, if no discrete profile is found
- RACF will look for a generic profile and will then use the most qualified generic profile available.

See *OS/390 Security Server (RACF) Security Administrator's Guide*, SC28-1915, for more detail on how this works.

Some of the generic profile naming for general resources has been enhanced with some of the same concepts as generics for data set profiles as valid generic characters as follows:

***** You may have an asterisk (*) within a profile name, representing one qualifier of a resource name, or specify * in the profile name to match more than one character in the same position of the resource name.

****** You may also use a double asterisk (**) to represent zero or more qualifiers within a general resource generic profile or at the end of such a profile, or specify ** in the profile name to match more than one character in the same position of the resource name. Use of the double asterisk (**) in general resource generic profiles is not controlled by the SETROPTS EGN option, which applies only to the data set profiles.

% Specify % any single non-blank character (except a period) in the same position of the resource name

For example, the following profile names all match in the first three character positions (A.B), and are shown in the order RACF examines them:

```
A.B
A.B.B
A.BA
A.BZ
A.B0
A.B9
A.B&X
A.B%
A.B*
```

In the visual example, SALES.YEARLY.QUOTA

2.9.8.1 Choosing between discrete and generic data set profiles

Decide which type of profile to create as follows:

Generic Choose a generic profile for the following reasons:

- If you want to protect more than one data set with the same security requirements.
- If you have a single data set that might be deleted, then re-created, and you want the protection to remain the same, you can create a fully qualified generic profile. The name of a fully qualified generic profile matches the name of the data set it protects. Unlike a discrete profile, a fully qualified generic profile is not deleted when the data set itself is deleted.

Discrete Choose a discrete profile for the following reasons:

- To protect one data set that has unique security requirements. The name of a discrete profile matches the name of the data set it protects.
- To allow changes to a data set profile to take effect immediately, without needing to refresh in-storage copies of the profile.

In the example on the visual, a resource manager issues a security check for the data set SALES.YEARLY.QUOTA. The visual shows the three different types of profiles that could be defined in the RACF database:

1. A discrete profile
2. A fully qualified generic profile
3. Find the most specific generic profile

The example shows that RACF looks for a profile in the order shown. If no discrete profile is found, check for a fully qualified profile. If not found, then find the most specific generic profile, which is the second one in the example, SALES.YEARLY.%%%%%%.

Note: By using generic profiles, your installation can reduce both the number of profiles required to protect data sets and the size of the RACF database, thus making RACF protection easier to administer. In addition, generic profiles are loaded into storage when first needed, are not deleted when the data set they protect is deleted, and are not volume-specific (that is, data sets protected by a generic profile can reside on any volume).

You can create a profile with a generic name when the following is true for the class of the profile:

`SETROPTS GENERIC(DATASET) option is in effect.`

Not only does this option allow the creation of generic profiles, it also causes RACF to use generic profiles during authorization checking.

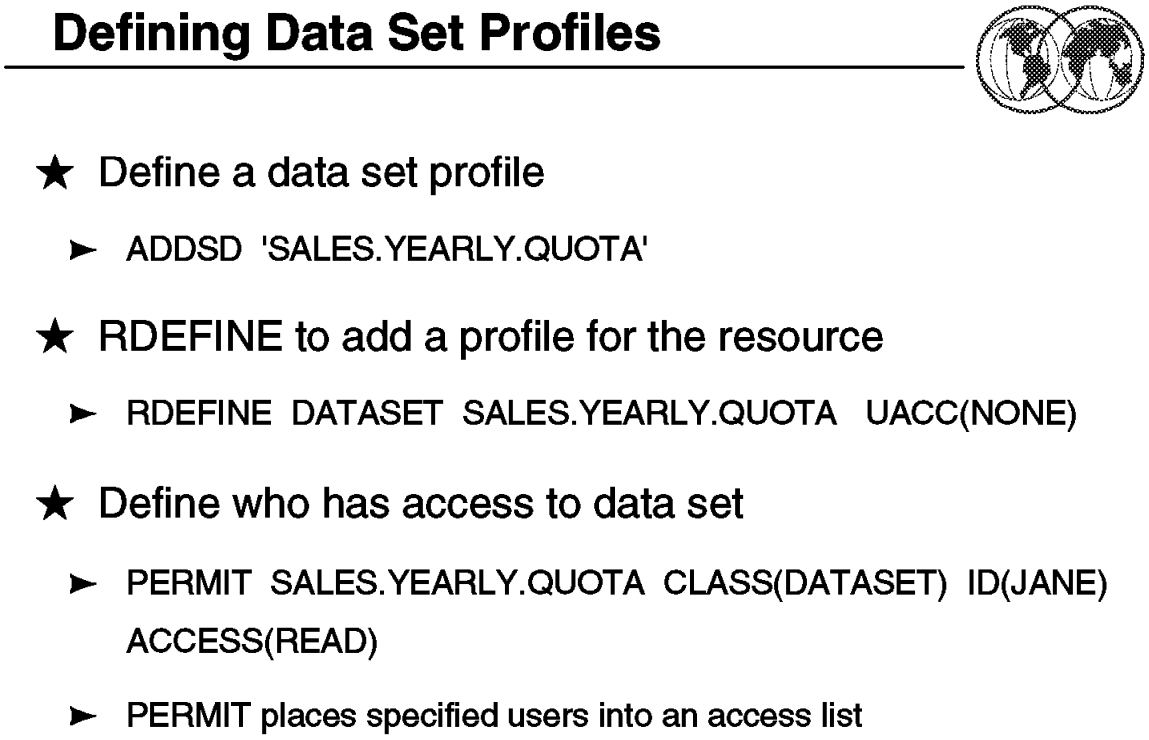


Figure 122. Defining data set profiles

2.9.9 Defining data set profiles

Use the ADDSD command to add RACF protection to data sets with either discrete or generic profiles.

Use the RDEFINE command to define to RACF all resources belonging to classes specified in the class descriptor table. You can also use the RDEFINE command to create entries in the global access checking table and in the lists of security categories and security levels, and to define classes (as profiles in the RACGLIST class) for which RACF saves RACLISTed results on the RACF database.

The RDEFINE command adds a profile for the resource to the RACF database in order to control access to the resource. It also places your user ID on the access list and gives you ALTER authority to the resource unless SETROPTS NOADDCREATOR is in effect.

Note: You cannot use the RDEFINE command to define users, groups, or data sets.

2.9.9.1 Data set profiles

By default, RACF expects a data set name (and the data set profile name) to consist of at least two qualifiers. RACF also expects the high-level qualifier of the data set profile name to be either a RACF-defined user or a RACF-defined group name.

Each data set profile defined to RACF requires a RACF-defined user or group as the owner of the profile. The owner (if a user) has full control over the profile, including the access list.

If the owner of the data set profile is a group, users with group-SPECIAL in that group have full control over the profile.

Ownership of data set profiles is assigned when the profiles are defined to RACF. Note that ownership of a data set profile does not mean that the owner can automatically access that data set. To access a data set, the owner must still be authorized in the profile's access list, unless the high-level qualifier of the profile name is the owner's user ID.

2.9.9.2 Data set profile examples

The RDEFINE in the data set class, DATASET, specifies that no users have access to the data set except the creator of the profile since the universal access, UACC, is none.

To allow users to have access to the data set, the PERMIT command shown specifies that user ID JANE has only READ access to the data set, ACC(READ). User ID JANE exists in the access list for the data set profile via the PERMIT command.

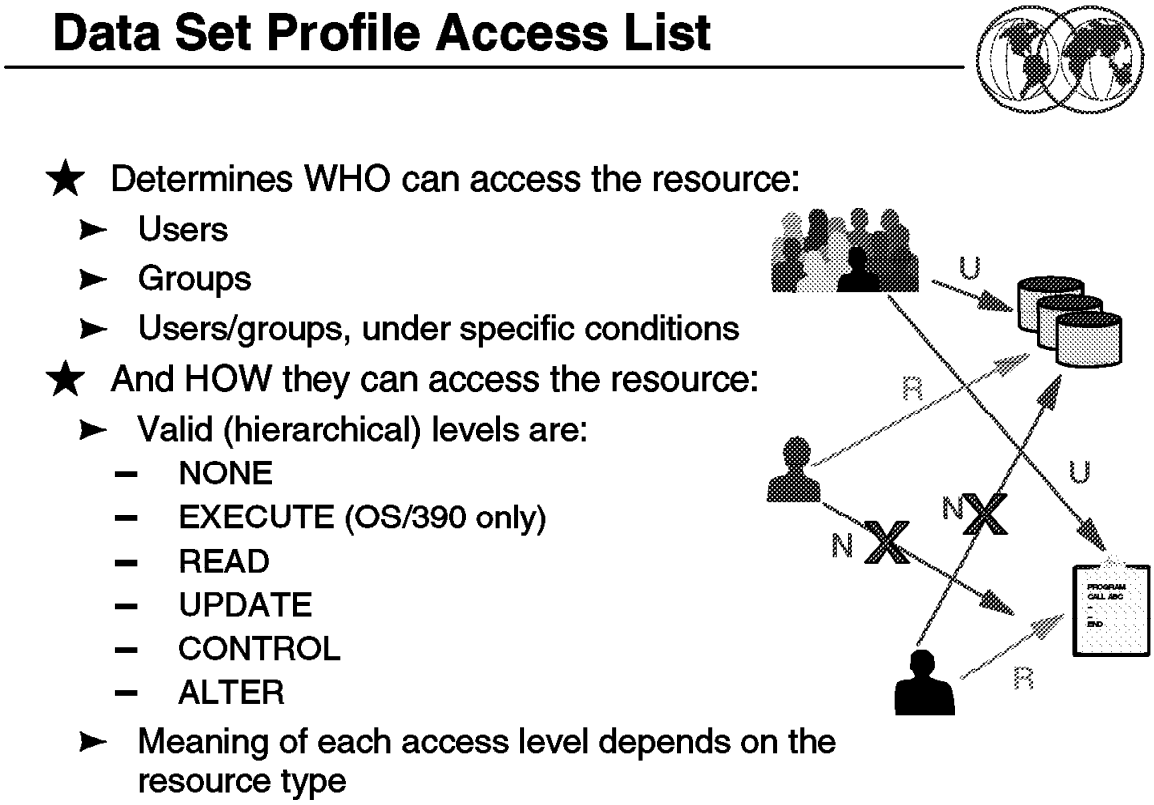


Figure 123. Data set profile access list

2.9.10 Data set profile access list

When a user requests access to a RACF-protected resource (such as a data set), the resource manager issues a RACF authorization request. RACF then performs two checks.

Using the PERMIT command maintains a list of users and groups authorized to access a particular resource. RACF provides two types of access lists: standard and conditional.

Standard The standard access list includes the user IDs and group names authorized to access the resource and the level of access granted to each.

Conditional The conditional access list includes the user and group names authorized to access the resource and the level of access granted to each when a certain condition is met.

2.9.10.1 Types of access levels

ALTER ALTER allows users to read, update, delete, rename, move, or scratch the data set.

When specified in a discrete profile, ALTER allows users to read, alter, and delete the profile itself including the access list.

ALTER does not allow users to change the owner of the profile using the ALTDSD command. However, if a user with ALTER access authority to a discrete data set profile renames the data set, changing the high-level qualifier to his or her own user ID, both the data set and the profile are renamed, and the OWNER of the profile is changed to the new user ID.

When specified in a generic profile, ALTER gives users no authority over the profile itself.

NONE The specified user or group is not permitted to access the resource or list the profile.

EXECUTE For a private load library, EXECUTE allows users to load and execute, but not to read or copy programs (load modules) in the library.

READ Allows users to access the data set for reading only. (Note that users who can read the data set can copy or print it.)

UPDATE Allows users to read from, copy from, or write to the data set. UPDATE does not, however, authorize a user to delete, rename, move, or scratch the data set.

CONTROL For VSAM data sets, CONTROL is equivalent to the VSAM CONTROL password; that is, it allows users to perform improved control interval processing. This is control-interval access (access to individual VSAM data blocks), and the ability to retrieve, update, insert, or delete records in the specified data set.

For non-VSAM data sets, CONTROL is equivalent to UPDATE.

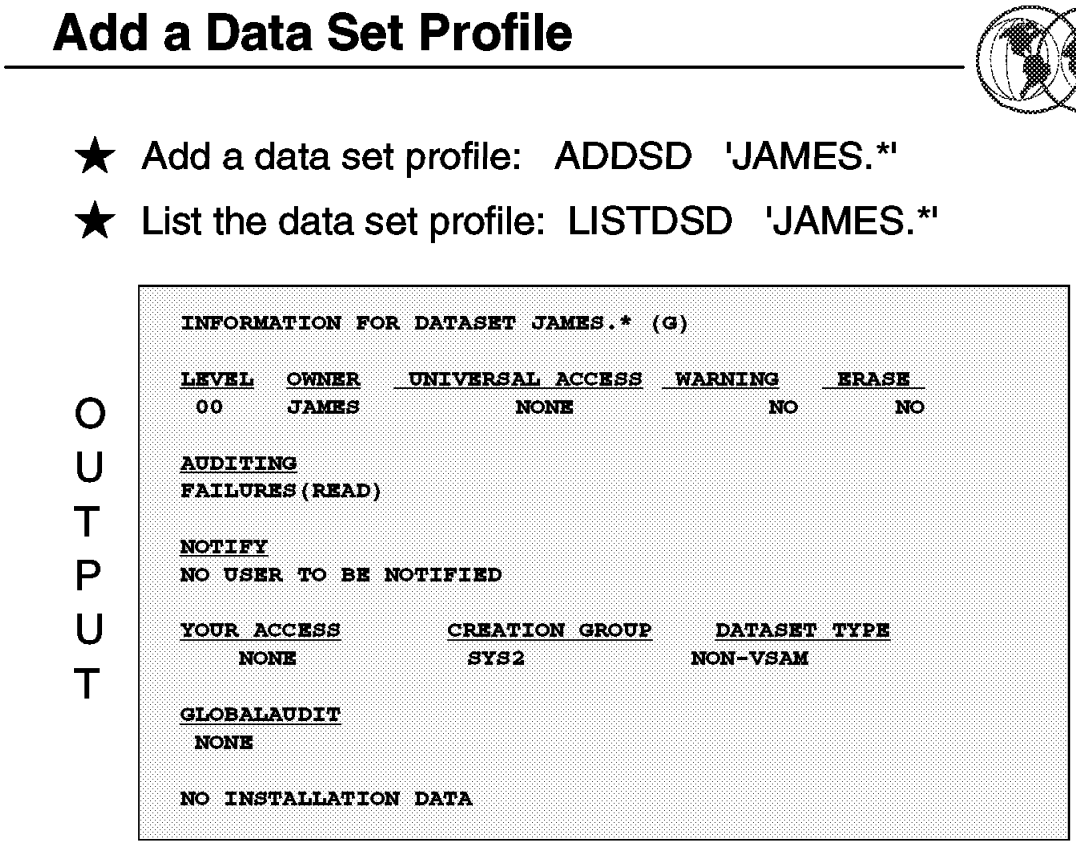


Figure 124. How to add a data set profile

2.9.11 How to add a data set profile

When you define data set profiles to RACF, you can use either standard or nonstandard naming conventions. If you use nonstandard naming conventions, the data set naming convention table and the single-level data set names option are ways to help “fit” RACF standard naming conventions.

The descriptions of naming conventions are followed by rules for protecting and allocating user and group data sets.

By default, RACF expects a data set name (and the data set profile name) to consist of at least two qualifiers. RACF also expects the high-level qualifier of the data set profile name to be either a RACF-defined user or a RACF-defined group name.

This command added a generic profile for data sets with a high level qualifer of james.* (the * character is a valid generic character for more than one character in this position).

```
ADDSD 'JAMES.*'
```

The visual shows sample output from the LISTDSD command:

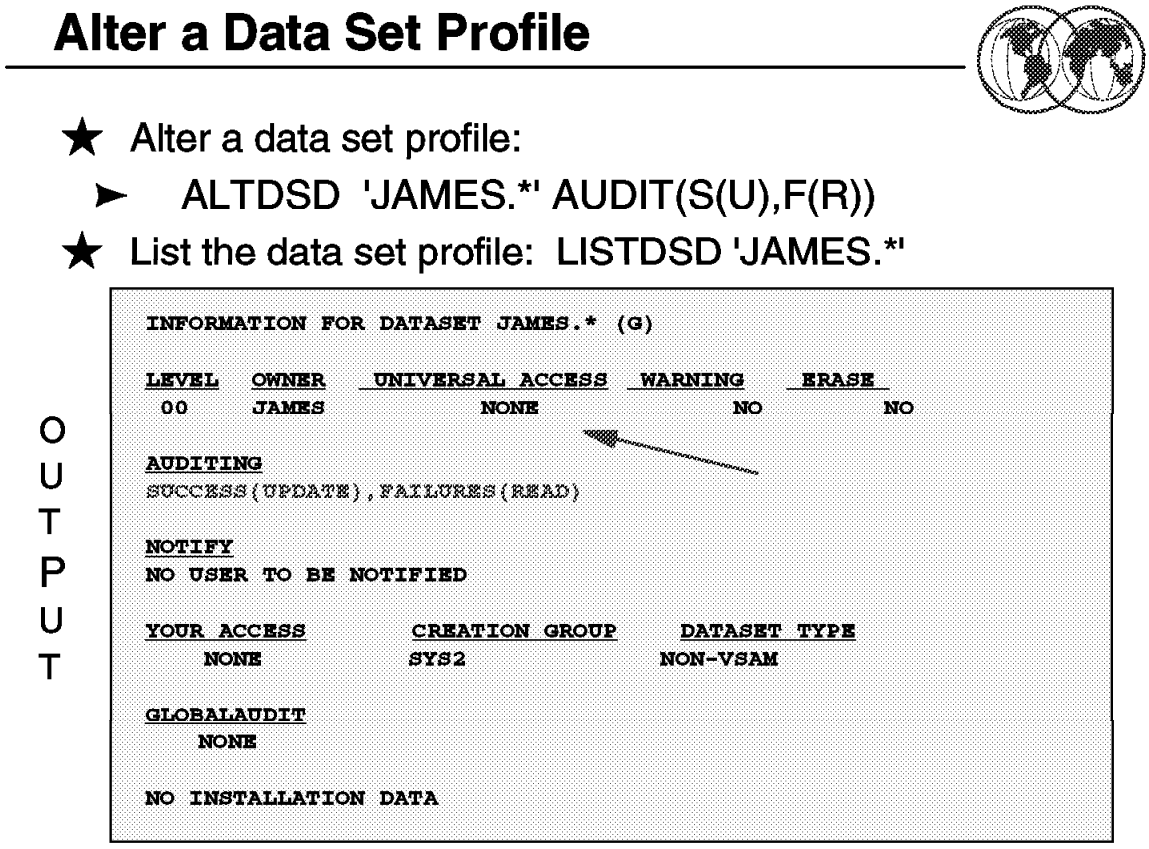


Figure 125. How to alter a data set profile

2.9.12 How to alter a data set profile

Use the ALTDSD command to:

- Modify an existing discrete or generic data set profile.
- Protect a single volume of either a multivolume tape data set or a multivolume, non-VSAM DASD data set. At least one volume must already be RACF-protected.
- Remove RACF-protection from either a single volume of a multivolume tape data set or a single volume of a multivolume, non-VSAM DASD data set. You cannot delete the last volume from the profile.

This visual shows a command to alter the auditing options for the previously data set, JAMES.*

```
ALTDSD'JAMES.*' AUDIT(S(U),F(R))
```

The command also specifies which new access attempts you want to log to the SMP data set.

SUCCESS S(U) Indicates that you want to log authorized accesses to UPDATE

FAILURES F(R) Indicates that you want to log detected unauthorized access attempts to read

The visual shows a sample output from the ALTDSD command, which shows the auditing options as:

```
SUCCESS(UPDATE),FAILURE(READ)
```

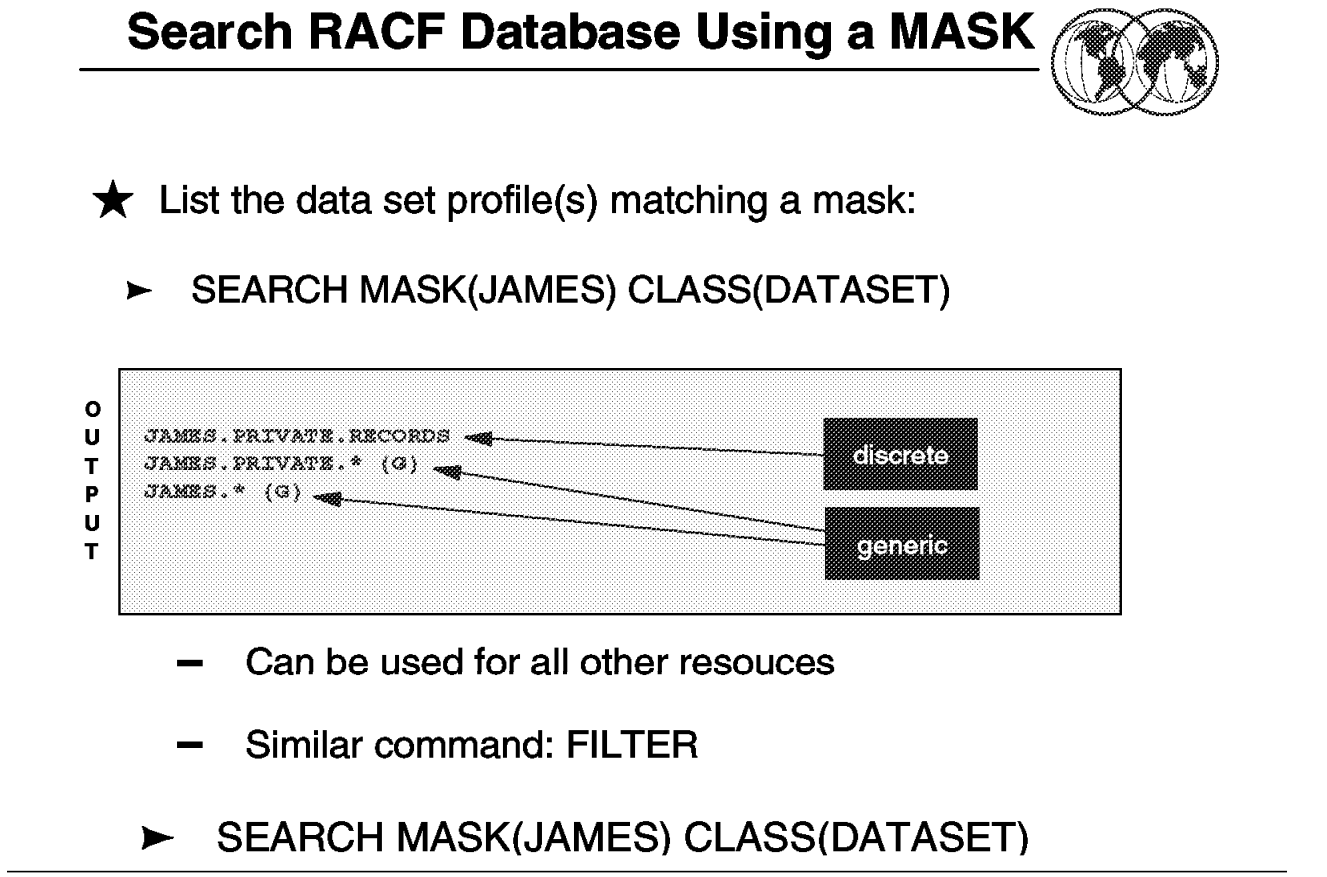


Figure 126. List a data set profile matching a mask

2.9.13 List a data set profile matching a mask

The SEARCH command obtains a list of RACF profiles, users, and groups from the RACF DATABASE using search criteria specified.

MASK specifies the strings of alphanumeric characters used to search the RACF database. This data defines the range of profile names selected. The two-character strings together must not exceed 44 characters for a tape or DASD data set name, or, for general resource classes, the length specified in the class descriptor table.

The visual shows a SEARCH command with the search criteria, MASK.

```
SEARCH MASK(JAMES) CLASS(DATASET)
```

This command allows RACF to list profiles starting with the MASK, in this case JAMES.

A second example allows RACF to list all profiles containing the filter string.

```
SEARCH FILTER(JAMES) CLASS(DATASET)
```

protected by a profile'.

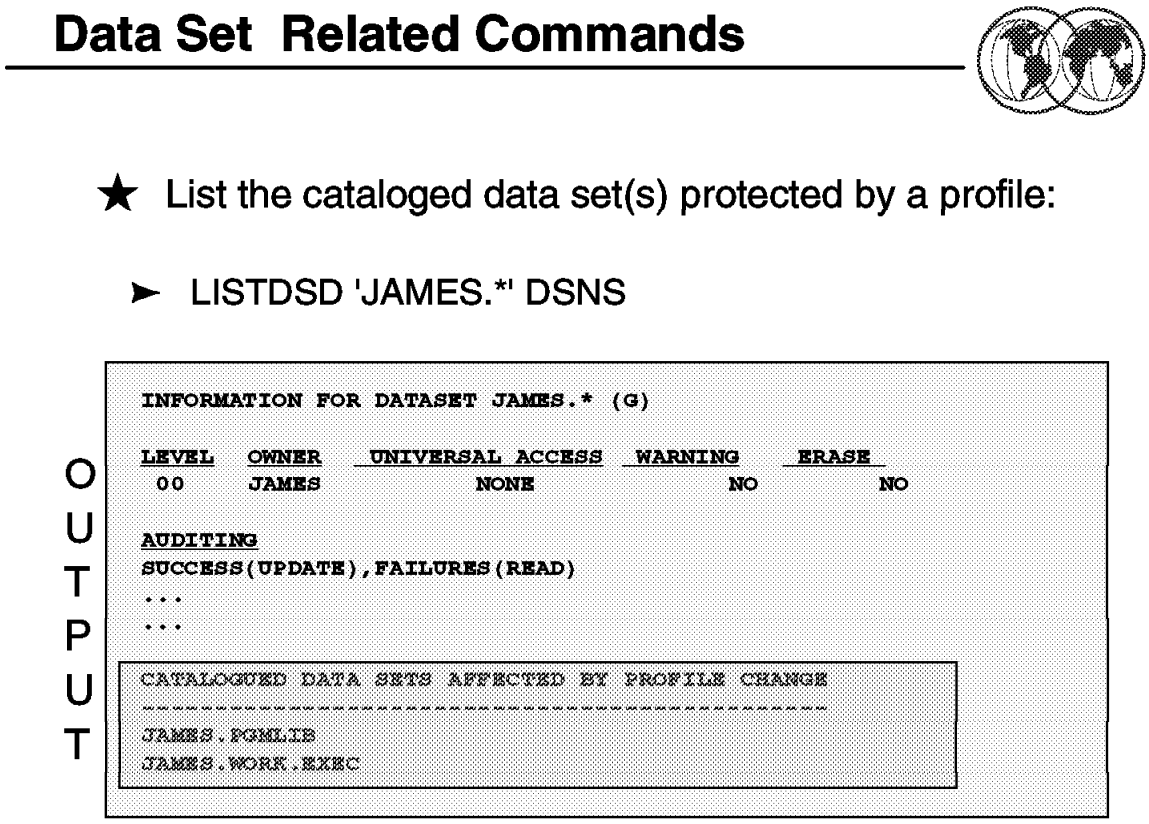


Figure 127. List a catalogued data set

2.9.14 List a catalogued data set

The visual shows sample output from the LISTDSD command:

```
LISTDSD'JAMES.*'DSNS
```

This command allows RACF to list data sets protected by a profile (in this case, the JAMES.* data set profile).

DSNS specifies that you want to list the cataloged data sets protected by the profile specified in the DATASET, ID, or PREFIX operand.

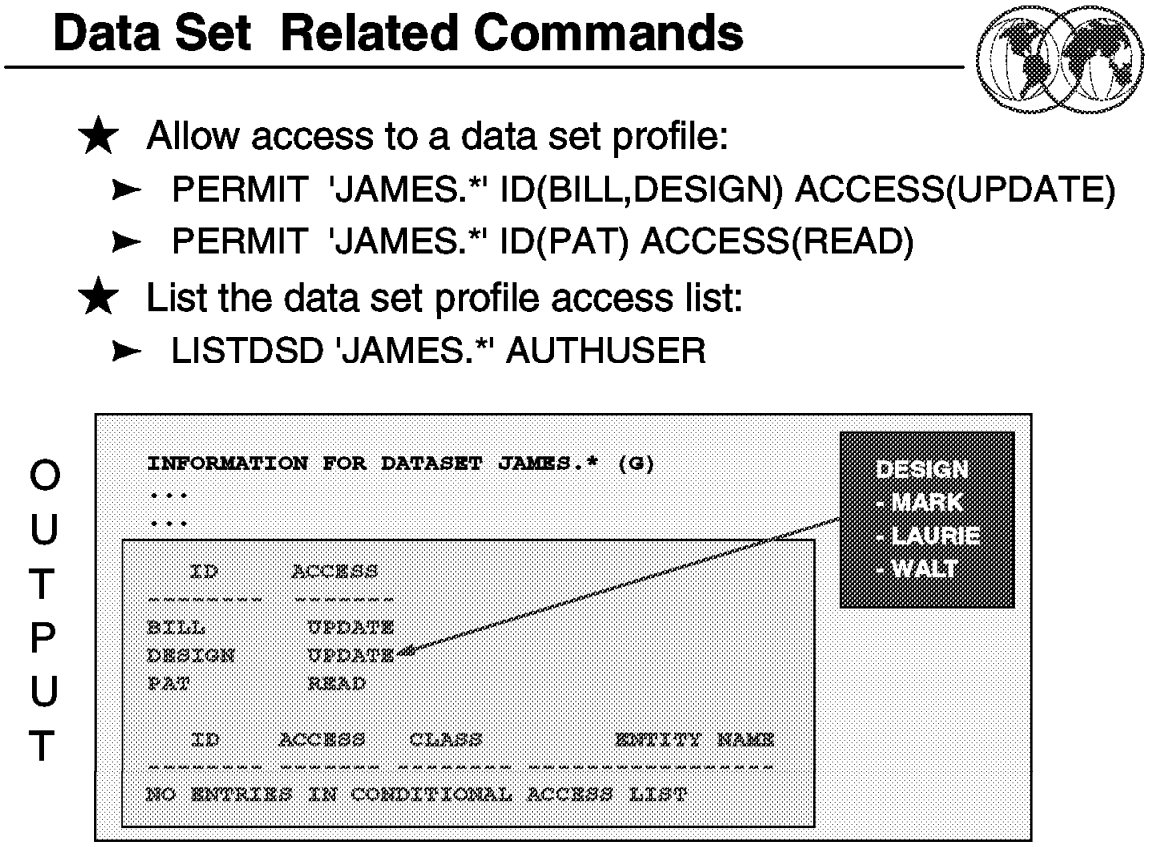


Figure 128. List who has access to a data set profile

2.9.15 List who has access to a data set profile

The visual shows sample output from the PERMIT command:

```
PERMIT 'JAMES.*' ID(BILL,DESIGN) ACCESS(UPDATE)DSNS
```

This command allows Bill and the DESIGN group update access to the files protected by the James.* data set profile. Mark, Laurie, and Walt part of the DESIGN group will have UPDATE access, unless the access list contains their userid with another level of access.

```
PERMIT 'JAMES.*' ID(PAT) ACCESS(READ)DSNS
```

Pat will have read access to the files protected by the JAMES.* profile.

General Resources Related Commands

★ Add a general resource profile:

► RDEF PROGRAM MYMUSIC
ADDMEM('JAMES.PGMLIB'/VOL123/NOPADCHK)

★ List the data set profile: RL PROGRAM MYMUSIC

OUTPUT

```
CLASS      NAME
-----      ----
PROGRAM    MYMUSIC

MEMBER CLASS NAME
------ ----- ----
PMBR

DATA SET NAME                                  VOLSER   PADS CHECKING
---------------------------------------------  ------   -------------
JAMES.PGMLIB                                   VOL123        NO

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    JAMES           NONE             NONE         NO

INSTALLATION DATA
-----------------
NONE
.../...
```

Figure 129. How to add a general resource profile

2.9.16 How to add a general resource profile

The visual shows sample output from the RDEFINE command:

RDEF PROGRAM MYMUSIC ADDMEM('JAMES.PGMLIB'/VOL123/NOPADCHK)

This command defines a new resource profile called MYMUSIC that will run in PROGRAM class. The program MYMUSIC is located in JAMES.PGMLIB member on DASD volume VOL123.

Setting NOPADCHK means that RACF will not check for program-accessed data sets when a user is executing the control programs.

General Resources Related Comands

★ Alter a general resource profile:

► RALT PROGRAM MYMUSIC UACC(READ)

★ List the data set profile: RL PROGRAM MYMUSIC

OUTPUT

```
CLASS       NAME
-----       ----
PROGRAM     MYMUSIC
...
...
DATA SET NAME                          VOLSER   PADS CHECKING
-------------------------------------  ------   -------------
JAMES.PGMLIB                           VOL123         NO

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    JAMES           READ            READ        NO

INSTALLATION DATA
-----------------
NONE
...
```

Figure 130. How to change universal access authority

2.9.17 How to change universal access authority

The visual shows sample output from the RALTER command:

```
RALT PROGRAM MYMUSIC UACC(READ)
```

This command sets the Universal Access Authority (UACC) to read. The UACC is the default access to a resource if the user or group is not specifically permitted access to the resource. The Alter command has set the default access of MYMUSIC at READ.

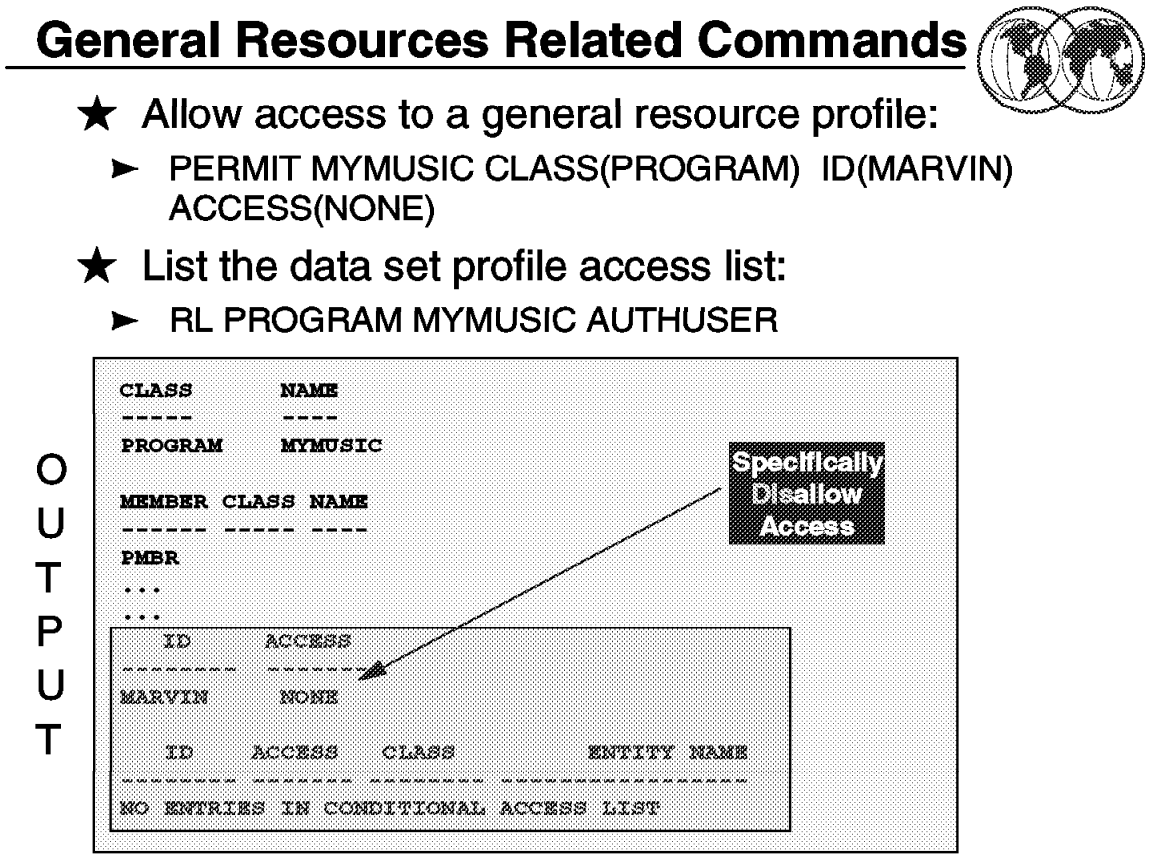


Figure 131. How to permit access to a resource profile

2.9.18 How to permit access to a resource profile

The visual shows sample output from the PERMIT command:

```
PERMIT MYMUSIC CLASS(PROGRAM)  ID(MARVIN) ACCESS(NONE)
```

Despite the UACC(READ) on the resource profile, MARVIN will not be able to access the resource as NONE as specified in the access list.

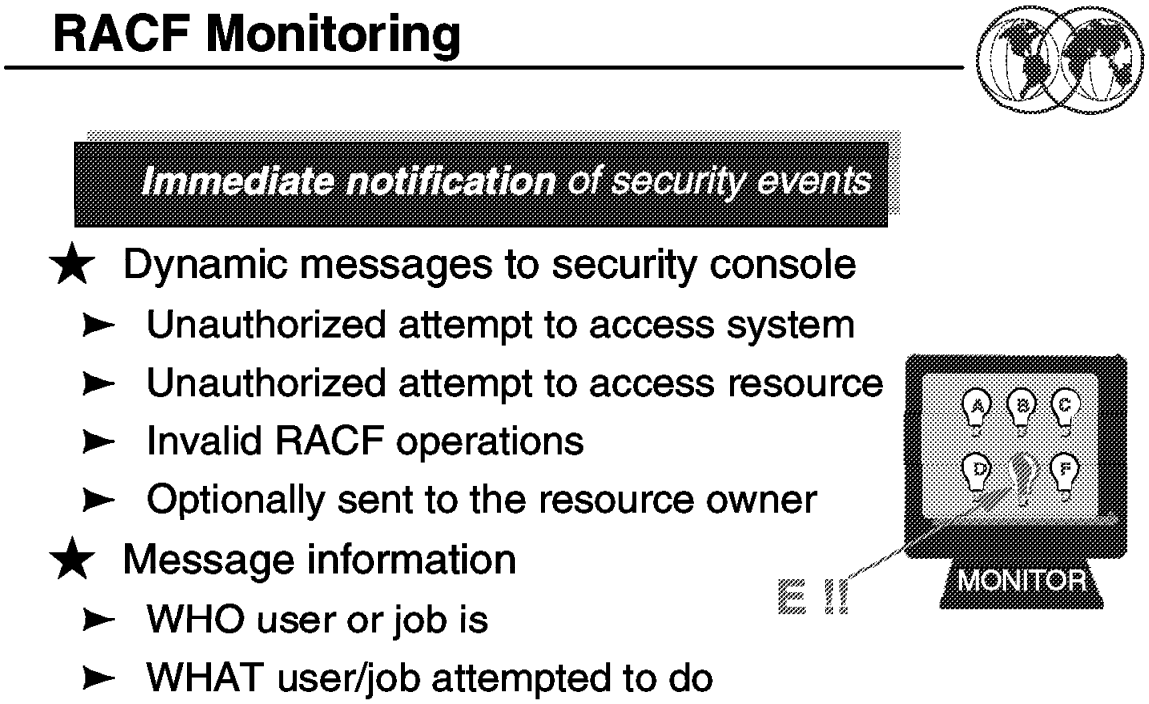


Figure 132. RACF monitoring

2.10 RACF monitoring

For more immediate action, the user can request notification to the master terminal at the time of violation. A non-zero value for the SECCNT keyword of the SECURITY macro causes the master terminal to be notified.

Because the number of violations for a large network may be high due to misspelled passwords, transaction codes, and commands, you can specify a threshold for notification. The master terminal is not notified until the specified number of violations occur without a valid input from a given terminal. You specify 1 to 3 invalid entries as the violation limit. This eliminates or reduces the number of notifications caused merely by operator error, while still providing evidence of real attempts to avoid security safeguards.

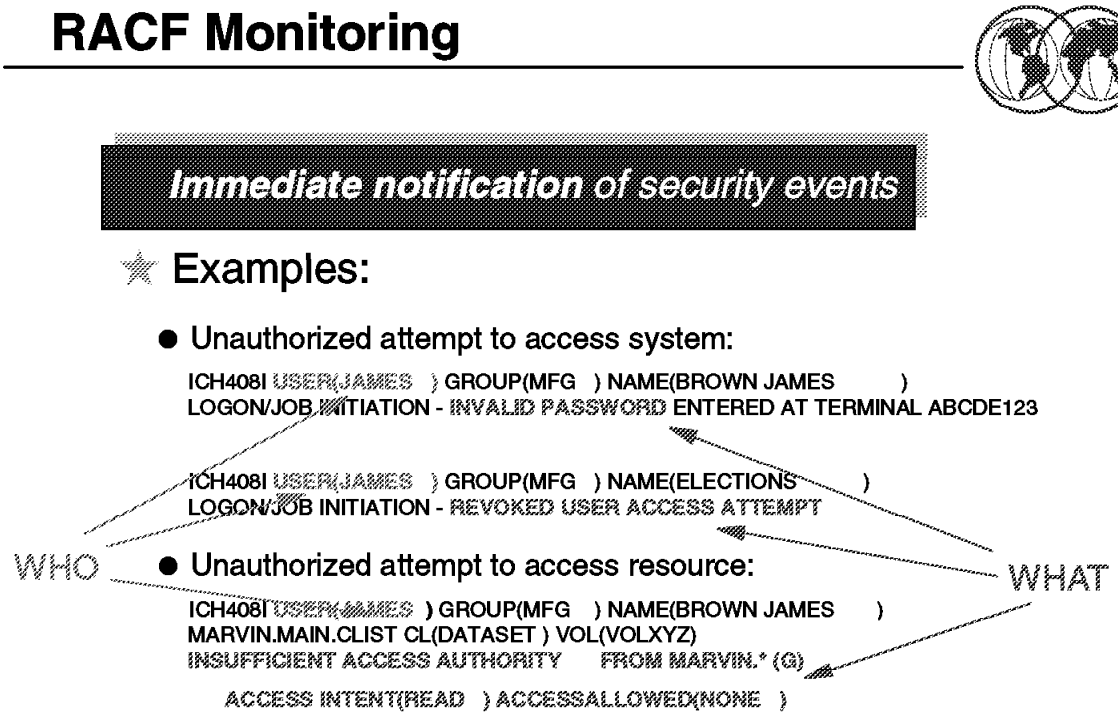


Figure 133. RACF immediate notification - example 1

2.10.1 Example of RACF immediate notification - example 1

The explanation of the RACF message ICH408I is as follows:

```
ICH408I USER(userid) GROUP(group-name) NAME(user-name)
```

This message is issued when RACF detects an unauthorized request (a violation) made by a user or job. The user and group indicated in the first line of the ICH408I message are the execution user ID and group ID under which the job was to run.

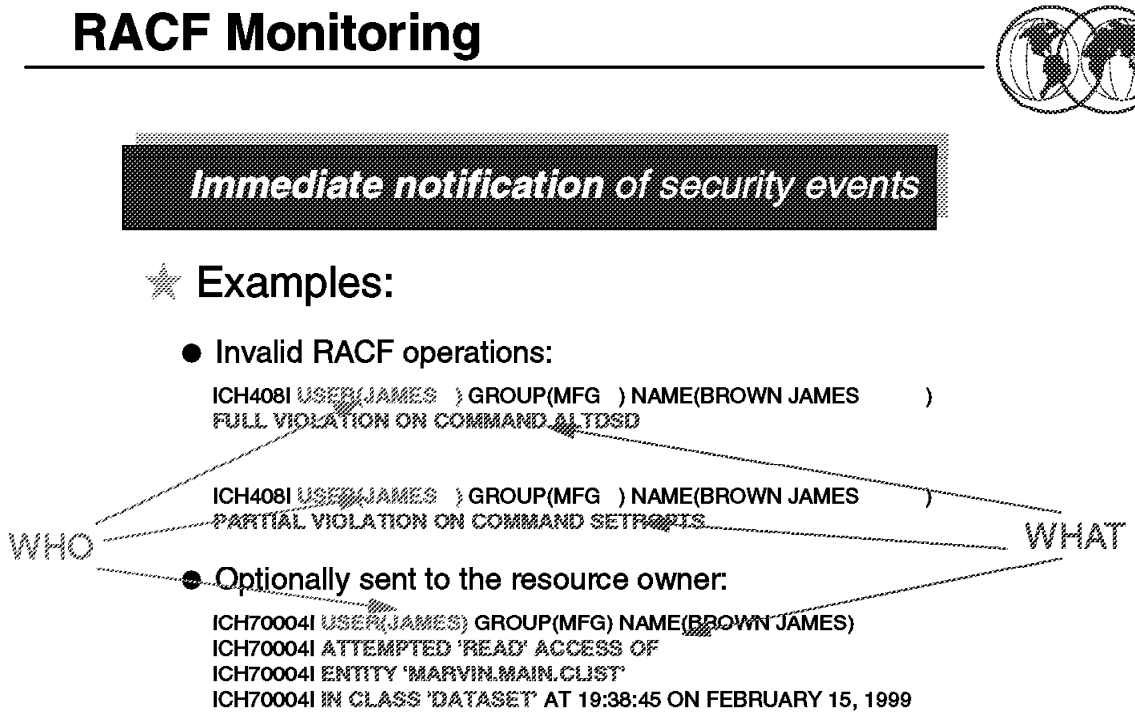


Figure 134. RACF immediate notification - example 2

2.10.2 Example of RACF immediate notification - example 2

The RACF message ICH70004I is as follows:

```
ICH70004I USER(accessor) GROUP(group-name)
NAME(user-name) ATTEMPTED 'access-type' ACCESS
OF ENTITY 'resource-name' IN CLASS 'class-name' AT
hh:mm:ss ON month day, year.
```

This message alerts a RACF user that an access violation has occurred against the indicated resource. This message is routed to the user specified in the NOTIFY field of the resource profile that denied the access.

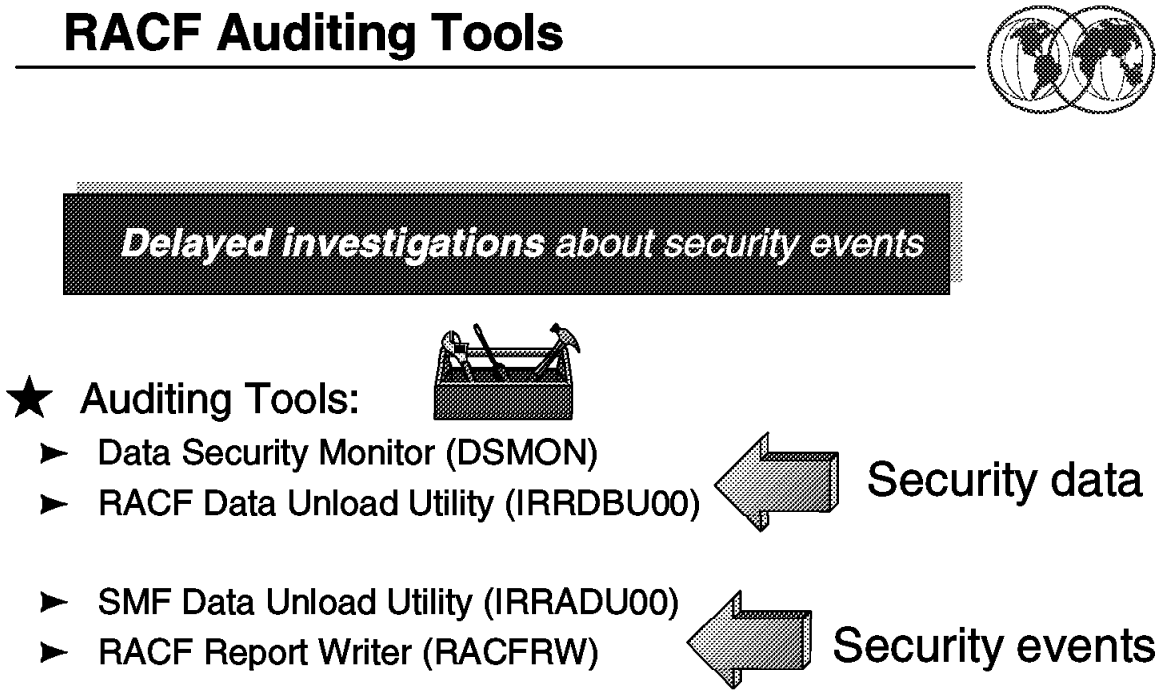


Figure 135. RACF auditing tools

2.11 RACF auditing tools

RACF auditing is basically verifying that the principals set forth by the installations security policy are not compromised. The problem with auditing is being able to reduce the amount of information to something that can be easily analyzed.

Two types of auditing data exist:

1. Security data content from the RACF database, this is a static image or a snapshot of the system parameters at any one time.
2. Security events data statistical information, such as the date, time, and the number of times a specific resource was accessed by any one user.

RACF writes security log records when it detects:

- Unauthorized attempts to enter the system
- Authorized or unauthorized attempts to enter RACF commands
- RACF status changes
- Warning mode resource access attempts
- Failsoft operator access decisions
- Optional authorized or unauthorized attempts to access RACF-protected resources

You can list the contents of these records to help you to detect possible security exposures or threats and verify the security of the system.

Each of the following programs can help you accomplish your goals, depending on your specific needs:

- SMF data unload utility
- RACF data unload utility
- RACF report writer
- Data security monitor (DSMON)

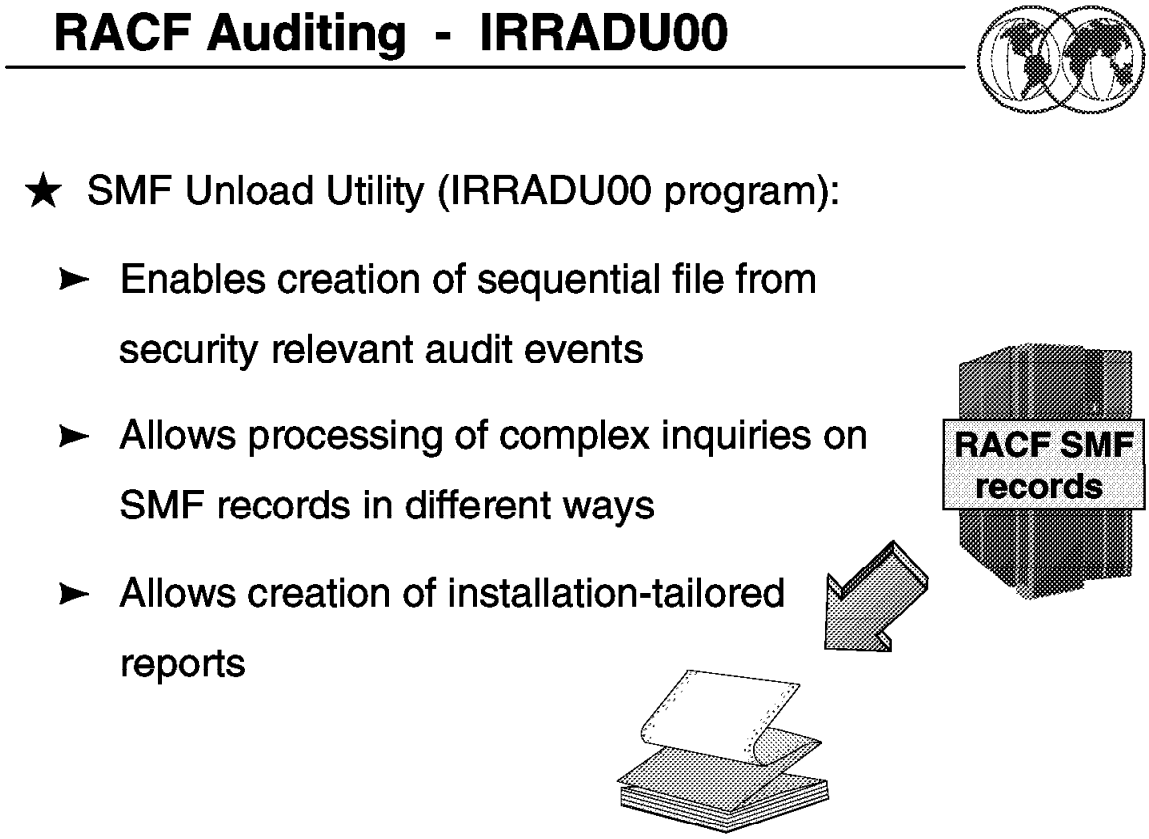


Figure 136. SMF Data Unload Utility

2.11.1 SMF Data Unload Utility (IRRADU00 program)

The SMF data unload utility processes SMF records and permits more complex auditing than the RACF report writer.

Output from the SMF data unload utility can be:

- Viewed directly
- Used as input for installation-written programs
- Manipulated by sort/merge utilities
- Uploaded to a database manager, such as DB2

You can process complex inquiries and generate custom-tailored reports from the output of the SMF data unload utility. These reports can be useful in identifying suspicious patterns of access by authorized users that another program might miss. Because data is more often misused by authorized users than stolen by unauthorized users, reports like this are essential to security.

RACF Auditing

► SMF Unload Utility output example:

```
JOBINIT  SUCCESS  13:31:44 1996-10-08 9672                    ADMIN
JOBINIT  SUCCESS  13:31:46 1996-10-08 9672                    ADMIN
JOBINIT  SUCCESS  13:31:47 1996-10-08 9672                    ADMIN
JOBINIT  TERM     13:31:48 1996-10-08 9672                    ADMIN
JOBINIT  TERM     13:31:48 1996-10-08 9672                    ADMIN
JOBINIT  TERM     13:31:48 1996-10-08 9672                    ADMIN
SETROPTS INSAUTH  13:39:12 1996-10-08 9672 YES  NO   NO   JAMES    MFG      NO   NO   NO   NO
ALTUSER  SUCCESS  13:40:12 1996-10-08 9672 NO   NO   NO   JAMES    MFG      NO   YES  NO   NO
JOBINIT  SUCCESS  13:40:46 1996-10-08 9672                    JAMES    MFG
SETROPTS SUCCESS  13:41:46 1996-10-08 9672 NO   NO   NO   JAMES    MFG      NO   NO   NO   YES
SETROPTS SUCCESS  13:42:17 1996-10-08 9672 NO   NO   NO   JAMES    MFG      NO   YES  NO   YES
DIRSRCH  SUCCESS  13:43:11 1996-10-08 9672 NO   NO   NO   ARTHUR   FINC     YES  NO   NO   NO
DIRSRCH  SUCCESS  13:43:11 1996-10-08 9672 NO   NO   NO   ARTHUR   FINC     YES  NO   NO   NO
DIRSRCH  NOTAUTH  13:43:11 1996-10-08 9672 YES  NO   NO   ARTHUR   FINC     YES  NO   NO   NO
CHKFOWN  OWNER    13:44:28 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
CHMOD    SUCCESS  13:44:28 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
CHOWN    SUCCESS  13:44:28 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
FACCESS  SUCCESS  13:44:28 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
JOBINIT  SUCCESS  13:44:30 1996-10-08 9672                    REBECCA
SETEUID  SUCCESS  13:44:31 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
DIRSRCH  SUCCESS  13:44:31 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
DIRSRCH  SUCCESS  13:44:31 1996-10-08 9672 NO   NO   NO   REBECCA  SYS1     NO   NO   NO   NO
```

SMF Unload Utility

Figure 137. How to run the SMF Data Unload Utility

2.11.2 How to run the SMF Data Unload Utility (IRRADU00)

Following is RACF SMF Data Unload Utility sample JCL:

```
//KHEWITT1 JOB (ITSO),'SMF FLAT',MSGCLASS=X
//SMFDUMP  EXEC PGM=IFASMFDP
//SYSPRINT DD  SYSOUT=*
//ADUPRINT DD  SYSOUT=*
//OUTDD    DD  DISP=(NEW,CATLG),DSN=KHEWITT.RACF.IRRADU00,
//         UNIT=SYSDA,SPACE=(CYL,(10,5),RLSE),
//         LRECL=5096,RECFM=VB
//SMFDATA  DD  DISP=SHR,DSN=SYS1.SC42.MAN2
//SMFOUT   DD  DUMMY
//SYSIN    DD  *
     INDD(SMFDATA,OPTIONS(DUMP))
     OUTDD(SMFOUT,TYPE(000:255))
     ABEND(NORETRY)
     USER2(IRRADU00)
     USER3(IRRADU86)
//SYSIN    DD DUMMY
```

To display the active SMF data set, use the D SMF command from the system console.

```
IEE974I 10.12.27 SMF DATA SETS 796
          NAME                VOLSER SIZE(BLKS) %FULL  STATUS
        P-SYS1.SC42.MAN1      MVS004      1200     0  ALTERNATE
        S-SYS1.SC42.MAN2      MVS004      1200    86  ACTIVE
        S-SYS1.SC42.MAN3      MVS004      1200     0  ALTERNATE
```

MAN2 is the active SMF data set.

The output file in this example is KEWITT.RACF.IRRADU00.

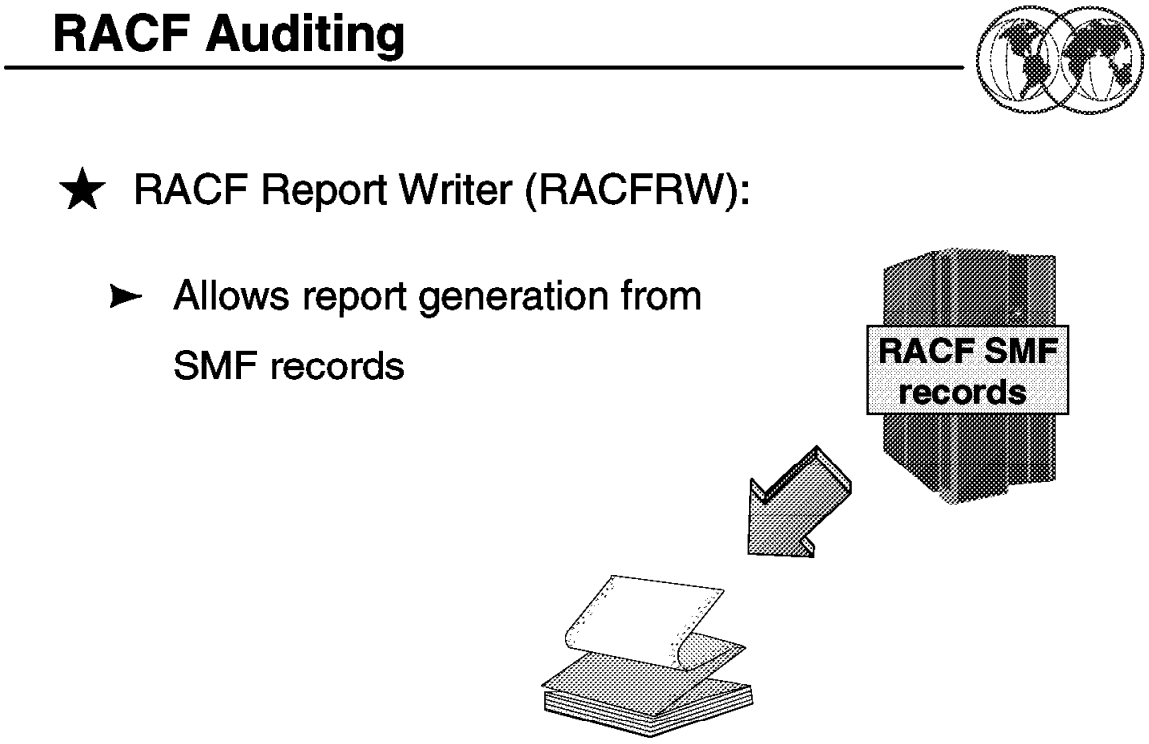


Figure 138. RACF report writer

2.12 RACF report writer

The RACF report writer lists the contents of system management facility (SMF) records in a format that is easy to read. It also uses the same SMF data to generate the following specialized reports:

- Reports that describe attempts to access a particular RACF-protected resource in terms of user identity, number and type of successful accesses, and number and type of attempted security violations.
- Reports that describe user and group activity.
- Reports that summarize system use and resource use.

The RACF report writer is stabilized at the RACF 1.9.2 level, and is not able to report on many of the later RACF functions.

RACF Auditing

► RACF Report Writer report example:

```
199.049 00:24:29                                                RACF REPORT
0                                                    ACCESSES TO MYMUSIC  PROGRAM
0COMMAND GROUP ENTERED -
  RACFRW                LINECNT(60)                         FORMAT                          GENSUM
   SELECT PROCESS
   EVENT ACCESS CLASS(PROGRAM) NAME(MYMUSIC )
   LIST TITLE('LIST ACCESSES TO MYMUSIC  PROGRAM')
   SUMMARY RESOURCE BY(USER)   TITLE('SUMMARY BY USER')
 END
                                                                                    .../...
```

RACF Report Writer

Figure 139. How to run RACF report writer

2.12.1 How to run RACF report writer

Following is RACF Report Writer sample JCL:

```
//KHEWITT1 JOB (ITSO),'RACF FLAT',MSGCLASS=X
//HEWITT   EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSTSIN  DD *
RACFRW DSNAME('KHEWITT.RACF.IRRADU00')
LIST
SUMMARY USER
END
/*
```

The RACF report writer can also be run from the TSO command line.

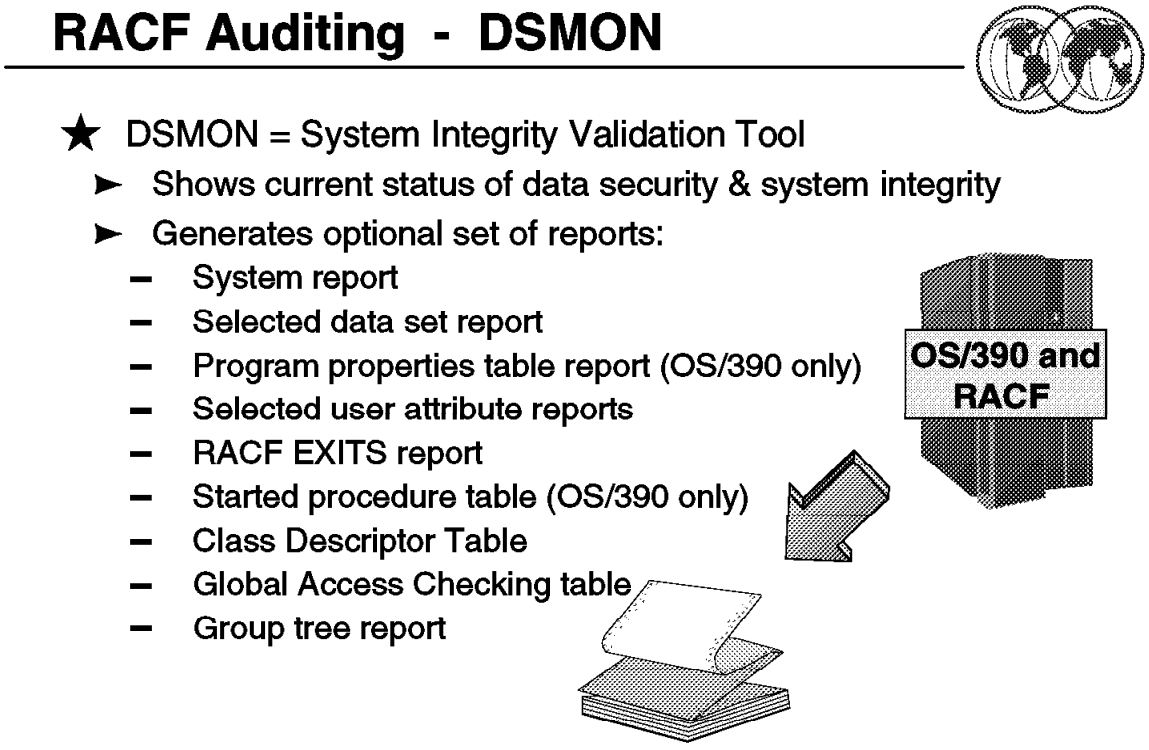


Figure 140. The RACF Data Security Monitor

2.13 RACF Data Security Monitor

The RACF data security monitor (DSMON) enables you to verify the basic system integrity and data security controls.

RACF auditors can use the DSMON reports to evaluate the level of security at the installation and to compare the actual level of security at an installation with the planned level of security.

2.13.1.1 DSMON reports

DSMON produces the following reports:

- System report
- Group tree report
- Program properties table report
- RACF authorized caller table report
- RACF class descriptor table report
- RACF EXIT report
- RACF global access checking table report
- RACF user attribute report
- RACF user attribute report summary report

- RACF Selected data set report

2.13.1.2 The System Report

The system report contains information such as the identification and model of the processor complex, and the name, version, and release of the operating system. This report also specifies the RACF version and release number and whether RACF is active. If RACF is inactive, DSMON prints a message that tells you whether RACF was not activated at IPL or was deactivated by the RVARY command.

2.13.1.3 The Group Tree Report

This report lists, for each requested group, all of its subgroups, all of the subgroups′ subgroups, and so on, as well as the owner of each group listed in the report, if the owner is not the superior group. You can use the group tree report to examine the overall RACF group structure for your system. You can also determine the scope of the group for group related user attributes (group SPECIAL, group OPERATIONS, and group AUDITOR).

2.13.1.4 The Program Properties Table Report

This report lists all of the programs in the MVS program properties table (PPT). The report also indicates, for each program, whether the program is authorized to bypass password protection and whether it runs in a system key.

You can use the program properties table report to verify that only those programs that the installation has authorized to bypass password protection are, in fact able to do so. Such programs are normally communication and database control programs and other system control programs.

You can also verify that only those programs that the installation has authorized are able to run in a system key.

2.13.1.5 The RACF Authorized Caller Table Report

This report lists the names of all of the programs in the RACF authorized-caller table. The programs in this table are authorized to issue REQUEST=VERIFY (which performs user verification) or REQUEST=LIST (which loads profiles into main storage).

You can use this report to verify that only those programs that are supposed to be authorized to modify an ACEE (accessor environment element) are able to issue a REQUEST=VERIFY. This verification is a particularly important security requirement because the ACEE contains a description of the current user. This description includes the user ID, the current connect group, the user attributes, and the group authorities. A program that is authorized to issue REQUEST=VERIFY could alter the ACEE to simulate any user.

You can also use this report to verify that only those programs that are supposed to be authorized to access profiles are able to issue REQUEST=LIST. Because profiles contain complete descriptions of the characteristics that are associated with RACF-defined entities, you must carefully control access to them.

2.13.1.6 The RACF Class Descriptor Table Report

This report lists, for each general resource class the class name, the default UACC, whether the class is active, whether auditing is being done, whether statistics are being kept, and whether OPERATIONS attribute users have access.

You can use the class descriptor table report to determine which classes (besides DATASET) are defined to RACF and active, and therefore can contain resources that RACF protects.

2.13.1.7 The RACF Exits Report

This report lists the names of all of the installation-defined RACF exit routines and specifies the size of each exit routine module.

You can use the RACF exits report to verify that the only active exit routines are those that your installation has defined. The existence of any other exit routines might indicate a system security exposure, because RACF exit routines can be used to bypass RACF security checking. Similarly, if the length of an exit routine module differs from the length of the module when it was defined by your installation, the module might have unauthorized modifications.

2.13.1.8 The RACF Global Access Checking Table Report

This report lists, for each resource class in the global access table, all of the entry names and their associated resource access authorities.

Because global access checking allows anyone to access the resource at the associated access authority, you should verify that each entry has an appropriate level of access authority.

The RACF Started Procedures Table Reports RACF generates two started procedures table reports.

If the STARTED class is active, the report uses the STARTED class profiles and contains the TRACE attribute. The trace uses module ICHDSM00.

If the STARTED class is not active, the trace uses the installation replaceable load module, ICHRIN03.

The reports list the procedure name, the user ID and group name to be associated with the procedure, and whether the procedure is privileged or trusted.

You can use the report to determine which started procedures are defined to RACF, and which have the privileged attribute. If a started procedure is privileged or trusted, it bypasses all REQUEST=AUTH processing (unless the CSA or PRIVATE operand was specified on REQUEST=AUTH), including checks for security classification of users and data.

2.13.1.9 Selected User Attribute Report

The selected user attribute report:

Lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, or REVOKE attributes

Specifies whether they possess these attributes on a system-wide (user) or group level

Indicates whether they have any user ID associations

You can use this report to verify that only those users who need to be authorized to perform certain functions have been assigned the corresponding attribute.

2.13.1.10 Selected User Attribute Summary Report

The selected user attribute summary report shows the number of installation-defined users and totals for users with the SPECIAL, OPERATIONS, AUDITOR, and REVOKE attributes, at both the system and group level. You can use this report to verify that the number of users with each of these attributes, on either a system or group level, is the number that your installation wants. In particular, you should make sure that you have assigned the SPECIAL attribute (on a system level) to at least one user and the AUDITOR attribute (on a system level) to at least one user.

2.13.1.11 Selected Data Sets Report

This report lists the names of selected system data sets and, for each data set, specifies the criterion for selection, the serial number of the volume on which it resides, whether the data set is RACF-indicated or RACF-protected, and the universal access authority (UACC). If a data set meets more than one selection criterion, there is a separate entry in the report for each criterion. The selected data sets include system data sets, the MVS master catalog, user catalogs, the RACF primary and backup data sets, and user-specified data sets.

You can use the selected data sets report to determine which of these data sets are protected by RACF and which are not. You can also check whether the UACC associated with each of the data sets is compatible with your installation′s resource access control requirements.

RACF Auditing

► DSMON report example:

```
1RACF DATA SECURITY MONITOR                                                    DATE: 02/17/99
-                                          S Y S T E M     R E P O R T
0-------------------------------------------------------------------------------------------
0CPU-ID                              046293
0CPU MODEL                           9672
0OPERATING SYSTEM/LEVEL              MVS/ 3.8   SP6.0.4 JBB6604
0SYSTEM RESIDENCE VOLUME             RES123
0SMF-ID                              PROD
0RACF VERSION 2 RELEASE  4.0 IS ACTIVE
1RACF DATA SECURITY MONITOR                                                    DATE: 02/17/99
0                          P R O G R A M     P R O P E R T I E S     T A B L E     R E P O R T
0PROGRAM          BYPASS PASSWORD          SYSTEM
 NAME             PROTECTION               KEY
 -------------------------------------------------------------------------------------------
0IEDQTCAM              NO                  YES
0ISTINM01              YES                 YES
0IKTCAS00              YES                 YES
                                                                                  .../...
```

RACF DSMON

Figure 141. How to run DSMON

2.13.2 How to run the DSMON program

DSMON runs as an authorized program facility (APF) authorized batch program. DSMON can also be run on TSO if SYS1.PARMLIB(IKJTSO00) is configured correctly.

Following is sample DSMON JCL:

```
//P390S  JOB 1,P390,MSGCLASS=X
//TSOBAT01 EXEC PGM=ICHDSM00
//SYSPRINT DD   SYSOUT=*
//SYSUT2   DD   SYSOUT=*
//SYSIN    DD *
  LINECOUNT 55
  FUNCTION all
  USEROPT USRDSN  sivle.memo.text
/*
```

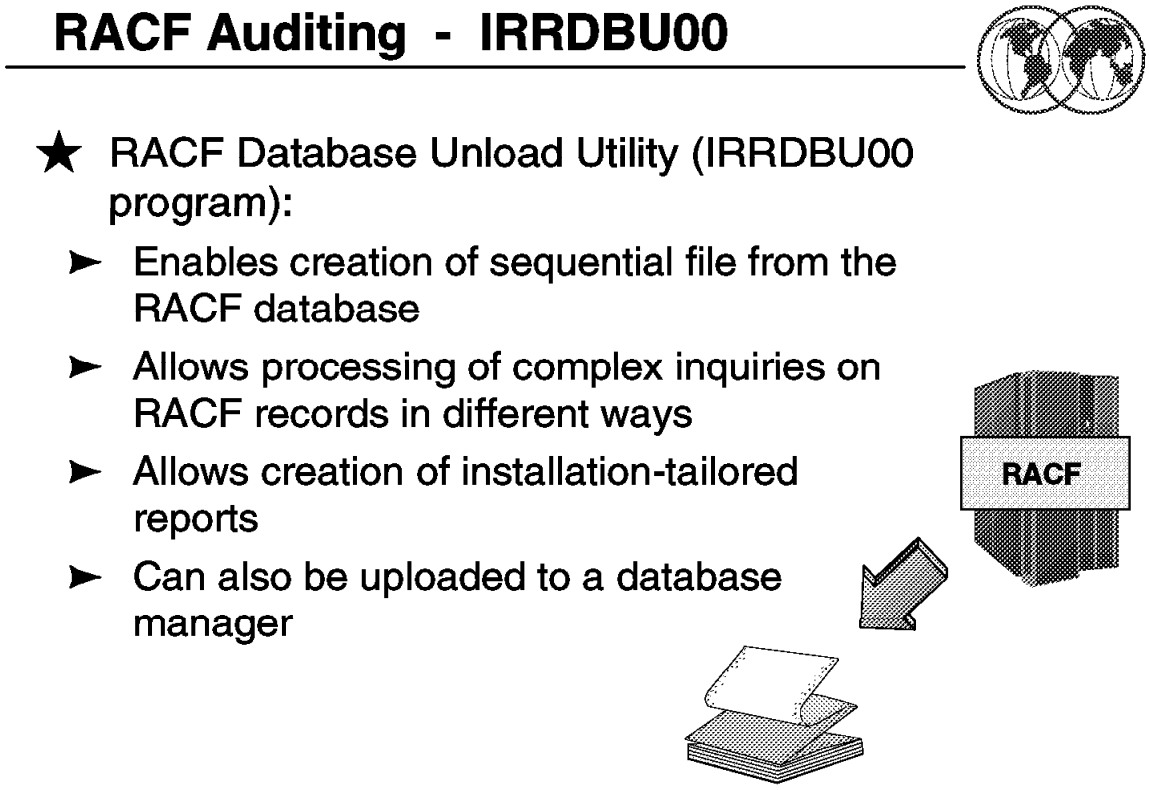


Figure 142. RACF Database Unload Utility

2.14 RACF Database Unload Utility

The RACF Database Unload Utility (IRRDBU00 program) is used to unload data from the RACF database (except password fields) into a flat file.

The output file from the database unload utility can be:

- Viewed directly
- Used as input to your own programs
- Manipulated with sort/merge utilities
- Used as input to a database management system

Installations can produce reports that are tailored to their requirements.

Using the Database Unload Utility Output with DB2 you can use the DB2 Load Utility or its equivalent to process the records produced by the database unload utility. The definition and control statements for a DB2 to use this output, are shipped with OS/390 in the SYS1.SAMPLIB.

RACF Auditing

- RACF DBU output example:
  - 01xx GROUPrecords

```
0101 DIVISB    POU
0101 DIVISB    KGN
0100 DIVISB    SYS1      1999-02-17 ADMGRPS  NONE      NO
0101 POU       DESIGN
0101 POU       TEST
0101 POU       MFG
0100 POU       DIVISB    1999-02-17 ADMGRPS  NONE      NO
0102 DESIGN    MARK      USE
0102 DESIGN    LAURIE    USE
0102 DESIGN    WALT      USE
0100 DESIGN    POU       1999-02-17 ADMGRPS  NONE      NO
0101 SYS1      DIVISA
0101 SYS1      DIVISB
0103 SYS1      IDENTITY  SYSTEM
0103 SYS1      ATTRIB    JOBNAMEXBYPASSPW
0100 SYS1                1999-01-01 IBMUSER  NONE      NO
```

RACF DBU flat file

Figure 143. How to run the RACF Data Unload Utility

2.14.1 How to run IRRDBU00

Following is RACF Data Unload Utility sample JCL:

```
//KHEWITT1 JOB (ITSO),'RACF FLAT',MSGCLASS=X
//UNLOAD   EXEC PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD  SYSOUT=*
//INDD1    DD  DISP=SHR,DSN=SYS1.RACFESA
//OUTDD    DD  DISP=(NEW,CATLG),DSN=KHEWITT.RACFDB.FLATFILE,
//         UNIT=SYSDA,SPACE=(CYL,(70,10),RLSE),
//         LRECL=4096,RECFM=VB
//SYSIN    DD DUMMY
```

The output file name in this example is KEWITT.RACF.IRRADU00. To display the active SMF data set, use the command D SMF.

Chapter 3. OS/390 UNIX System Services

The OS/390 support for OS/390 UNIX System Services (OS/390 UNIX) enables two open systems interfaces on the OS/390 operating system:

- An application program interface (API)

  The application interface is composed of C interfaces. Some of the C interfaces are managed within the C Run-Time Library (RTL), and others access kernel interfaces to perform authorized system functions on behalf of the unauthorized caller.

  With the API, programs can run in any environment, which includes batch jobs, jobs submitted by TSO/E users, most other started tasks, and in any other MVS application task environment. A program can request:

  - Only MVS services
  - Only OS/390 UNIX
  - Both MVS and OS/390 UNIX

- An interactive OS/390 shell interface

  The OS/390 shell is modeled after the UNIX System V shell with some of the features found in the Korn Shell. As implemented for OS/390 UNIX services this shell conforms to POSIX standard 1003.2, which has been adopted as ISO/IEC International Standard 9945-2: 1992.

  The shell is a command processor that you use to:

  - Invoke shell commands or utilities that request services from the system.
  - Write shell scripts using the shell programming language.
  - Run shell scripts and C-language programs interactively (in the foreground), in the background, or in batch.
  - The OS/390 shell provides commands and utilities that give the user an efficient way to request a range of services.

Product and Component Support for OS/390 UNIX Services

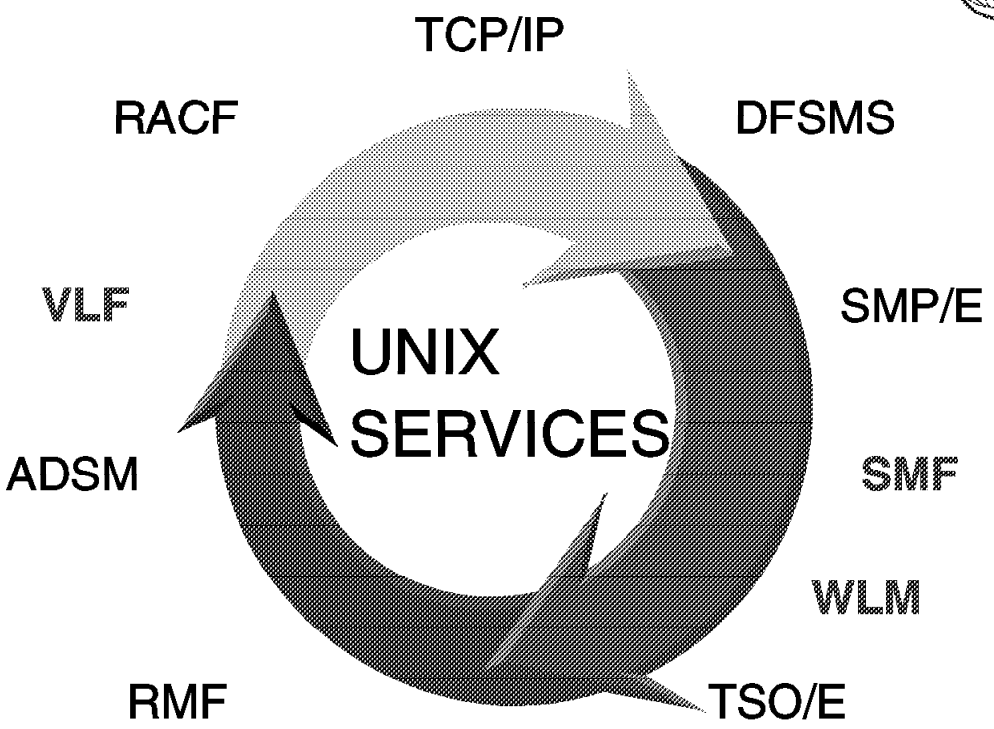


Figure 144. Component support for UNIX services

3.1 Products and components with OS/390 UNIX

This visual shows the products, (RACF, TCP/IP, DFSMS, ADSM, RMF, TSO/E, and SMP/E) and components (VLF, SMF, and WLM) used with OS/390 UNIX System Services.

OS/390 components

WLM The Work Load Manager (WLM) transaction initiators provide address spaces when programs issue the fork() or spawn() C function or OS/390 callable service.

VLF RACF allows caching of UID and GID information in the Virtual Lookaside Facility (VLF). Add the following VLF options to the COFVLFxx member of SYS1.PARMLIB to enable the caching:

```
CLASS NAME(IRRUMAP)
  EMAJ(UMAP)
CLASS NAME(IRRGMAP)
  EMAJ(GMAP)
```

This will improve the performance of OS/390 UNIX.

SMF System Management Facility (SMF) collects data for accounting. SMF job and job-step accounting records identify processes by user, process, group, and session identifiers. Fields in these records also provide information on resources used by the process. SMF File System records describe file system events such as file open, file close, and file system mount, unmount, quiesce, and unquiesce.

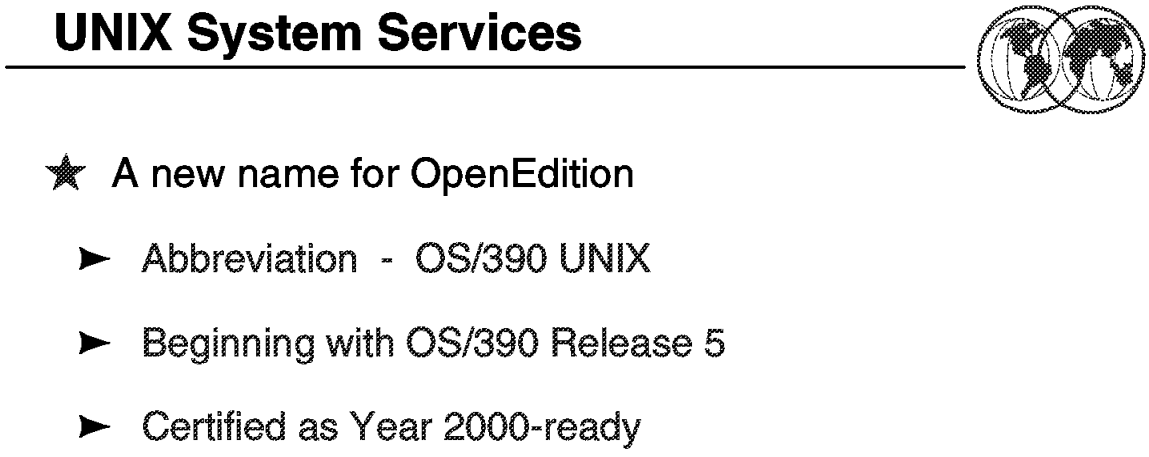


Figure 145. UNIX System Services

3.2 UNIX System Services

The name OpenEdition was changed to OS/390 UNIX System Services beginning with OS/390 Release 5. UNIX System Services can be abbreviated OS/390 UNIX. OS/390 and OS/390 UNIX are Year 2000 ready and require no migration action.

For more information about Year 2000 support, see *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, and the Web site `http://www.ibm.com/year2000/`.

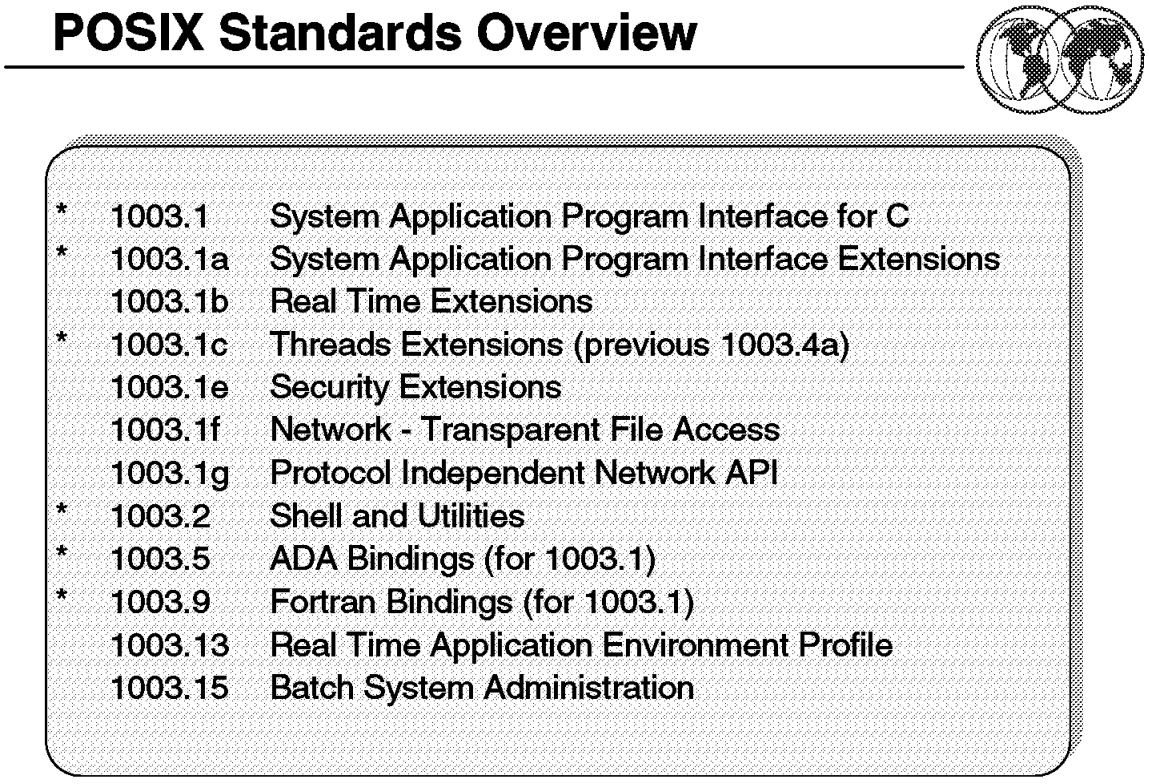


Figure 146. POSIX standards overview

3.3 POSIX standards overview

The work on Portability Operating Systems Interface (POSIX) started as an effort to standardize UNIX and was performed by a workgroup under the Institute of Electrical and Electronical Engineers (IEEE). What they defined was an application programming interface which could be applied not only to UNIX systems but to other operating systems like MVS.

POSIX is not a product. It is an evolving family of standards describing a wide spectrum of operating system components ranging from C language and shell interfaces to system administration.

The POSIX standard is sponsored by the International Organization for Standardization (ISO) and is incorporated into X/Open Portability Guides (XPG). Each element of the standard is defined by a 1003.* number.

POSIX defines the interfaces and not the solution or implementation. In this way POSIX can be supported by any operating system. Implementation of POSIX can be different in areas such as performance, availability, recoverability, and so on. All POSIX compliant systems are not the same although they all support basically the same interface.

Many people think open means UNIX. Although POSIX evolved from UNIX, it's not the same as UNIX.

The support for open systems in OS/390 is based on the POSIX standard. POSIX is a part or a subset of the XPG standard defined by X/Open.

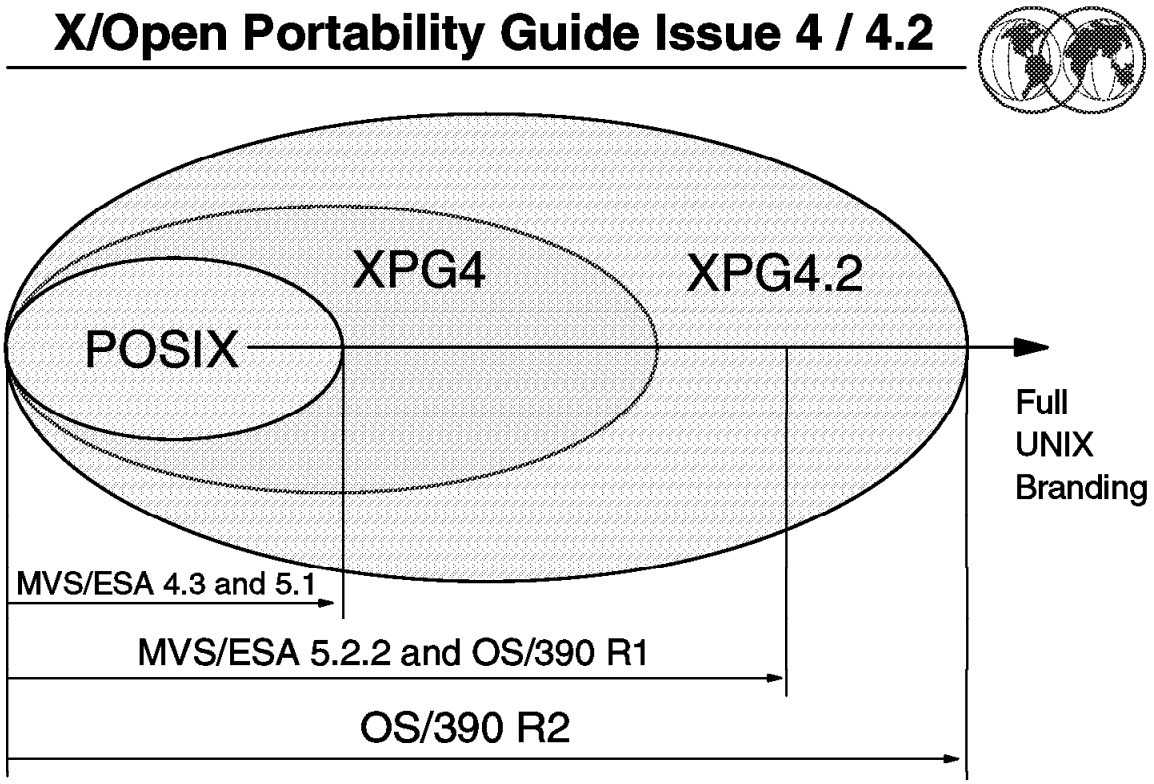


Figure 147. X/Open Portability Guide Issue 4/4.2

3.4 X/Open Portability Guide

OS/390 UNIX was originally implemented in MVS/ESA 4.3 as OpenEdition and supported the POSIX standards (1003.1, 1003.1a, 1003.1c, and 1003.2) with approximately 300 functions. In MVS/ESA 5.2.2 many additional functions were added to meet the XPG4 requirements.

In MVS/ESA 5.2.2 the number of functions included in the OpenEdition implementation was more than 1100. This incorporated the full X/Open Portability Guide issue 4 (XPG4) and over 90 percent of the single UNIX specification as defined in XPG4.2. The remaining functions were added afterwards and so OpenEdition became branded as a UNIX system.

The XPG4.2 support includes all commands and utilities, most of the additional C services defined in the standard and curses, which was included in specification 1170 but not in the XPG4.2 itself. Curses is the UNIX multicolor, multi-language screen control package which comes from the Novell SVID Edition 3 package.

Since OS/390 R2 the following items were added: STREAMS, X/Open Transport Interface (XTI), XPG4.2 regular expressions, XPG4.2 context switching, and XPG4.2 behavior specific to sockets.

Since 1992, multiple standards organizations have been involved in tasks to standardize UNIX and provide a set of open systems interface specifications. Many of these organizations cooperated and many vendors fully supported their efforts to create a UNIX-based set of application programming interfaces.

MVS started out to be a POSIX compliant system. This support has been extended to support the X/Open definitions of the X/Open Portability Guide issue 4 and 4.2.

The advantage of this goal was the portability of programs developed to these standards and the sharing of development across heterogeneous platforms. Also included in XPG4.2 were the protocols and methods for communicating between programs on different platforms. This then addressed both, the issues of portability and the interoperability which was defined earlier in this topic as the requirement of open systems.

Since 1996, OS/390 R2 has been UNIX branded.

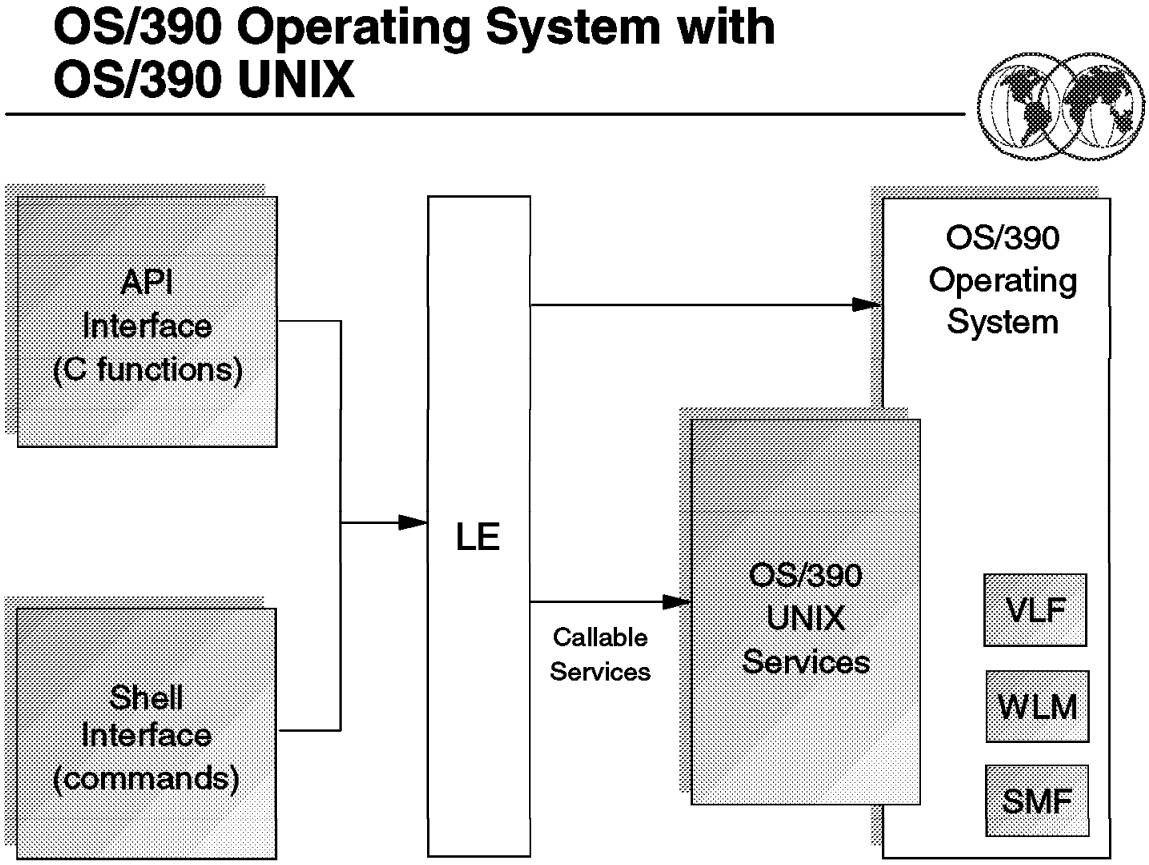


Figure 148. OS/390 operating system with OS/390 UNIX

3.5 OS/390 operating system with OS/390 UNIX

The OS/390 support for OS/390 UNIX System Services (OS/390 UNIX) enables two open systems interfaces on the OS/390 operating system:

- An application program interface (API)
- An interactive OS/390 shell interface

With the APIs, programs can run in any environment—including in batch jobs, in jobs submitted by TSO/E users, and in most other started tasks—or in any other MVS application task environment. The programs can request:

- Only MVS services
- Only OS/390 UNIX
- Both MVS and OS/390 UNIX

The shell interface is an execution environment analogous to TSO/E, with a programming language of shell commands analogous to the Restructured eXtended eXecutor (REXX) language. The shell work consists of:

- Programs run by shell users
- Shell commands and scripts run by shell users
- Shell commands and scripts run as batch jobs

To run a shell command or utility, or any user-provided application program written in C or C++, you need the C/C++ run-time library provided with the Language Environment (LE).

With the OS/390 UNIX System Services Application Services, users can:

- Request services from the system through shell commands. Shell commands are like TSO/E commands.
- Write shell scripts to run tasks. Shell scripts are analogous to REXX EXECs.
- Run programs interactively (in the foreground) or in the background.

Many users use similar interfaces on other systems, such as AIX for the RISC System/6000 or UNIX, and use terminology different from OS/390 terminology. For example, they refer to virtual storage as memory. The work done by their system administrators is handled by system programmers in an OS/390 system. To help you understand these users, this book and its glossary indicate equivalent terms and phrases.

Application programmers are likely to do the following tasks when creating UNIX-compliant application programs:

- Design, code, and test the programs on their workstations using XPG4 UNIX-conforming systems.
- Send the source modules from the workstation to OS/390.
- Copy the source modules from the OS/390 data sets to HFS files.
- Compile the source modules and link-edit them into executable programs.
- Test the application programs.
- Use the application programs.

An OS/390 UNIX program can be run interactively from a shell in the foreground or background, run as an OS/390 batch job, or called from another program. The following types of applications exist in an OS/390 system with OS/390 UNIX:

- Strictly conforming XPG4 UNIX-conforming applications
- Applications using only kernel services
- Applications using both kernel and OS/390 services
- Applications using only OS/390 services

An OS/390 program submitted through the job stream or as a job from a TSO/E session can request kernel services through the following:

- C/C++ functions
- Shell commands, after invoking the shell
- Callable services

At the first request, the system dubs the program as an OS/390 UNIX process, unless it is a POSIX(OFF) program (which is not dubbed).

With the OS/390 UNIX System Services application services, users can:

- Request services from the system through shell commands. Shell commands are like TSO/E commands.
- Write shell scripts to run tasks. Shell scripts are analogous to REXX EXECs.
- Run programs interactively (in the foreground) or in the background.

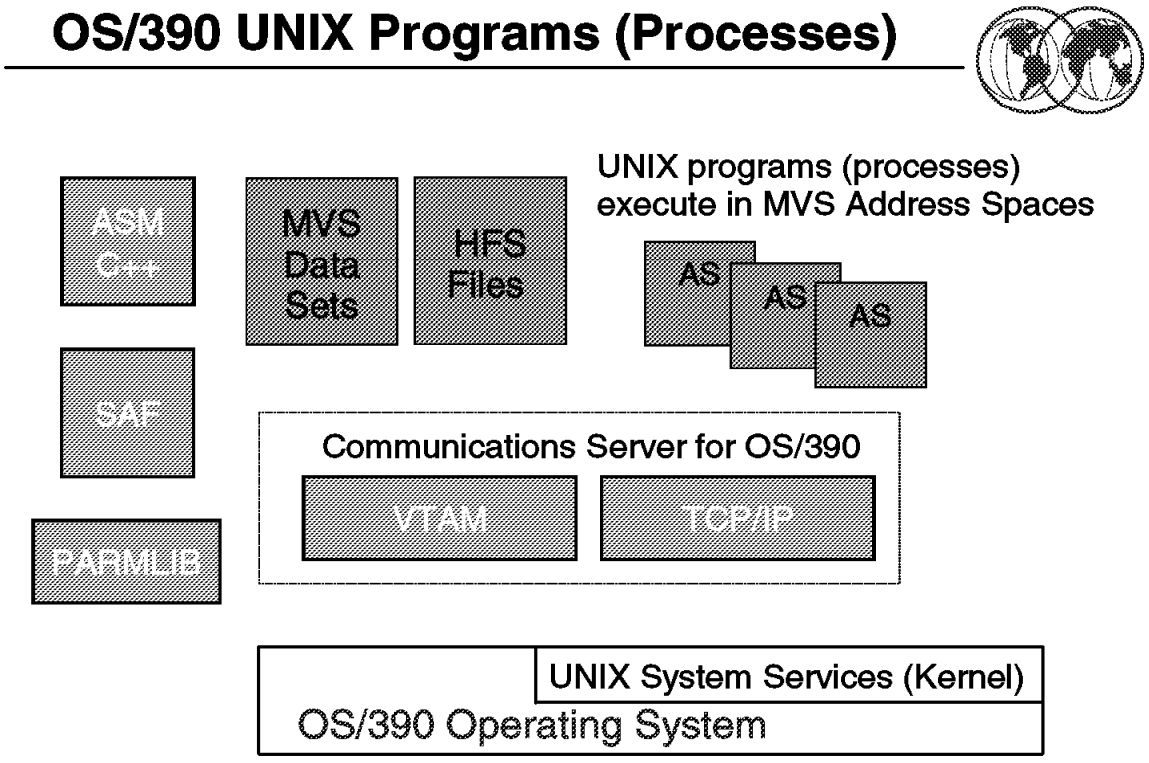


Figure 149. OS/390 UNIX programs (processes)

3.5.1 OS/390 UNIX programs (processes)

A process is a program using kernel services. The program can be created by the fork() function, fork callable service, or spawn() function; or the program can be dubbed because it requested kernel services. The three types of processes are:

- User processes, which are associated with a program or a shell user
- Daemon processes, which perform continuous or periodic system-wide functions, such as printer spooling
- Kernel processes, which perform system-wide functions for the kernel such as cleaning up zombie processes (init process)

When you enter a shell command, you start a process that runs in an MVS address space. When you enter that command, the OS/390 shell runs it in its own process group. As such, it is considered a separate job and the shell assigns it a job identifier—a small number known only to the shell. (A shell job identifier identifies a shell job, not an MVS job.) When the process completes, the system displays the shell prompt.

UNIX programs running in MVS address spaces use or access the following:

ASM/C++ These programming languages can be used to create the programs.

SAF A security product is required when running UNIX System Services. SAF is the interface to RACF, Top Secret, or ACF2.

BPXPRMxx This parmlib member determines the number of processes that may be started in the OS/390 system.

MVS data sets UNIX programs may access MVS data sets.

HFS files UNIX programs may access HFS files.

TCP/IP Workstation users may enter the shell environment by using `rlogin` or `telnet` in a TCP/IP network. User-written applications may use TCP/IP as a communication vehicle.

VTAM Workstation users may access TSO/E through VTAM. OS/390 UNIX is then accessed from TSO/E.

The system also assigns a process group identifier (PGID) and a process identifier (PID). When only one command is entered, the PGID is the same as the PID. The PGID can be thought of as a system-wide identifier. If you enter more than one command at a time using a pipe, several processes, each with its own PID, will be started. However, these processes all have the same PGID and shell job identifier. The PGID is the same as the PID for the first process in the pipe.

Process identifiers associated with a process are as follows:

PID A process ID (PID) is a unique identifier assigned to a process while it runs. When the process ends, its PID is returned to the system. Each time you run a process, it has a different PID (it takes a long time for a PID to be reused by the system). You can use the PID to track the status of a process with the `ps` command or the `jobs` command, or to end a process with the `kill` command.

PGID Each process in a process group shares a process group ID (PGID), which is the same as the PID of the first process in the process group. This ID is used for signaling related processes.

If a command starts just one process, its PID and PGID are the same.

PPID A process that creates a new process is called a parent process; the new process is called a child process. The parent process ID (PPID) becomes associated with the new child process when it is created. The PPID is not used for job control.

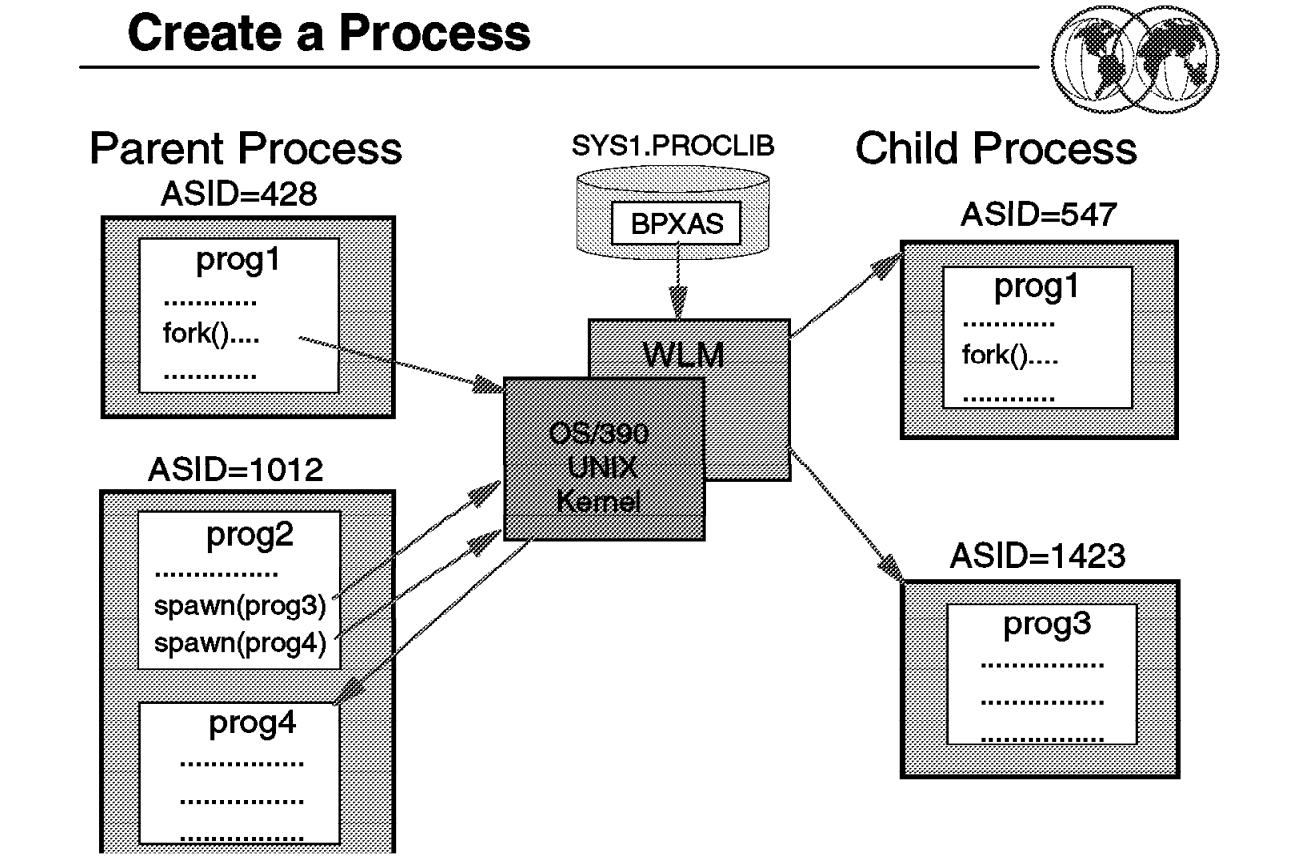


Figure 150. Create a process

3.5.2 Create a process

Fork() is a POSIX/XPG4 function which creates a duplicate process referred to as a child process. The process that issues the fork() is referred to as the parent process.

With OS/390 UNIX a program that issues a fork() function creates a new address space which is a copy of the address space where the program is running. The fork() function does a program call to the kernel which uses WLM/MVS facilities to create the child process. The contents of the parent address space are then copied to the child address space.

After the fork() function, the program in the child AS will start at the same instruction as the program in the parent AS. Control is returned to both programs at the same point. The only difference that the program sees is the return code from the fork() function. A return code of zero is returned to the child after a successful fork(). All other return codes are valid only for the parent.

A process will be created by using any of the following methods:

- C fork() function: Will create a child process which is identical to the parent process in a new address space scheduled by the Workload Manager (WLM).
- C spawn() function: Will create a child process which will be executing a different program than the parent, either in a new address space scheduled by WLM, or in the same address space as the parent process (local spawn).

- OS/390 UNIX callable services: When a program uses an OS/390 UNIX assembler callable service, the OS/390 address space will be *dubbed* an OS/390 UNIX process. The address space will get a PID. The dub will not result in a new address space.

The kernel interfaces to WLM to create the new address space for a fork or spawn. WLM uses an IBM supplied procedure *BPXAS* to start up a new address space. This new address space will then initialize the UNIX child process to run in the address space. After the child process completes, this address space can be reused for another fork or spawn. If none is waiting, BPXAS will time out after being idle for 30 minutes.

To be able to decide useful values for the BPXPRMxx keywords which affect process resources, it is necessary to understand how OS/390 UNIX processes are created and which functions can create a process.

When the program “prog1” in the parent address space (AS) issues the fork() function, the kernel is called to perform the action. The kernel calls the Workload Manager (WLM) to provide a new address space.

Once the child address space has been created, the child gets the required storage from a STORAGE request. The kernel then copies the contents of the parent AS to the child AS using the MVCL instruction. Once the copy has been completed, a short conversation between the kernel and the child process takes place. At this point, both the parent and child process are activated. The program in the child AS gets control at the same point as the program in the parent AS. The only difference is the return code from the fork() function.

The child address space is almost an identical copy of the parent address space. User data, for example private subpools, and system data, like RTM (Recovery Termination Management) control blocks, are identical.

Any linkage stack in the parent is not carried over to the child. Also, dataspaces and hiperspaces are not carried over since access list and access register contents are not copied. Internal timers together with SMF and SRM accounting data are set to zero in the child address space.

HFS files are allocated to the kernel and I/O is done by calling the kernel. On a fork(), all open files are inherited by the child address space. The child does not inherit any file locks for the HFS files.

MVS data sets are allocated to an individual address space, and after a fork() the child address space does not have access to the MVS data sets allocated by the parent.

The visual shows two of the methods of creating processes. The reason for reviewing how processes are created is that WLM is used for scheduling an AS for the functions that create a child process in a separate AS.

The OS/390 UNIX kernel will receive the requests for creating a new process, and depending on whether the process will be located in a new AS, the kernel will ask WLM to schedule an address space. If a new address space is not required the kernel will create a child process in the same AS as the parent.

The BPXAS address space is very similar to a JES2 initiator—have a look at the procedure in the next few pages. It actually enters the system as a started task. When it starts to run, a child process created by the kernel, it becomes a UNIX unit of work. When the child process completes, if there is no further work, the BPXAS reverts to an STC and waits on further fork/spawn work from WLM. If nothing appears for 30 minutes, the address space goes away. However, if you want to bring OS/390 down quickly, you need a way of quickly stopping all outstanding BPXAS address spaces.

3.5.2.1 fork() function

The fork() function comes from UNIX and the implementation in OS/390 is a very resource consuming function. The fork() function creates a new child process which is an identical copy of the parent process, and the child process will be executing the same program as the parent. Storage areas are copied from the parent to the child using MVCL. However, OS/390 UNIX provides the capability of using shared pages instead of copying the storage. In most cases where a fork() is used, the function exec() follows immediately to start a new program in the child process. Because of this, it seems quite unreasonable to copy all the storage and get rid of it without using it. Using shared pages is one solution and how to activate that will be covered in the customization unit.

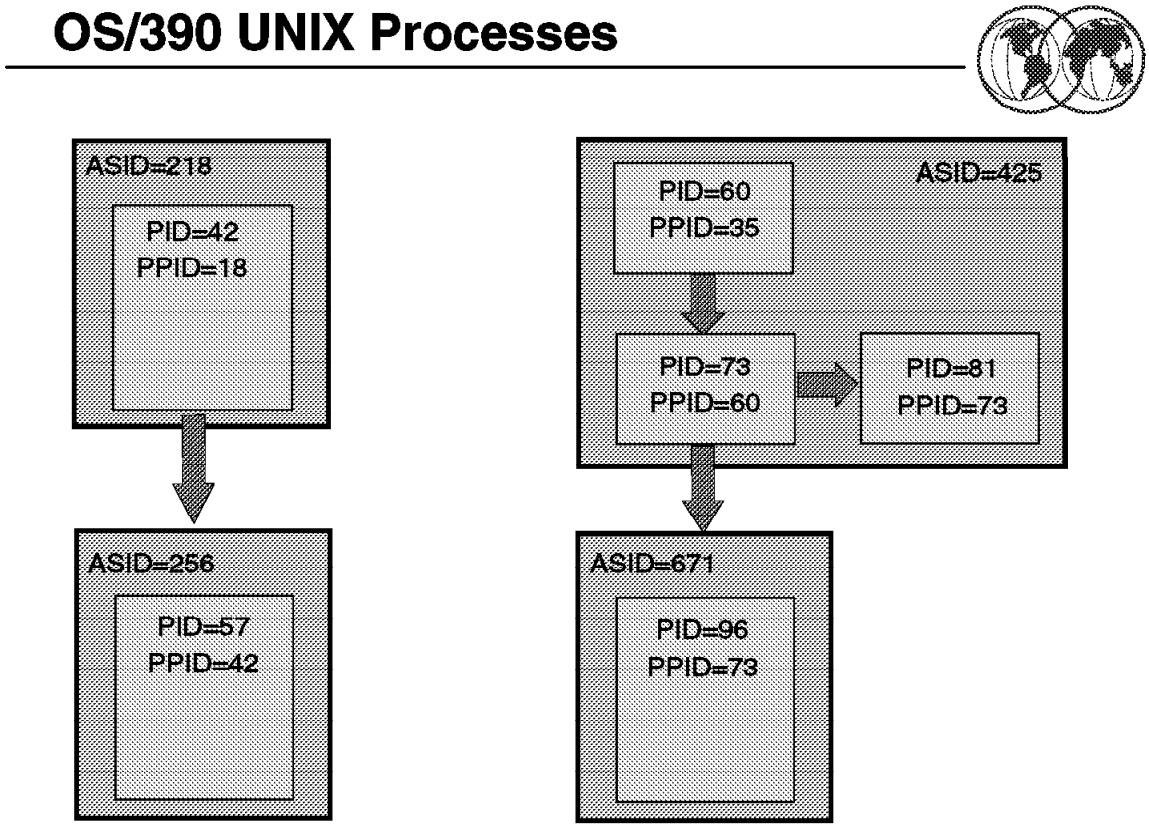


Figure 151. OS/390 UNIX processes

3.5.3 OS/390 UNIX processes

OS/390 UNIX uses processes to run programs, and to associate resources to the programs. An OS/390 address space can contain one or multiple processes.

A process is created by another process, or by a request for an OS/390 UNIX service. The process that creates a new process is called a parent process and the new process is called a child process. There will be a hierarchy of processes in the system.

Every process is identified by a process ID (PID) and is associated with its parent process by a parent process ID (PPID).

OS/390 UNIX supports processes that run in unique address spaces. These would be created by fork() and exec() services. It also supports local processes. These will share an address space and are created by the spawn() service.

The use of processes comes from UNIX where a process is used for executing some work. In the first OS/390 UNIX release, a process was the same as an address space. Since later releases of OS/390 UNIX provide the capability of having multiple processes within an address space, it is more suitable to compare a process to a task in OS/390. OS/390 UNIX was changed to allow multiple processes within an address space to save resources and improve performance.

The OS/390 UNIX shell is a large consumer of processes. The shell will create a new process for almost each shell command a user executes. The process will only exist as long as it takes for the command to execute. This concept comes from UNIX. With later releases of OS/390 UNIX more

commands are inline commands, which means they will not be executed in a separate process. The shell will also use the capability of having multiple processes within the user shell process so that the shell commands which will execute in a different process will still be in the same AS.

A process can create one or more child processes, and the child processes can be parent processes of new child processes, thus creating a hierarchy of related processes. The PPID maintains the relationship between processes. Usually, a process creates a child process to perform a separate task, for example a shell command. The child process ends when the task is completed while the parent process continues to execute. If for some reason a parent process should terminate before a child process, the child process will become an orphan process. Orphan processes are inherited by the first process created in the system called the “init” process.

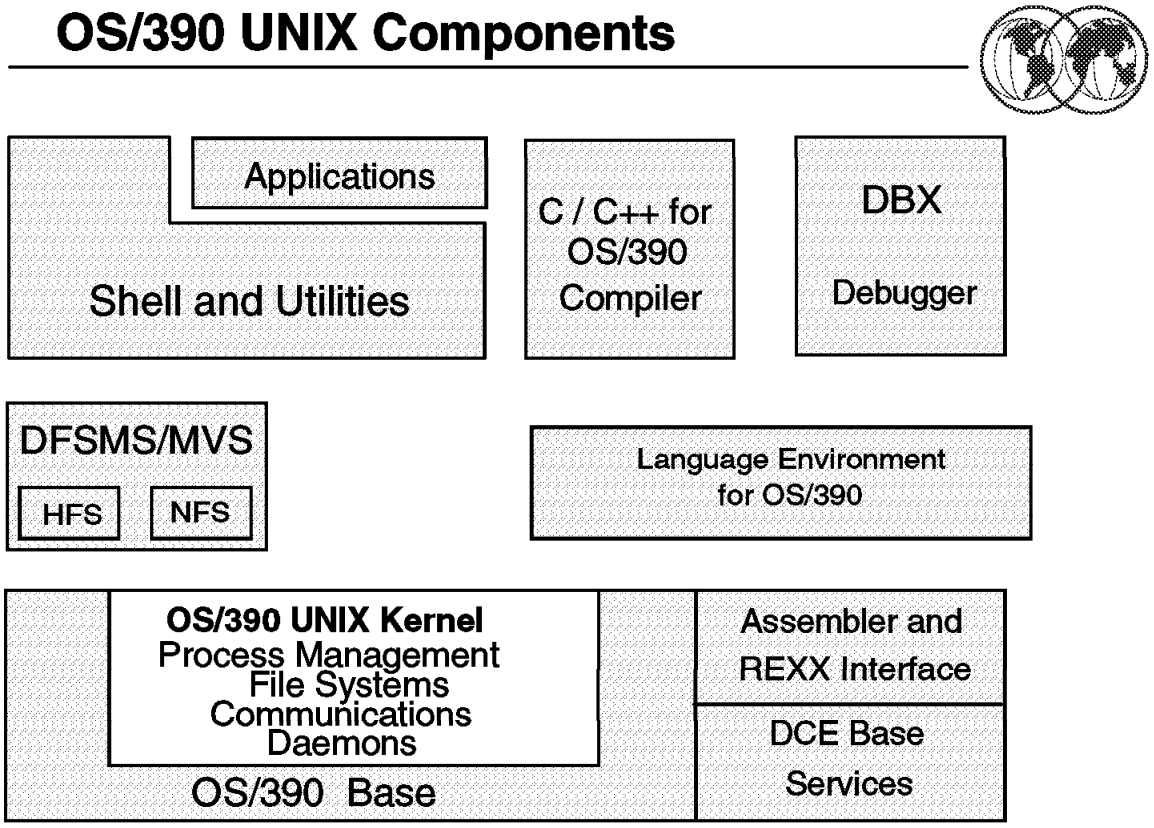


Figure 152. OS/390 UNIX components

3.5.4 OS/390 UNIX components

OS/390 UNIX offers open interfaces for applications and interactive users on an OS/390 system. The OS/390 UNIX components and their functions are:

- The OS/390 UNIX kernel

  At system IPL time, kernel services are started automatically. The kernel provides OS/390 UNIX System Services in answer to requests from programs and the shell. The kernel manages the file system, the communications, and the program processes.

  In an OS/390 UNIX system the file system is part of the kernel. In an OS/390 UNIX environment the file system is also considered part of the kernel because it is allocated to the kernel. The support for the file system is provided by the DFSMS/MVS product. The visual showing the file system as part of the kernel shows a logical view of the solution.

  OS/390 UNIX requires MVS/ESA SP 4.3 or higher. The hierarchical file system is shipped as part of DFSMS/MVS.

  The POSIX standard introduces a completely new terminology in the MVS environment. A typical UNIX operating system consists of a kernel which interfaces directly with the hardware. Built on the kernel is the shell and utilities layer that defines a command interface. Then there are application programs built on the shell and utilities.

  The OS/390 UNIX API conforms to the POSIX and XPG4 standard. OS/390 was branded as a UNIX system by the Open Group in 1996. To support the APIs, the OS/390 system must provide some

system services which are included in the kernel, such as the file system and communication services.

Daemons are programs that are typically started when the operating system is initialized and remain active to perform standard services. Some programs are considered daemons that initialize processes for users even though these daemons are not long-running processes. OS/390 UNIX supplies daemons that start applications and start a user shell session when requested.

- The OS/390 UNIX shell and utilities

  An interactive interface to OS/390 UNIX services which interprets commands from interactive users and programs.

  The shell and utilities component can be compared to the TSO function in OS/390.

- The OS/390 UNIX debugger (dbx)

  The dbx debugger is not part of the POSIX standard. It is based on the dbx debugger that is well known in many UNIX environments.

  The OS/390 UNIX debugger is a tool which application programmers can use to interactively debug a C program. The debugger is based on the dbx debugger which is common in many UNIX environments.

- The C/C++ compiler and C run-time libraries

  The C/C++ compiler and C run-time libraries are needed to make executables that the kernel understands and can manage.

- DFSMS/MVS

  DFSMS/MVS manages the hierarchical file system (HFS) data sets that contain the file systems. To use kernel services in full-function mode, SMS must be active.

  Network File System (NFS) enables users to mount file systems from other systems so that the files appear to be locally mounted. You end up with a mixture of file systems that come from systems where the UIDs and GIDs may be independently managed.

- Language Environment (LE)

  To run a shell command or utility, or any user-provided application program written in C or C++, you need the C/C++ run-time library provided with Language Environment.

  The C compiler and LE feature (Language Environment) library are changed and extended to include support for the POSIX and XPG4 C function calls. The LE product provides a common run-time environment and language-specific run-time services for compiled programs.

- DCE services

  DCE services are part of the OS/390 base and are exploiters of the POSIX and XPG services provided by OS/390 UNIX.

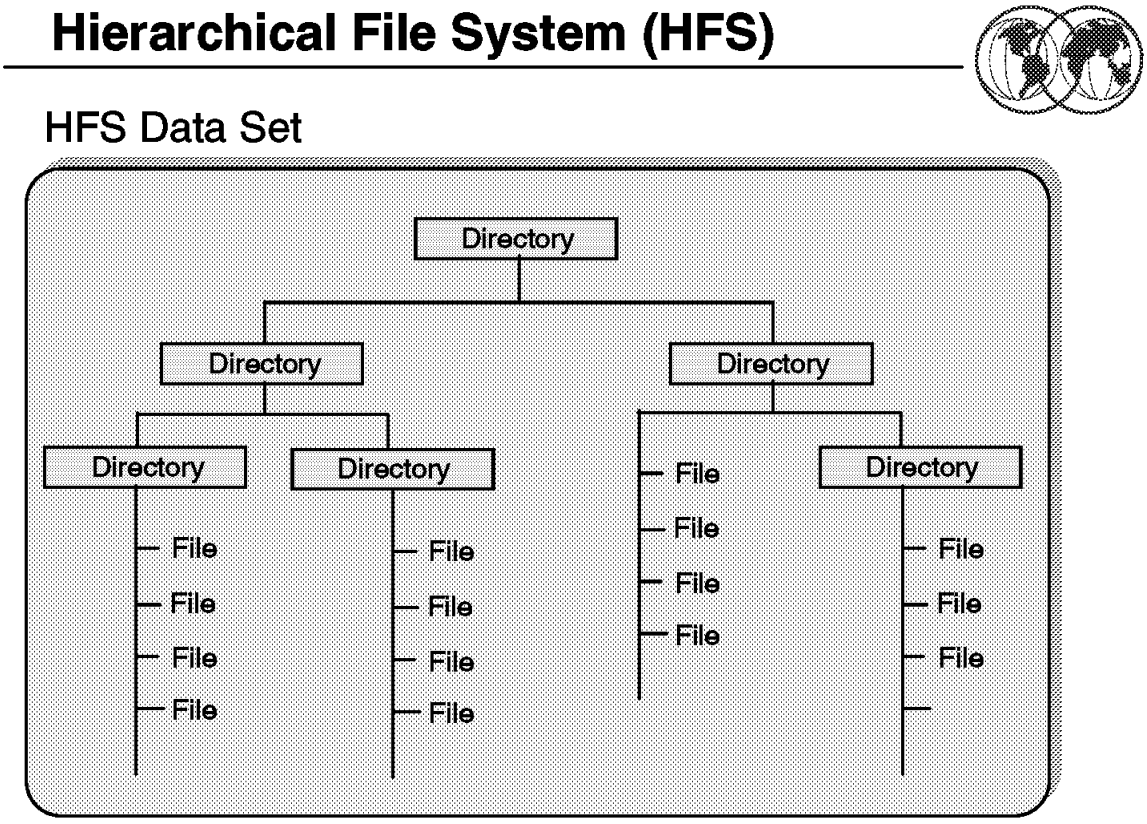


Figure 153. Hierarchical file system (HFS)

3.6 Hierarchical file system (HFS)

The OS/390 UNIX file system is hierarchical and byte-oriented. Finding a file in the file system is done by searching a directory or a series of directories. There is no concept of an OS/390 catalog that points directly to a file.

A path name identifies a file and consists of directory names and a file name. A fully qualified file name which consists of the name of each directory in the path to a file plus the file name itself, can be up to 1023 bytes long.

The hierarchical file system allows for file names in mixed case.

The hierarchical file system (HFS) data set which contains the hierarchical file system is an OS/390 (MVS) data set type.

The file system is very similar to hierarchical file systems on UNIX, DOS, and OS/2. The visual shows an example of a hierarchical file system structure.

The files in the hierarchical file system are sequential files, and they are accessed as byte streams. A record concept does not exist with these files other than the structure defined by an application.

The path name is constructed of individual directory names and a file name separated by the forward-slash character, for example, “/dir1/dir2/dir3/myfile.”

Like UNIX, OS/390 UNIX is case sensitive to file and directory names. For example, in the same directory the file “MYFILE” is a different file than “myfile.”

HFS data sets and OS/390 data sets can reside on the same SMS-managed DASD volume.

The integration of the HFS file system with existing MVS file system management services provides automated file system management capabilities which may not be available on other POSIX platforms. This allows file owners to spend less time on tasks such as back up and restore of entire file systems.

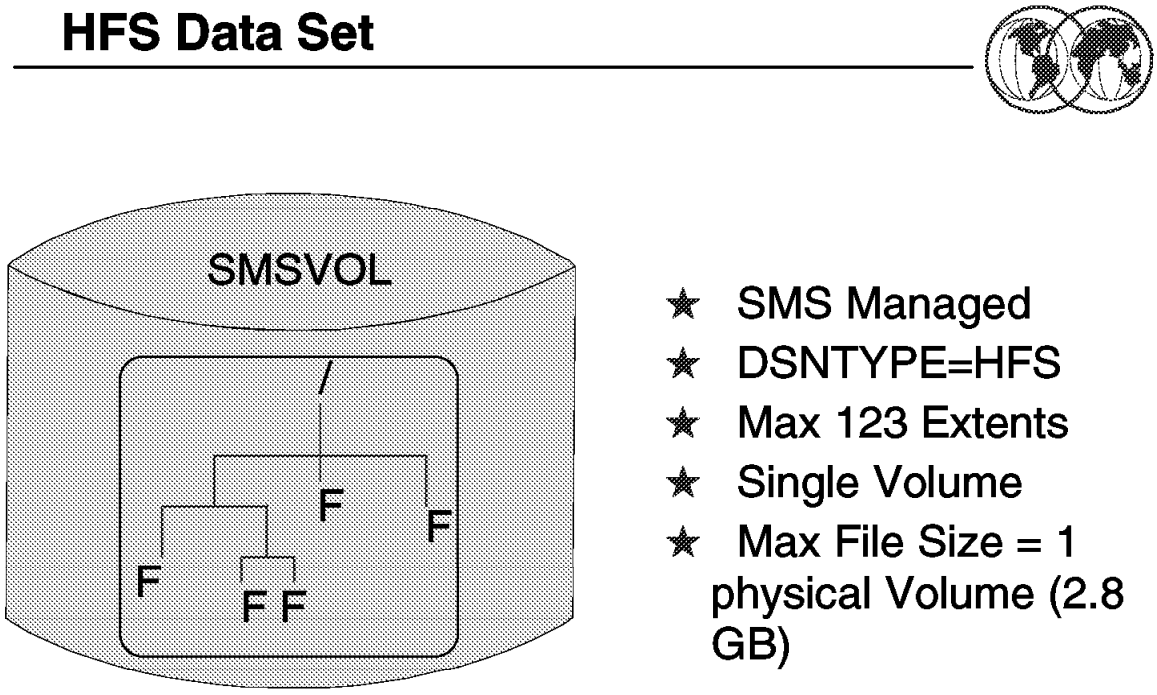


Figure 154. HFS data set

3.6.1 HFS data sets

An OS/390 UNIX hierarchical file system is contained in a data set type called HFS. An HFS data set must reside on an SMS-managed volume, and it is a single volume data set. HFS data sets can reside with other OS/390 data sets on SMS-managed volumes. Multiple systems can share an HFS data set if it is mounted in read-only mode.

An HFS data set can have up to 123 extents, and the maximum size of the data set is one physical volume. For a 3390-Model 3 the maximum size is 2.838 GB. HFS data sets can only be accessed by OS/390 UNIX.

An HFS data set is allocated by specifying HFS in the DSNTYPE parameter. You can also define a data class for HFS data sets. All HFS data sets must be system-managed, and an individual HFS data set can reside on only one DASD volume.

RACF or an equivalent security product must be installed and active on your system to use OS/390 UNIX or HFS data sets.

An HFS data set is an OS/390 data set, but it cannot be accessed by traditional OS/390 services like OPEN. Only OS/390 UNIX knows about the internal structures in the HFS data set which makes up a hierarchy of directory and files similar to the file system in OS/2, DOS, or UNIX. An HFS data set must be an SMS-managed data set.

The OS/390 UNIX file system is composed of multiple HFS data sets connected to each other in a hierarchy. On top of the hierarchy is the root file system. The root file system contains system

directories and files. Most installations will have user data in separate HFS data sets which are connected to the root. Connecting a file system to another is called “mounting.”

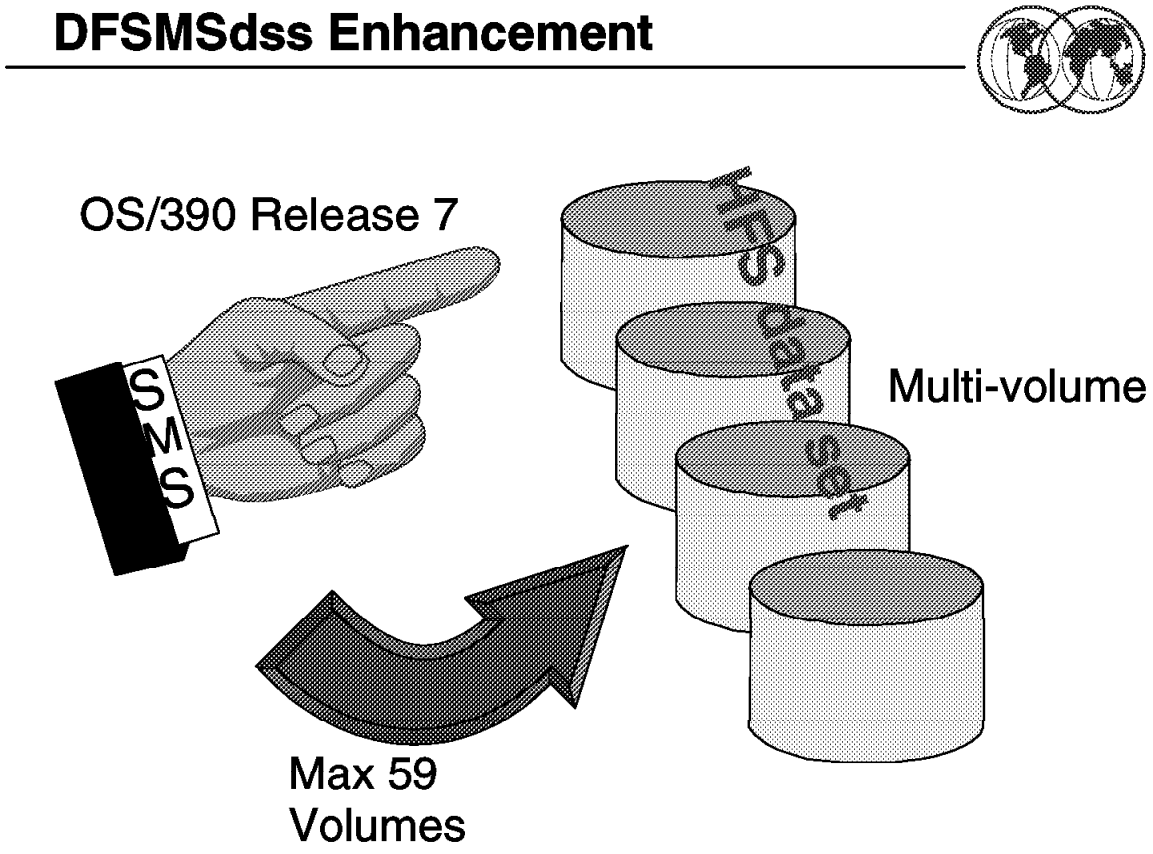


Figure 155. DFSMSdss enhancement

3.6.2 DFSMSdss enhancement for HFS data sets

Before OS/390 R7, the maximum size of the HFS was one physical volume. For the 3390-Model 3 DASD the maximum size was 2.8 GB.

OS/390 R7 now supports multivolume access up to 59 physical volumes. To make this clearer:

```
59 * 2.8 GB = 165.2 GB !!
```

This is only for one HFS data set. Imagine the possibilities if you are mounting HFS data in addition to the root HFS!

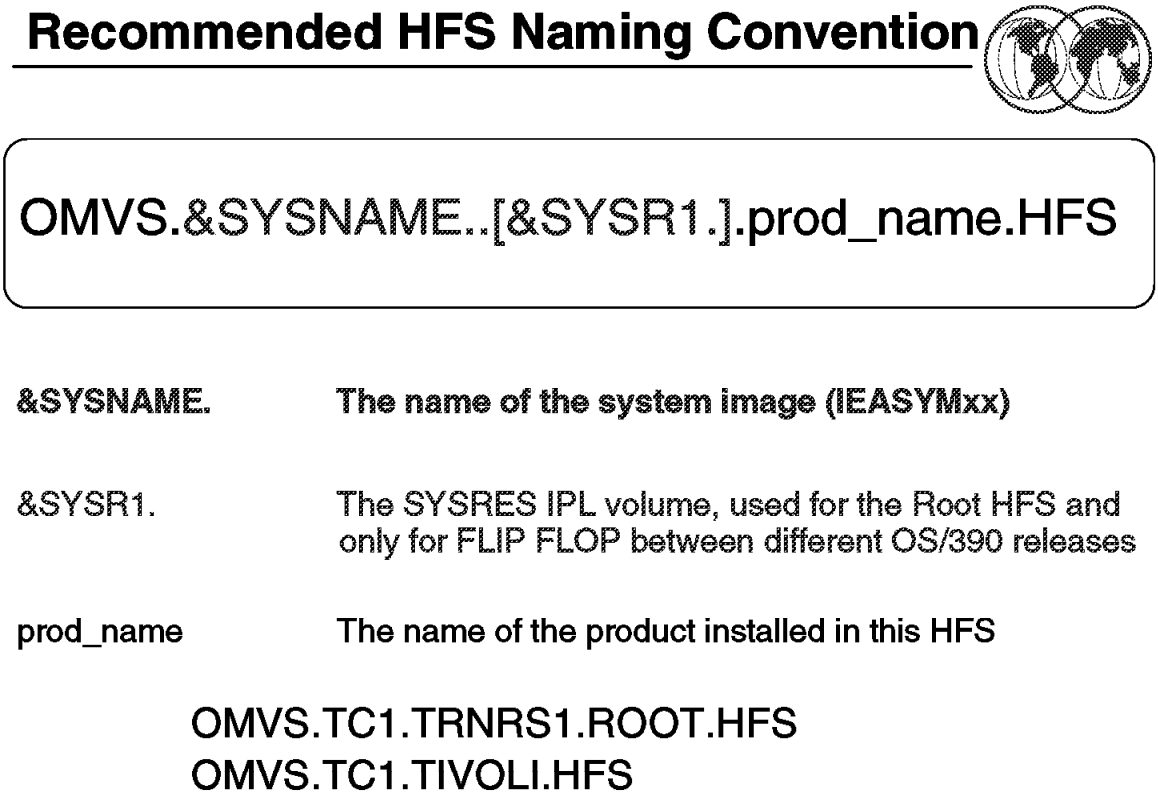


Figure 156. Naming convention for HFS

3.6.3 HFS naming convention

On this visual, you see a recommended naming convention for HFS data sets. This convention is based on practical experience, and takes into consideration a sysplex naming convention.

The first OMVS is used to assign a storage class through the SMS ACS routines. The <SYSNAME> is the name of the system where this HFS will be mounted for use. The <SYSR1> is the system residence volume used to identify the root HFS in a system, where you can have different releases of OS/390. If you want to switch the system without renaming the HFS data set, then use this qualifier. The prod_name describes the product name like root, Java, DCE, and so on. The last qualifier tells you that this file is an HFS, and not, for example, a source file for HFS.

Some examples:

- OMVS.TC1.TRNRS1.ROOT.HFS
- OMVS.TC1.TIVOLI.HFS
- OMVS.PROD.ETC.HFS
- OMVS.TEST.TMP.HFS

Note: The <SYSR1> name is dropped for HFS data sets that are not tied to the SYSRES volume.

You need to have the SYSRES volser as part of the HFS root data set because you need to keep track of the SMP/E configuration or environment. This HFS root data set is part of most MVS installation

builds for SMP/E target zones based on SYSRES name. Therefore, adding the SYSRES name to the HFS data set name will help you keep your SMP/E environment workable.

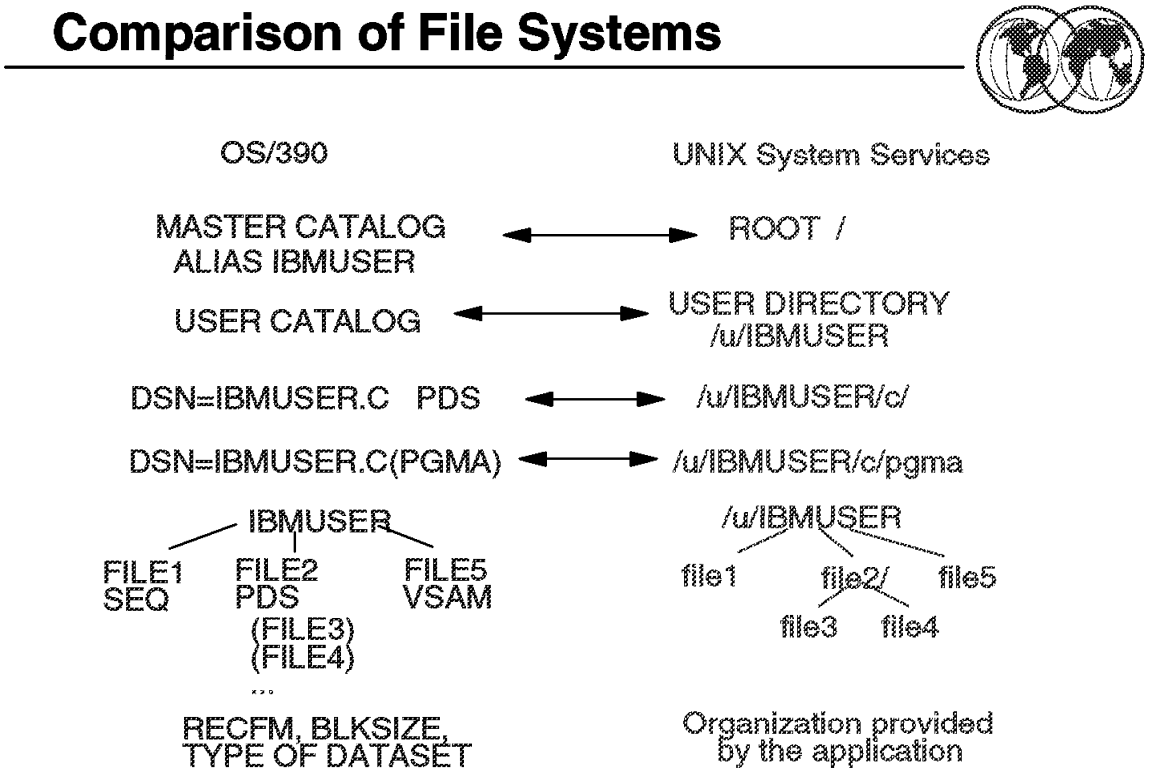


Figure 157. Comparison of file systems

3.6.4 Comparison of file systems

The OS/390 master catalog is analogous to the root directory in a hierarchical file system.

The user prefix assigned to OS/390 data sets points to a user catalog. The user catalog is organized analogous to a user directory (/u/ibmuser) in the file system. Typically, one user owns all the data sets whose names begin with his user prefix. For example, the data sets belonging to the TSO/E user ID ibmuser all begin with the prefix ibmuser. There could be data sets named IBMUSER.C, and IBMUSER.C(PGMA).

In the file system, ibmuser would have a user directory named /u/ibmuser. Under that directory there could be subdirectories named /u/ibmuser and /u/ibmuser/c/pgma.

Of the various types of OS/390 data sets, a partitioned data set (PDS) is most akin to a user directory in the file system. In a partitioned data set such as IBMUSER.C, you could have members PGMA, PGMB, and so on. For example, you could have IBMUSER.C(PGMA) and IBMUSER.C(PGMB). A subdirectory such as /u/ibmuser/c can hold many files, such as pgma, pgmb, and so on.

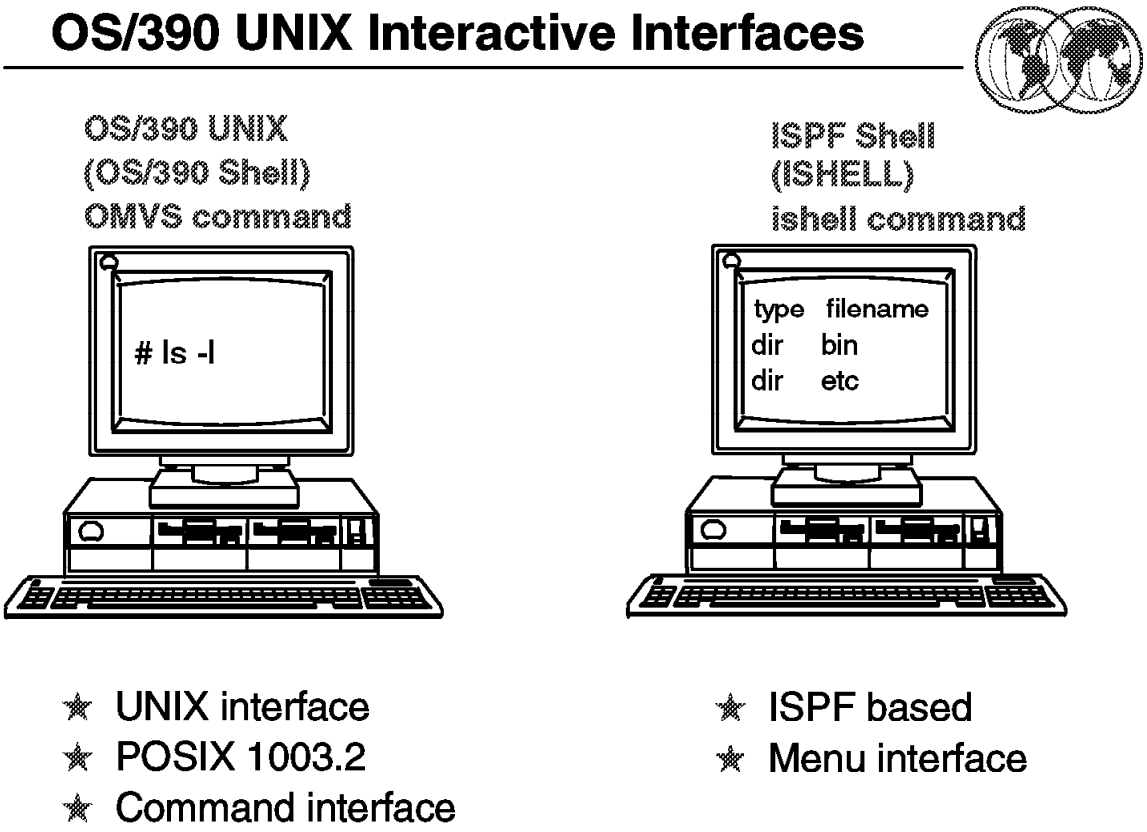


Figure 158. OS/390 UNIX interactive interfaces

3.7 OS/390 UNIX interactive interfaces

The visual shows an overview of the two interactive interfaces, OS/390 UNIX shell and ISHELL. In addition, there are some TSO/E commands to support OS/390 UNIX, but they are limited to certain functions such as copying files, and creating directories

3.7.1.1 OS/390 UNIX shell

The OS/390 UNIX shell is based on the UNIX System V shell and has some of the features from the UNIX Korn shell. The POSIX standard distinguishes between a command which is a directive to the shell to perform a specific task, and a utility which is the name of a program callable by name from the shell. To the user, there is no difference between a command and a utility.

Interactive users of OS/390 UNIX have a choice between using a UNIX-like interface (the shell), a TSO interface (TSO commands), and an ISPF interface (ISPF CUA dialog). With these choices, users can choose the interface which they are most familiar with and get a quicker start on OS/390 UNIX.

The OS/390 UNIX shell provides the environment that has the most functions and capabilities. Shell commands can easily be combined in pipes or shell scripts and thereby become powerful new functions. A sequence of shell commands can be stored in a text file which can be executed. This is called a shell script. The shell supports many of the features of a regular programming language.

3.7.1.2 The ISHELL

There are some TSO commands which provide support for UNIX System Services. An important command is the OMVS command which is the command to invoke the OS/390 UNIX shell. The ISHELL command will invoke the ISPF shell. The ISHELL is a good starting point for users familiar with TSO and ISPF that want/need to use OS/390 UNIX. The ISHELL provides CUA panels where users can work with the hierarchical file system. There are also panels for mounting/unmounting file systems and for doing some OS/390 UNIX administration.

The ISHELL is an ISPF dialog for users and system administrators which can be used instead of shell commands to perform many tasks related to file systems, files, and OS/390 UNIX user administration.

3.7.1.3 TSO REXX

The REXX support for OS/390 UNIX is not really an interactive interface, but I chose to introduce it here since it is most often used in TSO or in the shell. The SYSCALL environment is not built into TSO/E, but an external function call called SYSCALLS will initialize the environment. Note that the shell is the initial host environment, which means that the SYSCALL environment is automatically initialized. A difference between the REXX support and shell scripts is that a REXX EXEC can be invoked from a C program, while a shell script can only be interpreted from the shell. A REXX EXEC can be called from a shell script.

An interactive user who uses the OMVS command to access the shell can switch back and forth between the shell and TSO/E, the interactive interface to MVS.

Programmers whose primary interactive computing environment is a UNIX or AIX workstation find the OS/390 shell programming environment familiar.

Programmers whose primary interactive computing environment is TSO/E and ISPF can do much of their work in that environment.

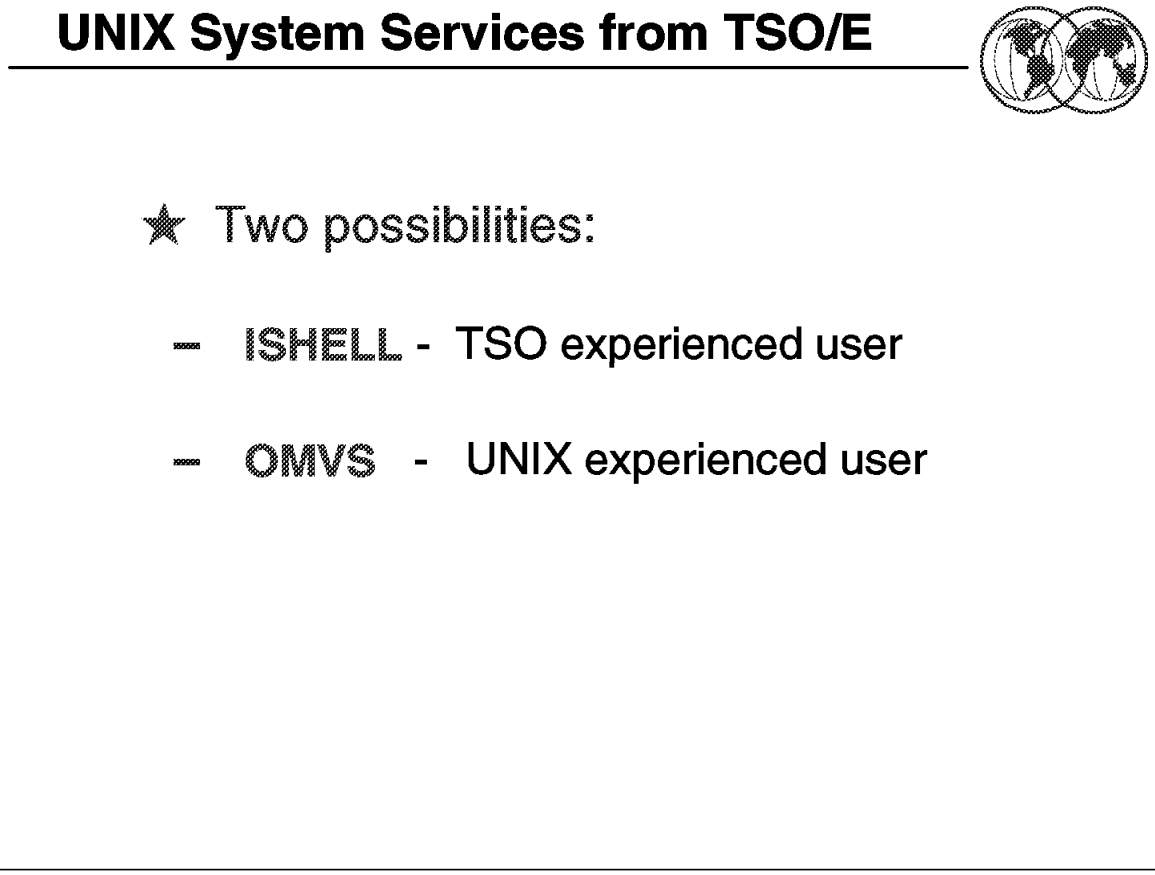


Figure 159. UNIX System Services from TSO/E

3.7.2 UNIX System Services from TSO/E

Interactive users of OS/390 UNIX have a choice between using a UNIX-like interface (the shell), a TSO interface (TSO commands), or an ISPF interface (ISPF CUA dialog). With these choices, users can choose the interface which they are most familiar with and get a quicker start on OS/390 UNIX.

There are some TSO commands which provide support for UNIX System Services, as follows:

ISHELL The ISHELL command will invoke the ISPF shell. The ISHELL is a good starting point for users familiar with TSO and ISPF that want/need to use OS/390 UNIX. The ISHELL provides CUA panels where users can work with the hierarchical file system. There are also panels for mounting/unmounting file systems and for doing some OS/390 UNIX administration.

OMVS An important command is the OMVS command which is the command to invoke the OS/390 UNIX shell.

The OS/390 UNIX shell provides the environment that has the most functions and capabilities. Shell commands can easily be combined in pipes or shell scripts and thereby become powerful new functions. A sequence of shell commands can be stored in a text file which can be executed. This is called a shell script. The shell supports many of the features of a regular programming language.

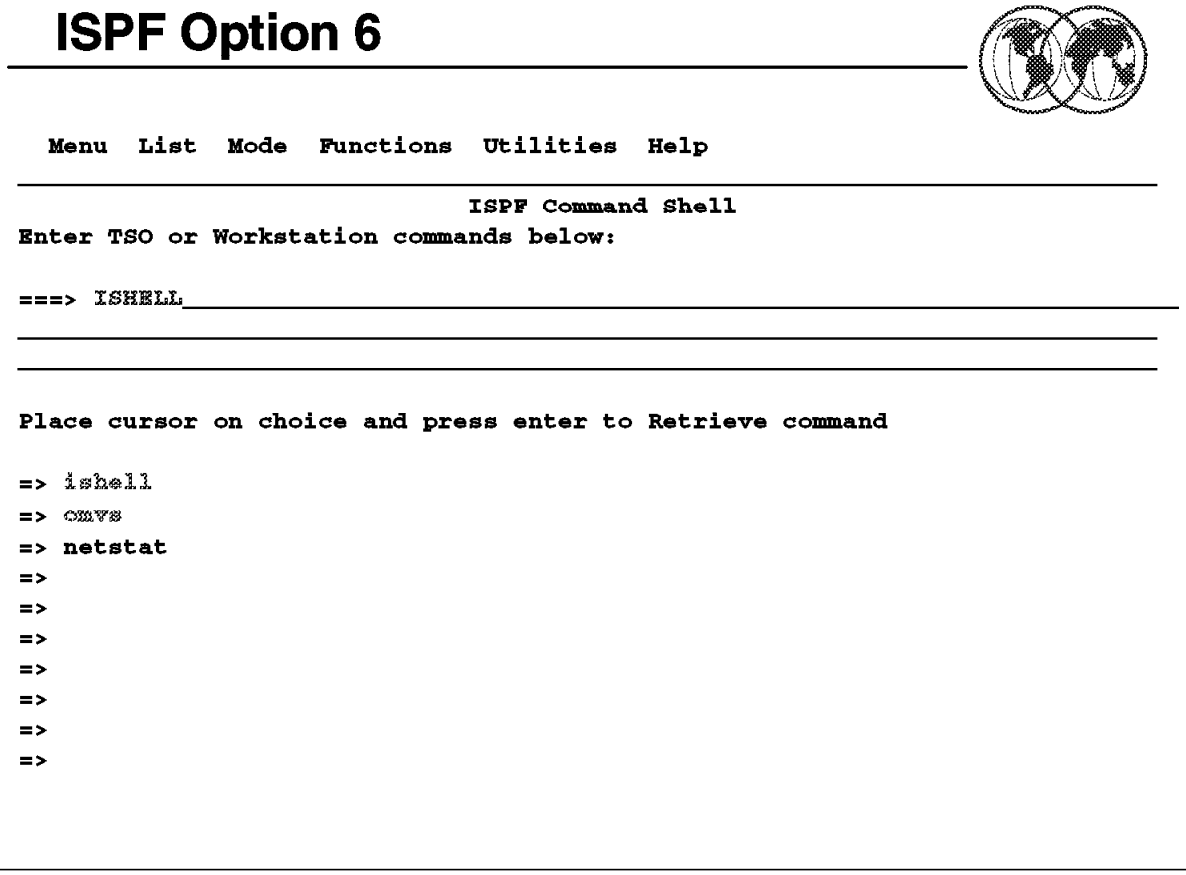


Figure 160. ISPF Option 6

3.7.3 ISPF Option 6

After a log on to TSO/E, enter Option 6 under ISPF to use the OMVS command and the ISHELL command.

If you are a user with an MVS background, you may prefer to use the ISPF shellpanel interface instead of shell commands or TSO/E commands to work with the file system. The ISPF shell also provides the administrator with a panel interface for setting up users for OS/390 UNIX access, for setting up the root file system, and for mounting and unmounting a file system.

You can also run shell commands, REXX programs, and C programs from the ISPF shell. The ISPF shell can direct stdout and stderr only to an HFS file, not to your terminal. If it has any contents, the file is displayed when the command or program completes.

ISHELL Command (ish)

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
-------------------------------------------------------------------------
                            OpenMVS ISPF Shell

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                                More:      +
   /u/paul_____________________________________________________________
   _____________________________________________________________________
   _____________________________________________________________________
   _____________________________________________________________________
```

Figure 161. ISHELL command panel

3.7.4 ISHELL command panel

This is the ISHELL or ISPF Shell panel as a result of the ISHELL command being entered from ISPF Option 6.

So to search a user's files and directories, type the following and press Enter:

```
/u/userID
```

At the top of the panel is the action bar, with seven choices:

```
File
Directory
Special file
Tools
File systems
Options
Setup
Help
```

When you select one of these choices, a pull-down panel with a list of actions is displayed.

My Files and Directories

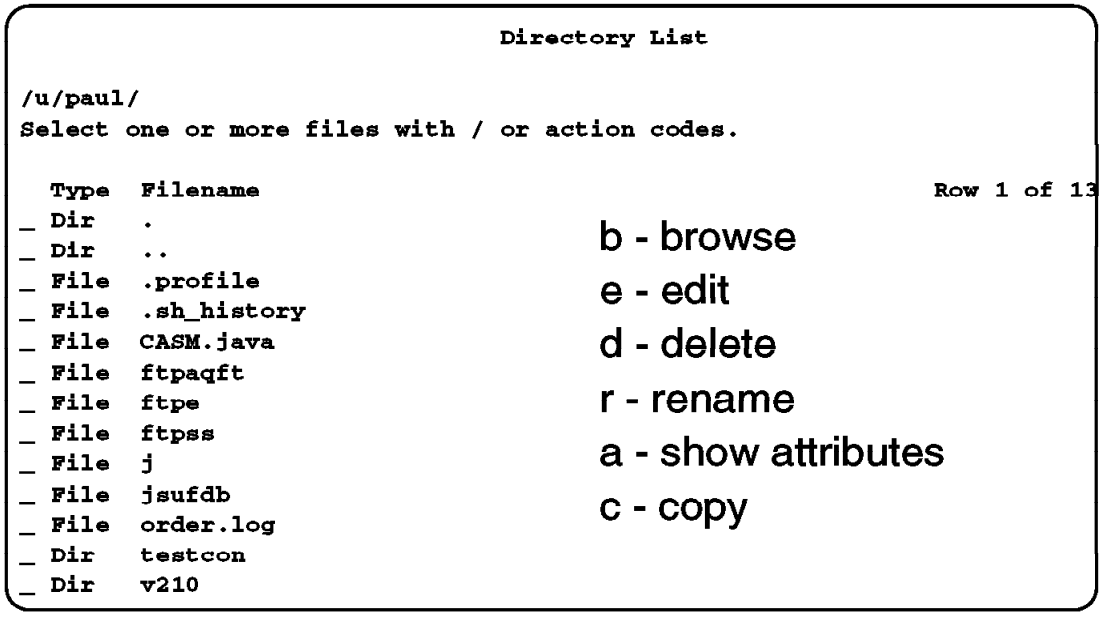


Figure 162. Files and directories

3.7.5 Files and directories

This visual shows the files and directories of user PAUL. The user can then use action codes to do the following:

b Browse a file or directory

e Edit a file or directory

d Delete a file or directory

r Rename a file or directory

a Show the attributes of a file or directory

c Copy a file or directory

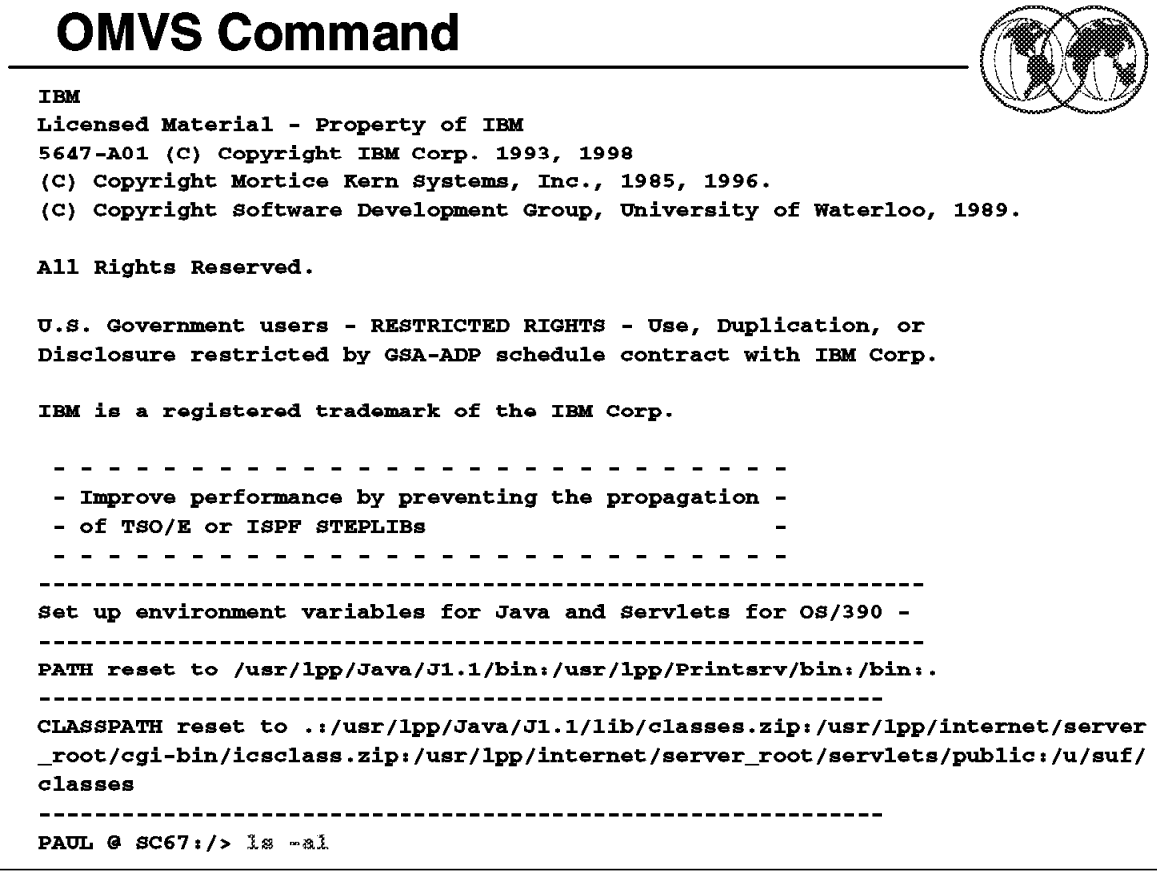


Figure 163. OMVS command

3.7.6 OMVS command

Use the OMVS command to invoke the OS/390 shell. After you are working in a shell session, you can switch to subcommand mode, return temporarily to TSO/E command mode, or end the session by exiting the shell.

The shell is a command processor that you use to:

- Invoke shell commands or utilities that request services from the system.
- Write shell scripts using the shell programming language.
- Run shell scripts and C-language programs interactively (in the foreground), in the background, or in batch.

The OS/390 shell provides commands and utilities that give the user an efficient way to request a range of services. A shell command is used to include both a command (a directive to a shell to perform a specific task) and a utility (the name of a program callable by name from a shell).

Shell commands often have options (also known as flags) that you can specify, and they usually take an argument—such as the name of a file or directory. The format for specifying the command begins with the command name, then the option or options, and finally the argument, if any. For example:

```
ls -a myfiles
```

Where:

`ls` is the command name, `-a` is the option, and `myfiles` is the argument.

This visual shows the screen when the `OMVS` command is issued from ISPF Option 6.

The command line is at the bottom of the screen and the following command is shown:

```
ls -al
```

This command lists the files and directories of the user. If the pathname is a file, `ls` displays information on the file according to the requested options. If it is a directory, `ls` displays information on the files and subdirectories therein. You can get information on a directory itself using the `-d` option.

If you do not specify any options, `ls` displays only the filenames. When `ls` sends output to a pipe or a file, it writes one name per line; when it sends output to the terminal, it uses the -C (multicolumn) format.

ls -al Command - List files in Root

```
PAUL @ SC67:/>ls -al
total 424
drwxr-xr-x  19 PAUL     DCEGRP         0 Sep 18 14:19 .
drwxr-xr-x  19 PAUL     DCEGRP         0 Sep 18 14:19 ..
dr-xr-xr-x   2 PAUL     OMVSGRP        0 Jan 27  1998 ...
-rw-------   1 PAUL     DCEGRP      2183 Sep 18 14:20 .sh_history
drw-------   2 PAUL     DCEGRP         0 May 31 17:31 ispf
----------   1 PAUL     DCEGRP         0 May 27 16:19 STDOUT
drwxr-xr-x   4 PAUL     OMVSGRP        0 Aug  2 17:53 bin
drwxr-xr-x   2 PAUL     OMVSGRP        0 Jun  7 15:32 dev
drwxr-xr-x  11 FWADM    FWADMIN        0 Jun  7 15:19 etc
drwxr-xr-x   2 PAUL     DCEGRP         0 May  7 10:57 junk
lrwxrwxrwx   1 PAUL     DCEGRP        16 Mar 23 14:31 krb5 -> etc/dce/var/krb5
drwxr-xr-x   2 PAUL     OMVSGRP        0 Jan 27  1998 lib
-rw-r--r--   1 PAUL     DCEGRP      1673 Jun  2 09:02 myenv
drwxr-xr-x   2 PAUL     DCEGRP         0 Aug  6 13:02 newjava
drwxr-xr-x   3 PAUL     OMVSGRP        0 Jan 27  1998 opt
drwxr-xr-x   3 PAUL     OMVSGRP        0 Apr 13 13:48 samples
drwxr-xr-x   2 PAUL     DCEGRP         0 Jul 31 10:29 service_r5
drwxr-xr-x  13 FWADM    FWADMIN        0 Apr  8 10:58 service_r6
drwxrwxrwx   2 PAUL     OMVSGRP     4000 Sep 18 14:19 tmp
drwxr-xr-x   8 PAUL     SYS1           0 Jul 30 16:00 u
drwxr-xr-x   3 PAUL     DCEGRP         0 Jun  5 17:59 user
drwxr-xr-x  12 PAUL     OMVSGRP        0 Jan 27  1998 usr
drwx--x--x   3 PAUL     SYS1           0 Apr 17 14:12 var
PAUL @ SC67:/>
 ===>
```

Figure 164. Files in a user's root directory

3.7.7 OMVS command results

This visual shows the result of the `ls -al` command. It displays the files and directories and shows the file access allowed for the user, the user's group, and other users.

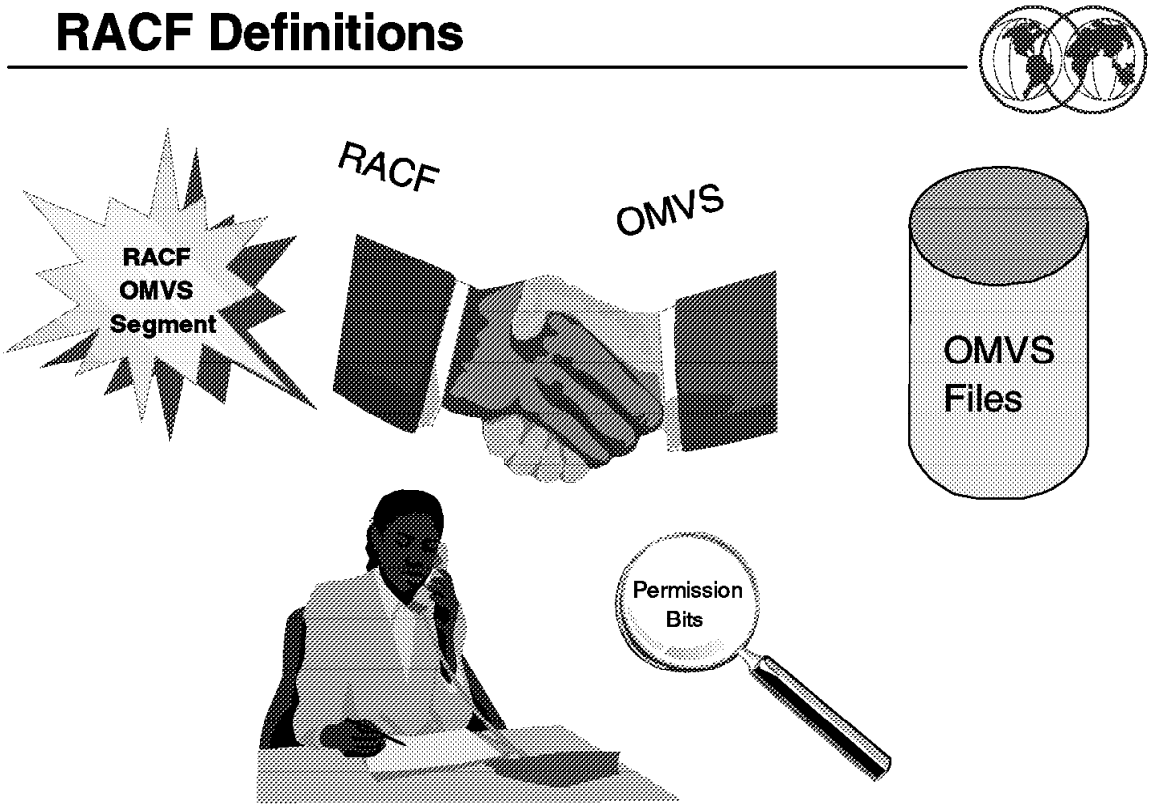


Figure 165. RACF definitions

3.8 RACF definitions

OS/390 UNIX provides security mechanisms that work with the security offered by the OS/390 system. A security product is required.

Before you can install and debug UNIX System Services you need to have access to UNIX System Services data sets and members which are named in directories and files.

RACF and also other security related products allow access to our MVS UNIX environment. This access can be allowed without being a TSO/E user.

If your user ID should have access to MVS UNIX your security administrator has to specify the home directory, the shell program, and a UID in the OMVS segment. In addition the security administrator has to provide a group ID (GID) for any RACF group that the security administrator is connected to. If this is done you should be able to access the UNIX shell or work with the ISPF driven ISHELL. The decision whether you will be able to access directories or files will be made by UNIX security.

In addition, your RACF administrator has to provide a user ID that is assigned to the OMVS and BPXOINIT address spaces (started procedure). This user ID needs only to have an OMVS segment and should be connected to a RACF group with a GID.

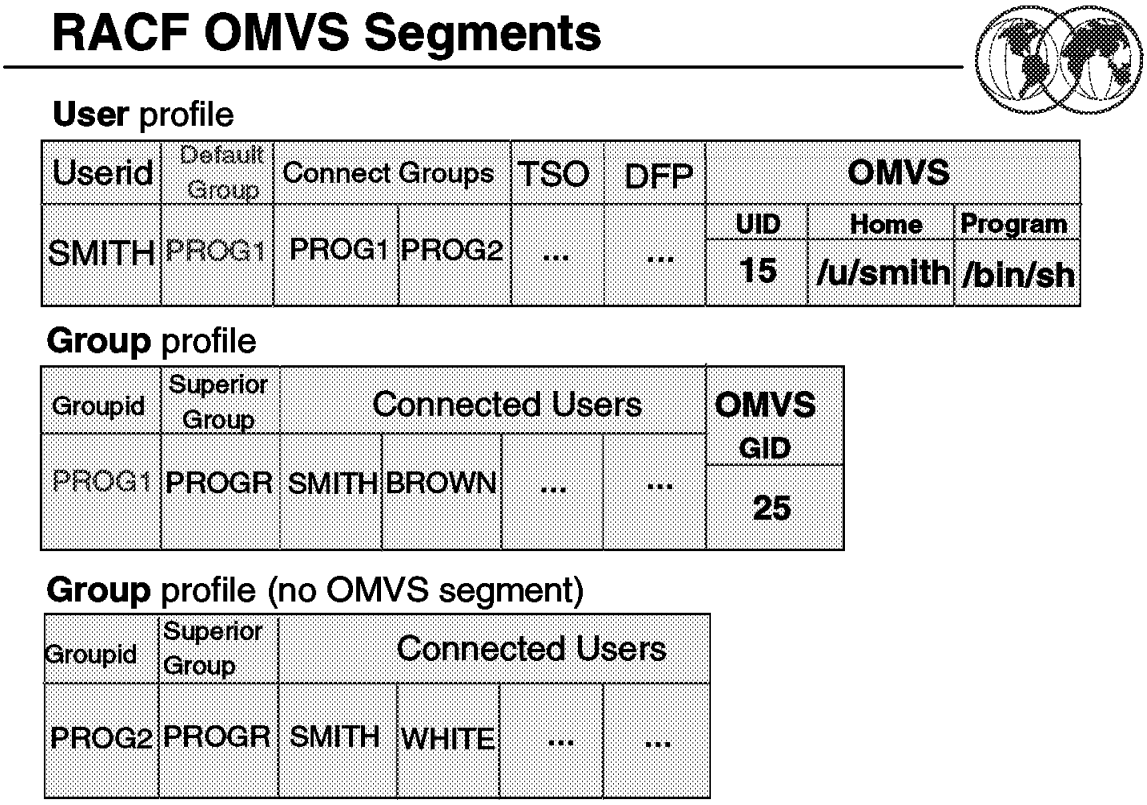


Figure 166. RACF OMVS segments

3.8.1 RACF OMVS segments

All users and programs that need access to OS/390 UNIX System Services must have a RACF user profile defined with an OMVS segment which has as a minimum a UID specified. A user without a UID cannot access OS/390 UNIX. Note that it is possible for multiple users to have the same UID number specified. However, this is not recommended.

The RACF user profile has a segment called OMVS for OS/390 UNIX support. A user ID must have an OMVS segment defined in order to use OS/390 UNIX System Services, for example access the ISHELL or the shell. This segment has three fields:

UID A number from 0 to 2147483647 that identifies an OS/390 UNIX user. An OS/390 UNIX user must have a UID defined.

HOME The name of a directory in the file system. This directory is called the home directory and becomes the current directory when the user accesses OS/390 UNIX. This field is optional.

PROGRAM The name of a program. This is the program that will be started for the user when the user begins an OS/390 UNIX session. Usually this is the program name for the OS/390 UNIX shell. This field is optional.

The RACF group also has a segment called OMVS to define OS/390 UNIX groups. It contains only one field:

GID A number from 0 to 2147483647 which identifies an OS/390 UNIX group.

The home directory is the current directory when a user invokes OS/390 UNIX. During OS/390 UNIX processing this can be changed temporarily by using the cd (change directory) shell command. The command will not change the value in the RACF profile. The directory specified as home directory in the RACF profile must exist (be pre-allocated) before a user can invoke OS/390 UNIX. If a home directory is not specified in RACF, the root (/) directory will be used as default.

The example on the visual shows a user profile for TSO/E user ID SMITH which is connected to two groups, PROG1 and PROG2. SMITH is defined as an OS/390 UNIX user because the user ID has a UID specified. The home directory is /u/smith and user SMITH will get into the shell after issuing the OMVS command because the name of the shell, /bin/sh is specified as program name.

The program name in the OMVS segment specifies the name of the first program to start when OS/390 UNIX is invoked. Usually this is the name of the OS/390 UNIX shell.

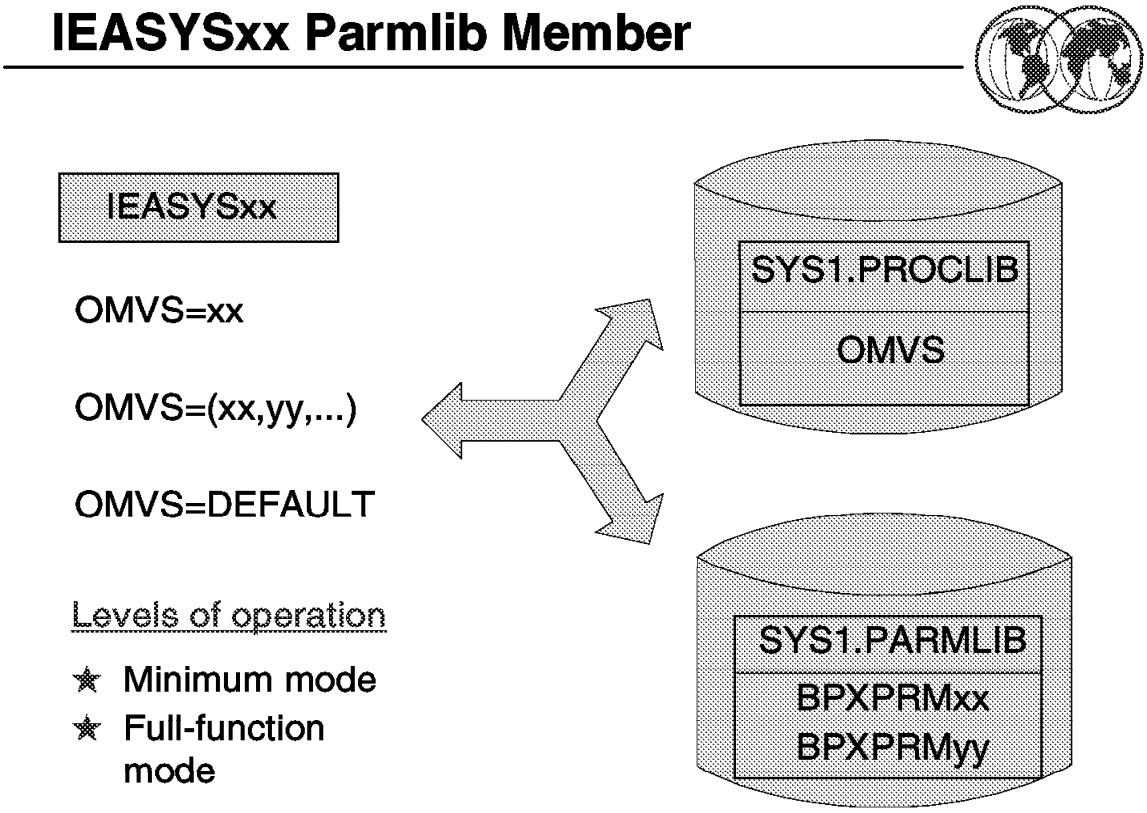


Figure 167. IEASYSxx parmlib member

3.9 IEASYSxx parmlib member

The initial parmlib settings for the OS/390 UNIX kernel are pointed to by the OMVS parameter in the IEASYSxx parmlib member.

The OMVS parameter in the IEASYSxx parmlib member lets you specify one or more BPXPRMxx parmlib members to be used to specify the initial parmlib settings for the kernel. If you do not specify the OMVS parameter, or if you specify OMVS=DEFAULT, the kernel is started in a minimum configuration mode with all BPXPRMxx parmlib statements taking their default values.

OMVS may also be left out or coded as DEFAULT. This allows the OS/390 UNIX kernel to start in a minimum configuration. All BPXPRMxx values will take their default values and a temporary root file system will be set up in memory.

Note: The start and stop commands for the OS/390 UNIX kernel are no longer supported.

Activation of kernel services is available in two modes:

- Minimum mode
- Full-function mode

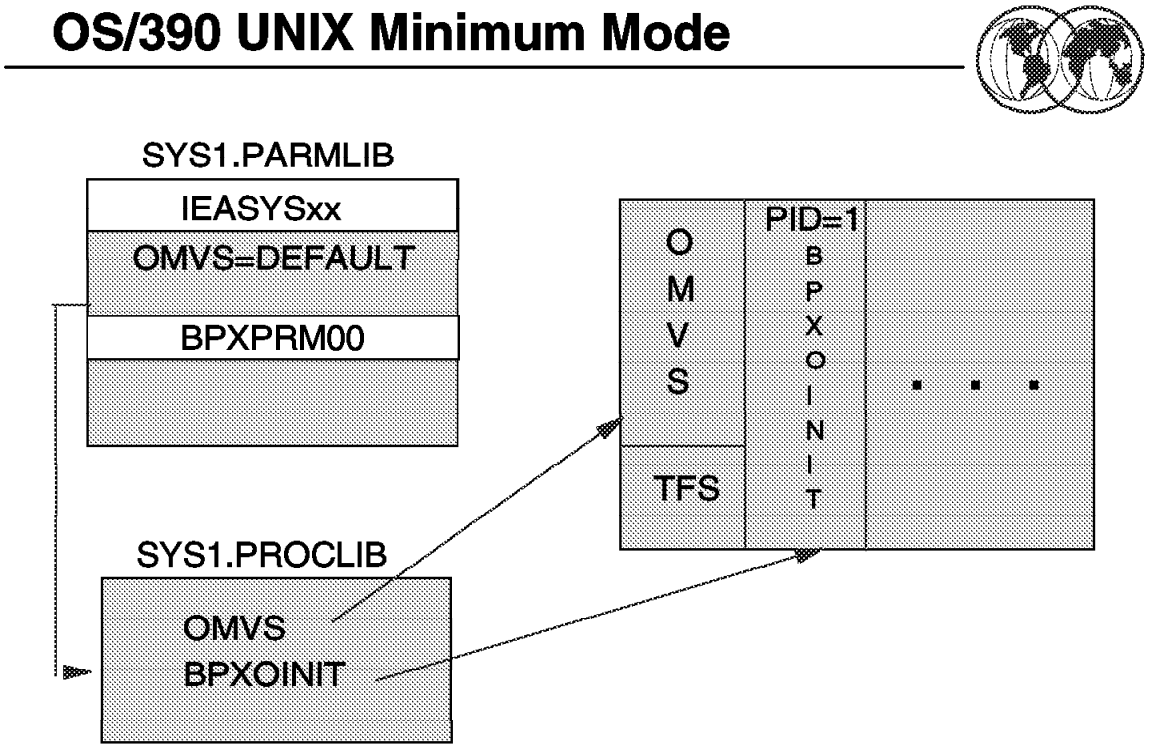


Figure 168. OS/390 UNIX minimum mode

3.9.1 OS/390 UNIX minimum mode

Minimum mode is intended for installations that do not intend to use OS/390 UNIX System Services, but which allows the IPL process to complete. In this mode many services are available to programs. Some that require further customization such as a fork() will fail.

If you do not specify OMVS= in the IEASYSxx parmlib member or if you specify OMVS=DEFAULT, then kernel services start up in minimum mode when the system is IPLed. This mode is intended for installations that do not plan to use the kernel services. In minimum mode:

- Many services are available, but some functions such as TCP/IP sockets that require other system customization, may not work.
- TCP/IP sockets (AF_INET) are not available.
- A *Temporary File System* (TFS) is used. A TFS is an in-storage file system, hence no physical DASD data set needs to exist or be mounted.

A temporary file system (kept in memory) is created for the root. The required directories (/bin, /etc, /tmp, /dev, and /dev/null) are built. There are no executables in this file system.

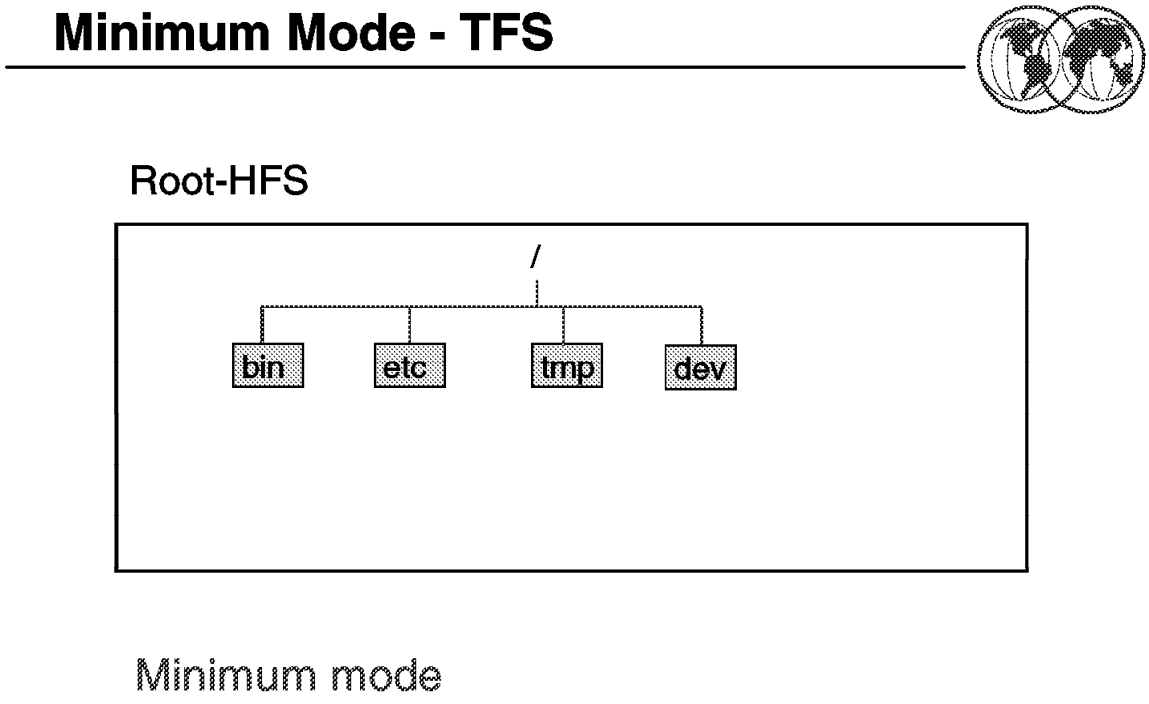


Figure 169. Minimum mode TFS

3.9.2 Minimum mode TFS

A temporary file system (TFS) is an in-memory file system which delivers high-speed I/O. If the kernel is started in minimum setup mode, the TFS is automatically mounted.

The system is in minimum mode when:

- OMVS=Default
- No OMVS Statement in IEASYSxx (no BPXPRMxx member in the parmlib)

A temporary file system is used as the root file system. The temporary file system is initialized and primed with a minimum set of files and directories.

Note: Any data written to this file system is not written to DASD.

The temporary file system is initialized with these directories and files:

```
/ (root directory)
/bin directory
/etc directory
/tmp directory
/dev directory
/dev/null file
```

There are no executables in the temporary file system (that is, you will not find the shell and utilities). Do not attempt to install OS/390 UNIX System Services application services in the TFS, since no data will be written to DASD.

Because the TFS is a temporary file system, unmounting it causes all data stored in the file system to be discarded. If, after an unmount, you mount another TFS, that file system has only a dot (.) and a dot-dot (..) and nothing else.

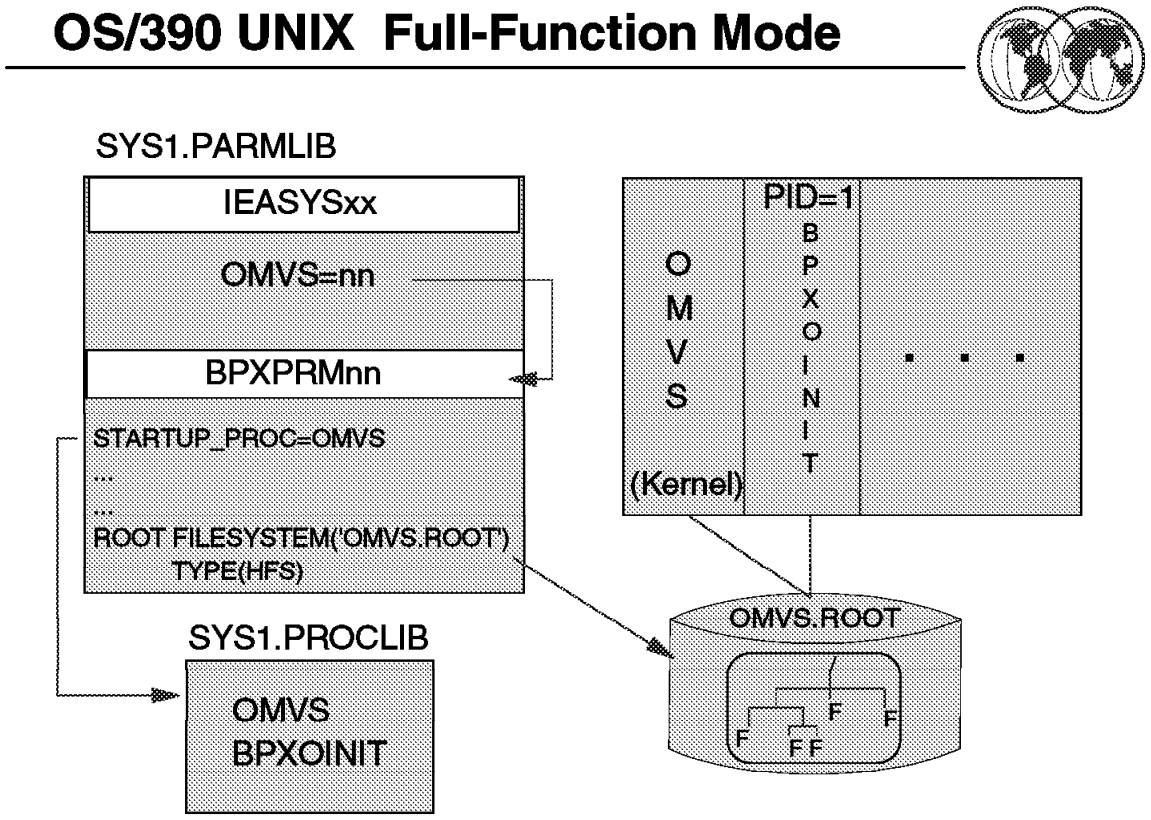


Figure 170. OS/390 UNIX full-function mode

3.9.3 OS/390 UNIX full-function mode

Full-function mode is started at IPL time when the OMVS parameter in the IEASYSxx parmlib member points to one or more BPXPRMxx parmlib members.

There must be a root HFS built and defined in BPXPRMxx for OMVS to initialize correctly and SMS, WLM, and RACF customization should be completed.

3.9.3.1 BPXPRMnn

BPXPRMnn is a 1-to-8 character name of a started procedure JCL that initializes the OS/390 UNIX kernel. The default is OMVS.

The ROOT statement defines and mounts the root file system. In this example:

- HFS is the TYPE of the file system.
- 'OMVS.ROOT' is the file system which is the name of an already defined PDSE/X data set.
- The root file system has a default of read/write mode.

If an active BPXPRMxx parmlib member specifies "FILESYSTEM TYPE(HFS)," then the kernel services start up in full-function mode when the system is IPLed. To use the full function, you need to:

- Set up BPXPRMxx parmlib definitions
- Set up DFSMS/MVS
- Set up the hierarchical file system/s (HFS)

- Install UNIX System Services application services
- Set up the security product definitions for OS/390 UNIX
- Set up the users′ access and their individual file systems

It is also possible to create a sockets-only level of OS/390 UNIX. This requires a BPXPRMxx parmlib member with only the socket file system defined. A minimal mode configuration with a temporary file system will be set up.

DFSMS/MVS manages the hierarchical file system (HFS) data sets that contain the file systems. To use kernel services in full-function mode, SMS must be active.

3.9.3.2 BPXOINIT

As of OS/390 Release 3, BPXOINIT is the started procedure that runs the initialization process. BPXOINIT is also the jobname of the initialization process. (Prior to OS/390 Release 3, the initialization process was created via an APPC allocate and the jobname was OMVSINIT.) BPXOINIT is shipped in SYS1.PROCLIB.

At system IPL time, kernel services are started automatically. If the OMVS parameter in the IEASYSxx parmlib parameter is not specified, the kernel services are started in minimum mode. If the OMVS parameter specifies one or more BPXPRMxx parmlib members, they are all used to configure the kernel services when the system is IPLed.

The BPXOINIT address space has two categories of functions:

1. It behaves as PID(1) of a typical UNIX system. This is the parent of /etc/rc, and it inherits orphaned children so that their processes get cleaned up using normal code in the kernel. This task is also the parent of any MVS address space that is dubbed and not created by fork or spawn. Therefore, TSO/E commands, batch jobs, and so on have a parent PID of 1.
2. Certain functions that the kernel performs need to be able to make normal kernel calls. This address space is used for these activities; for example, mmap() and user ID alias processing.

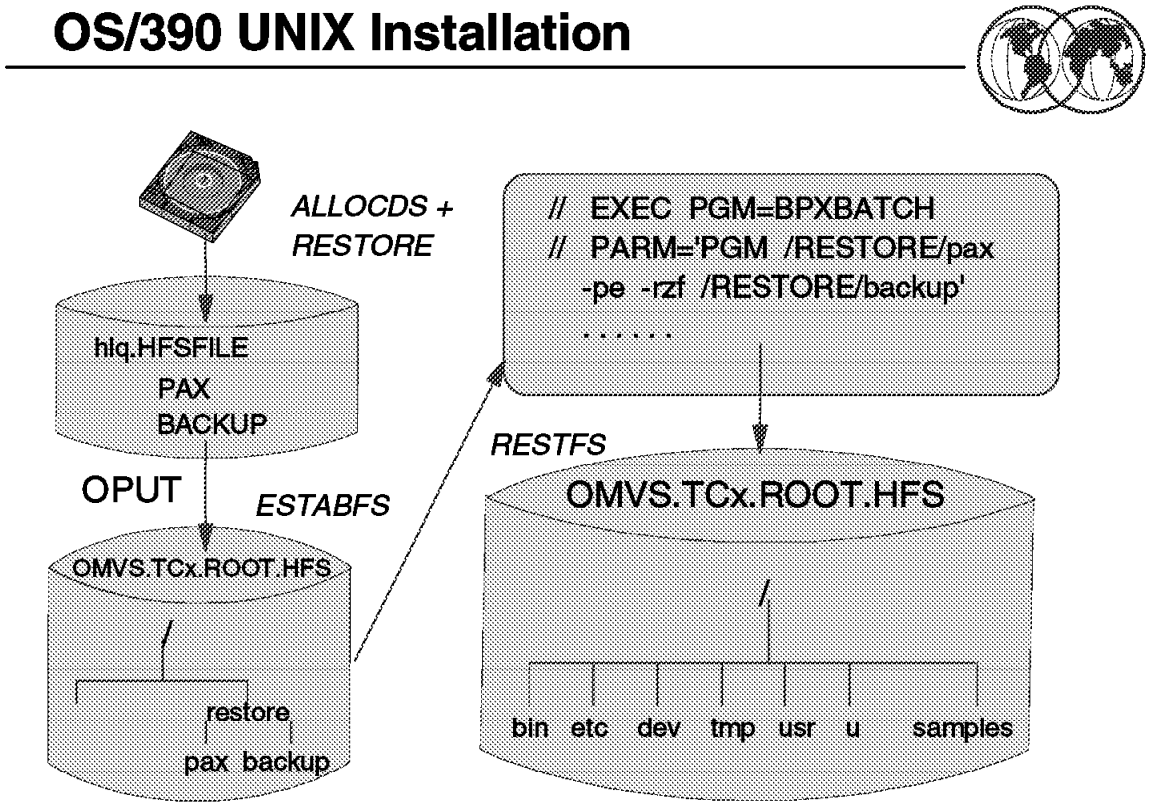


Figure 171. OS/390 UNIX installation

3.10 OS/390 UNIX installation

For a full configuration you need to build a complete root file system.

Beginning with OS/390 Version 2 Release 6, ServerPac supplies the jobs, ALLOCDS and RESTORE, to download the PDS from tape to DASD. The other two jobs, ESTABFS and RESTFS, will copy the two members of the “hlq.HFSFILE” to the HFS directory “/RESTORE” and decompress them.

After running these jobs you have a complete root file system containing all the root-level directories, files, and programs.

ServerPac builds the root for you with all the features that you ordered already installed in it. They use the UNIX pax utility to compress the hierarchical format into an HFS file. It is distributed to you as a member of ′hlq.HFSFILE′. Another member of this data set is the pax utility used to build the previous member.

pax pax reads and writes archive files. An archive file concatenates the contents of files and directories, and can also record file modification information such as dates, owner names, and so on. You can therefore use a single archive file to transfer a directory structure from one machine to another, or to back up or restore groups of files and directories.

```
hlq.HFSFILE(BACKUP)  Backup of Root File System
hlq.HFSFILE(PAX)     Decompress Utility
```

In OS/390 V2 R6 the ServerPac job ESTABFS performs the following tasks for you:

- Unmounts the IPLed ROOT temporary file system
- Creates ROOT HFS
- Mounts the created ROOT HFS to ′/′
- Creates ETC HFS
- Creates directory ′/etc′
- Mounts the created ETC HFS to ′/etc′
- Creates transient HFS
- Creates directory ′/RESTORE′
- Mounts the created transient HFS to ′/RESTORE′
- Updates the BPXPRMFS with the MOUNT FILESYSTYPE section
- Updates the BPXPRMFS with the FILESYSTYPE TYPE(....) and the corresponding entry points

After this job ends successfully, the next job to run is RESTFS. In OS/390 V2 R6 this job will do the following tasks for you:

- Re-mount the transient HFS (ends with RC=12 if already mounted)
- OPUT hlq.HFSFILE(BACKUP) to ′/RESTORE/backup′ in binary mode
- OPUT hlq.HFSFILE(PAX) to ′/RESTORE/pax′ in binary mode
- Allocates STDIN, STDOUT, and STDERR files (UNIX standard outputs). Receives an RC=4 if the directory is empty
- Unpack the ′/RESTORE/backup′ file with the ′/RESTORE/pax′ utility
  corresponding entry points
- Unlink the allocated file in the directory ′/RESTORE′
- Unmount transient HFS allocated to the directory ′/RESTORE′
- Remove the directory ′/RESTORE′
- Deletes the transient HFS DS

Chapter 4. Language Environment

Language Environment provides a common run-time environment for IBM versions of certain high-level languages (HLLs), namely, C, C++, COBOL, Fortran, and PL/I, in which you can run existing applications written in previous versions of these languages as well as in the current Language Environment-conforming versions. Prior to Language Environment, each of the HLLs had to provide a separate run-time environment.

Language Environment combines essential and commonly used run-time services, such as routines for run-time message handling, condition handling, storage management, date and time services, and math functions, and makes them available through a set of interfaces that are consistent across programming languages. With Language Environment, you can use one run-time environment for your applications, regardless of the application's programming language or system resource needs because most system dependencies have been removed.

Language Environment provides compatible support for existing HLL applications; most existing single-language applications can run under Language Environment without being recompiled or relink-edited. POSIX-conforming C applications can use all Language Environment services.

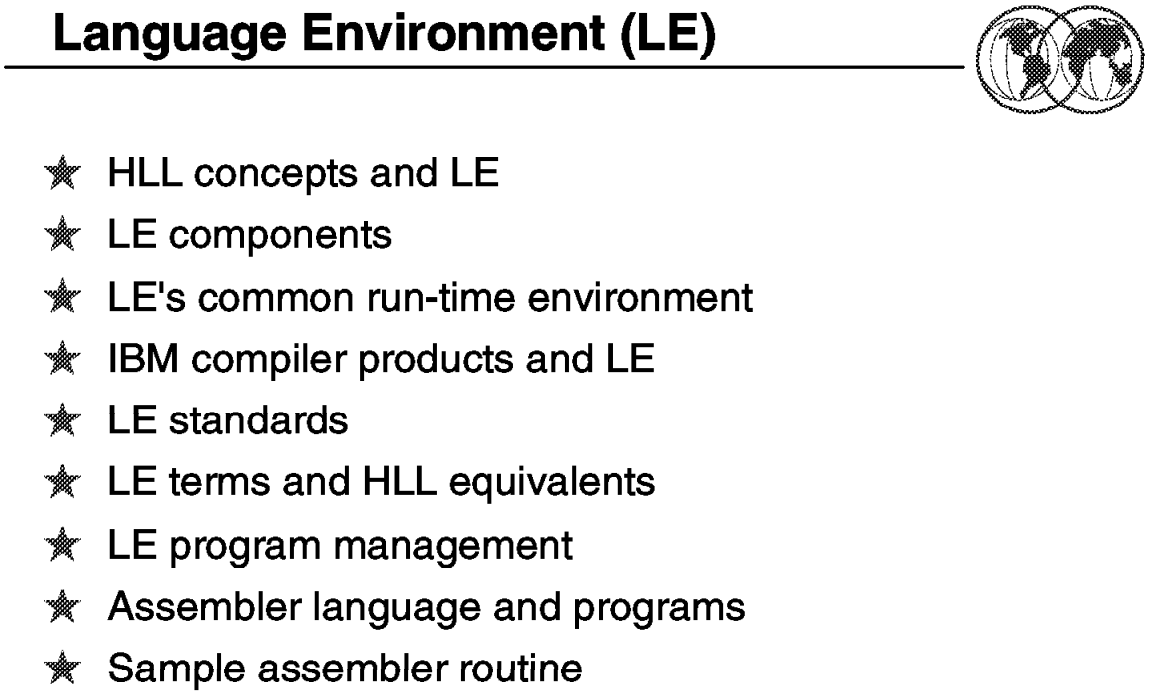


Figure 172. Language Environment (LE)

4.1 Language Environment (LE)

Today, enterprises need efficient, consistent, and less complex ways to develop quality applications and to maintain their existing inventory of applications. The trend in application development is to modularize and share code. Language Environment gives you a common environment for all Language Environment-conforming high-level language (HLL) products. An HLL is a programming language above the level of assembler language and below that of program generators and query languages.

In the past, programming languages also have had limited ability to call each other and behave consistently across different operating systems. This has constrained those who wanted to use several languages in an application. Programming languages have had different rules for implementing data structures and condition handling, and for interfacing with system services and library routines.

The visual describes the topics to be discussed in this chapter.

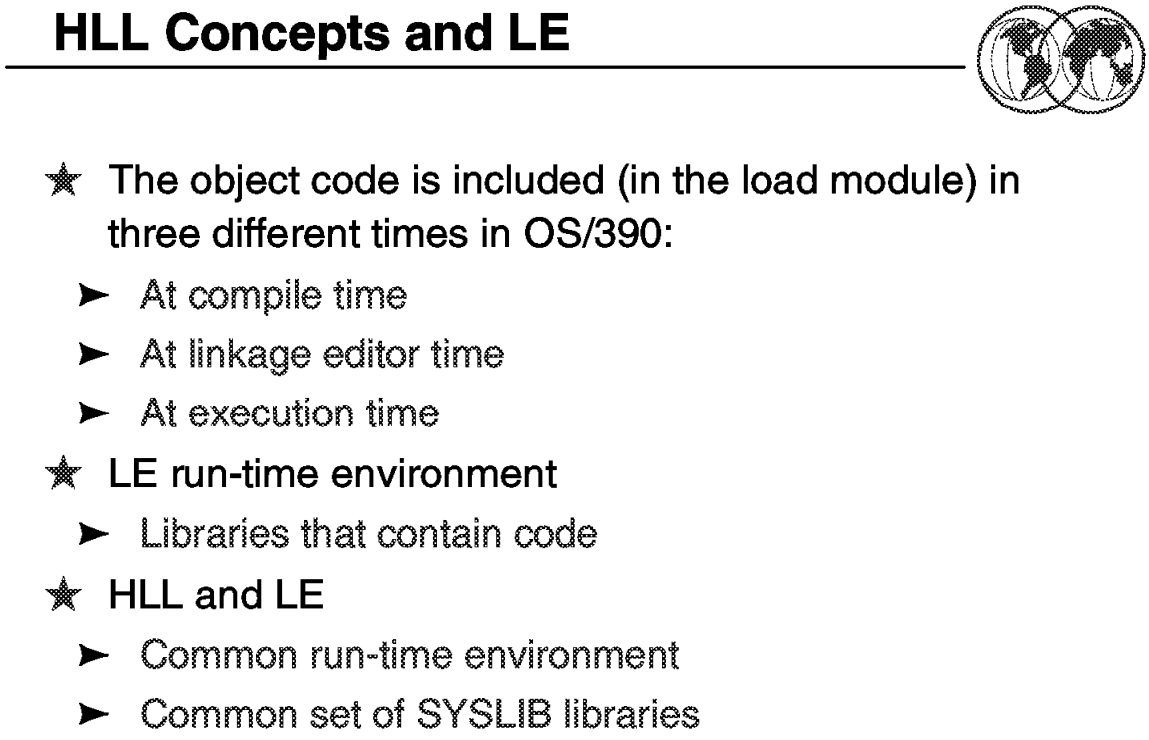


Figure 173. HLL concepts and LE

4.1.1 HLL concepts and LE

An HLL is implemented through a compiler code that translates the HLL statements in object code. This object code is the fabric of the executable program, also called the “load module” or “program object.” It has instructions recognized by the processor of the specific platform where it is supposed to be executed. The object code is included (in the load module) at three different times in OS/390:

- At compile time
- At linkage editor time, through the INCLUDE statement or resolving external references from the SYSLIB library
- At execution time, through MVS dynamic link macros such as LINK and LOAD, which fetch to memory, code from the load modules library. This approach is also called “late binding.”

The libraries containing the code invoked dynamically are called the run-time environment.

Language Environment establishes a common run-time environment plus a common set of the SYSLIB libraries for all participating HLLs. However, due to many language-specific functions, there is still a need of some language specific libraries as C/C++, COBOL, FORTRAN, PL/I. Language Environment combines essential run-time services, such as routines for run-time message handling, condition handling, and storage management. All of these services are available through a set of interfaces that are consistent across programming languages. You may either call these interfaces yourself, or use language-specific services that call the interfaces. With Language Environment, you can use one run-time environment for your applications, regardless of the application programming language or system resource needs.

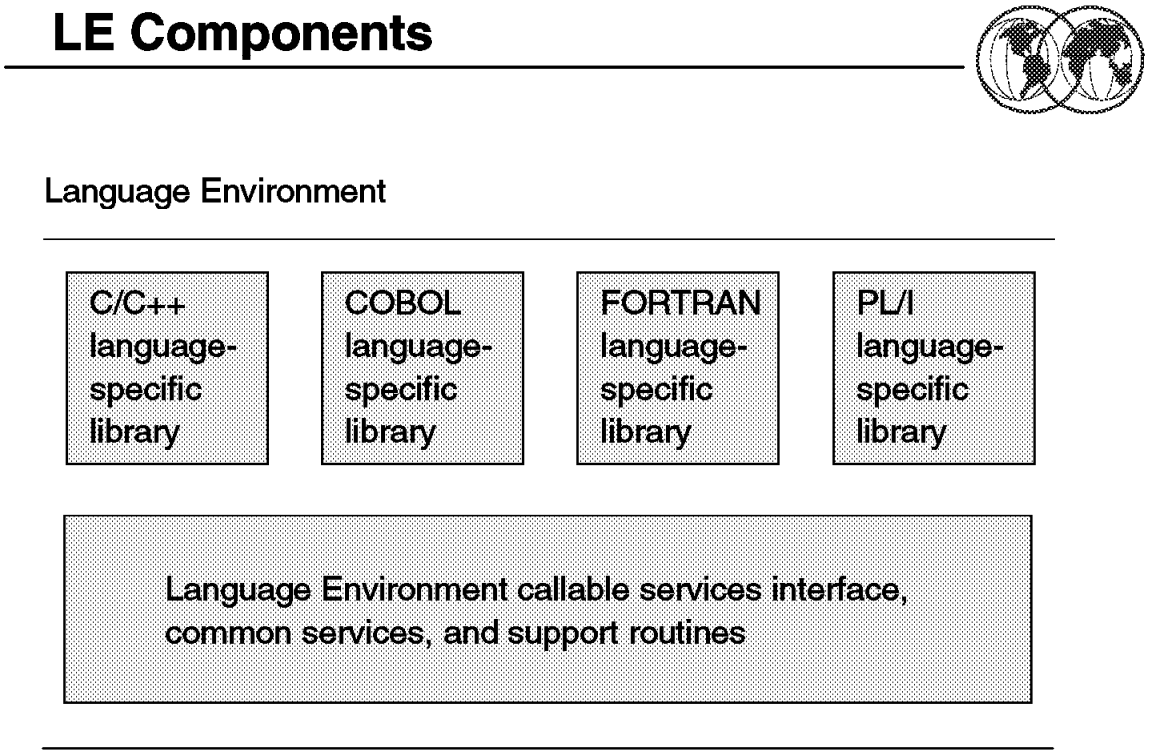


Figure 174. LE components

4.1.2 LE components

The visual shows the separate components that make up Language Environment.

Language Environment consists of:

- *Basic routines* that support starting and stopping programs, allocating storage, communicating with programs written in different languages, and indicating and handling conditions
- *Common library services*, such as math services and date and time services, that are commonly needed by programs running on the system. These functions are supported through a library of callable services.
- *Language-specific portions of the run-time library*, because many language-specific routines call Language Environment services. However, behavior is consistent across languages.

POSIX support is provided in the Language Environment base and in the C language-specific library.

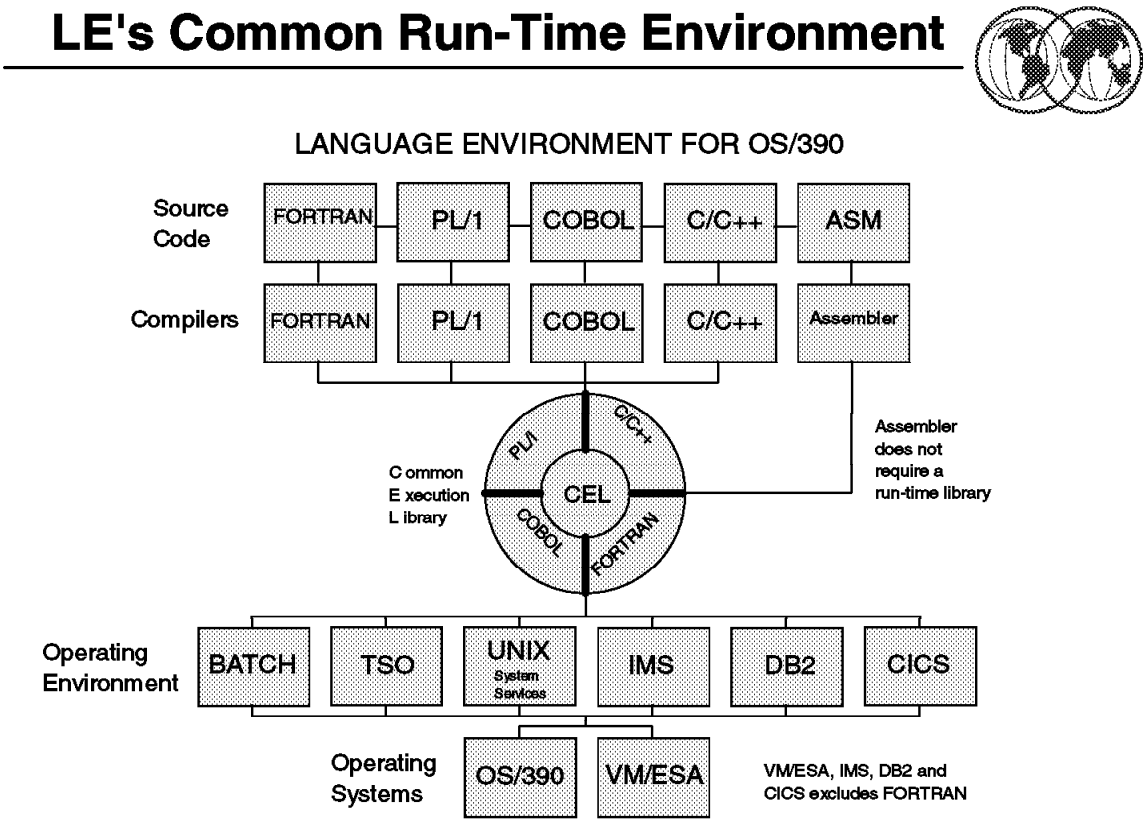


Figure 175. LE

4.1.3 LE′s common run-time environment

The graphic illustrates the common environment that Language Environment creates. It also shows that each HLL has its specific run-time and SYSLIB libraries, and shares with others HLLs a Common Execution Library (CEL). The graphic further shows that the load modules produced in such way can be executed in different operating environments under OS/390 or VM/ESA.

4.1.3.1 Using Language Environment

Language Environment helps you create mixed-language applications and gives you a consistent method of accessing common, frequently used services. Building mixed-language applications is easier with Language Environment-conforming routines because Language Environment establishes a consistent environment for all languages in the application.

Language Environment provides the base for future IBM language library enhancements in the OS/390 and VM environments. Many system dependencies have been removed from Language Environment-conforming language products.

Because Language Environment provides a common library, with services that you can call through a common callable interface, the behavior of your applications will be easier to predict. Language Environment′s common library includes common services such as messages, date and time functions, math functions, application utilities, system services, and subsystem support.

The following example illustrates how to invoke a Language Environment service in COBOL for OS/390:

```
ALL "CEESSSQT" using argument, feedback-code, result
```

You should use a CALL statement with the correct parameters for that particular service.

The language-specific portions of Language Environment provide language interfaces and specific services that are supported for each individual language. Language Environment is accessed through defined common calling conventions.

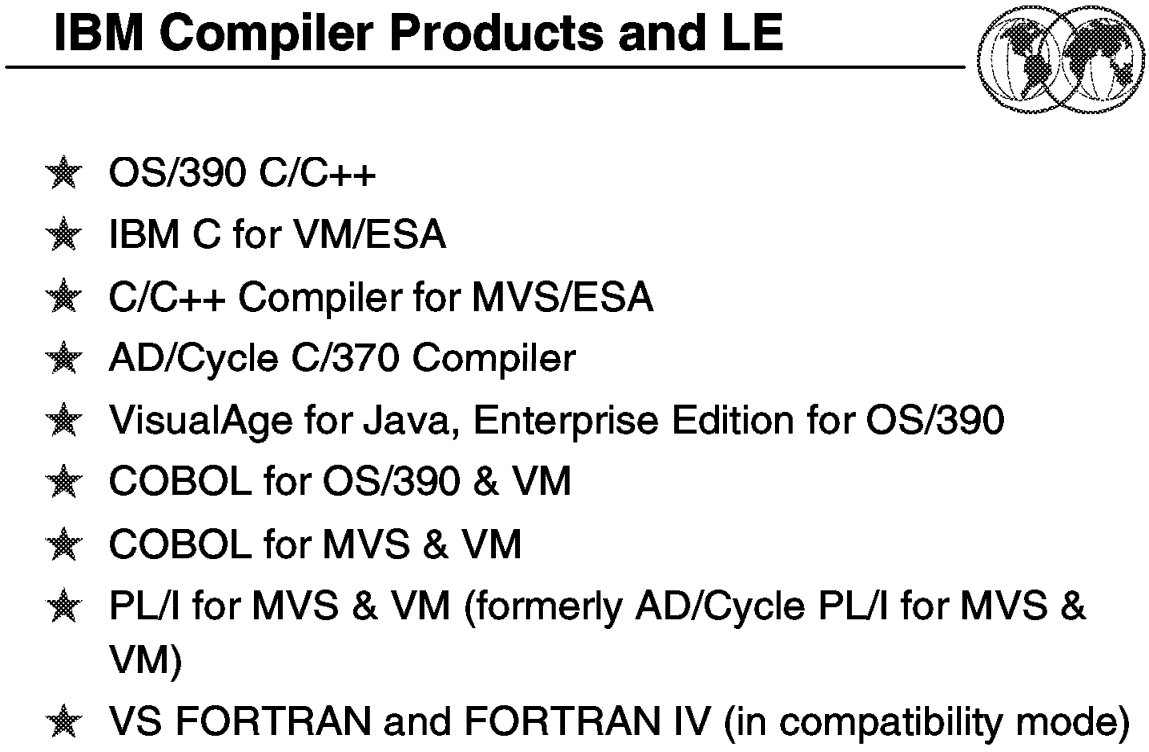


Figure 176. HLLs demanding LE

4.1.4 HLLs demanding LE

OS/390 Language Environment for OS/390 & VM is the prerequisite run-time environment for applications generated with the following IBM compiler products:

- OS/390 C/C++
- IBM C for VM/ESA
- C/C++ Compiler for MVS/ESA
- AD/Cycle C/370 Compiler
- VisualAge for Java, Enterprise Edition for OS/390
- COBOL for OS/390 & VM
- COBOL for MVS & VM
- PL/I for MVS & VM (formerly AD/Cycle PL/I for MVS & VM)
- VS FORTRAN and FORTRAN IV (in compatibility mode)

Language Environment supports, but is not required for, VS FORTRAN Version 2 compiled code (OS/390 only).

In many cases, you can run compiled code generated from the previous versions of the above compilers. A set of assembler macros is also provided to allow assembler routines to run with Language Environment.

For more information on IBM VisualAge for Java, Enterprise Edition for OS/390, refer to the product documentation.

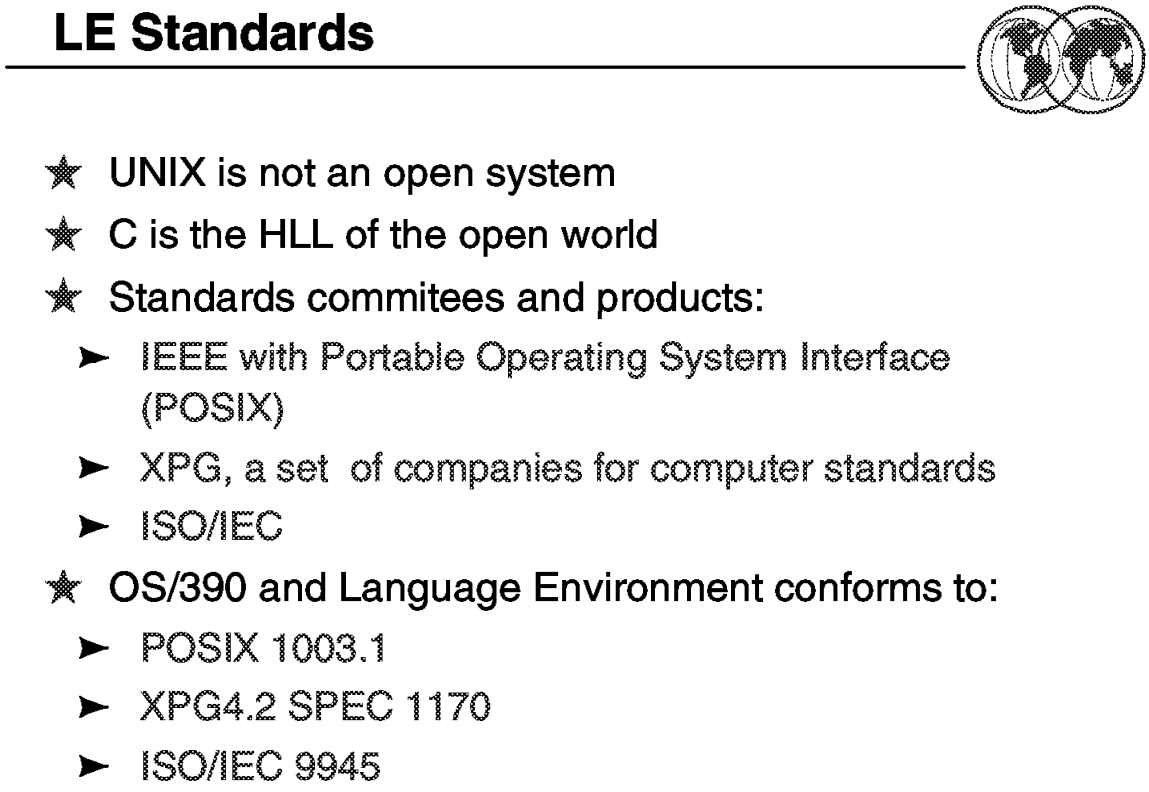


Figure 177. LE standards

4.1.5 LE standards

UNIX is not a real Open System due to having many different implementations that are incompatible with one another (more than 70 UNIX variants populated the market). Also, the source code was delivered together with the product, which makes each copy potentially unique in itself.

The UNIX market is very competitive, so each software house adds other functions to the kernel. This creates a “UNIX Proprietary.”

To address the problem, standards like Portable Operating System Interface (POSIX) were introduced to define interfaces based on UNIX.

The IEEE Portable Operating System Interface (POSIX) standard is a series of industry standards for code and user interface portability. POSIX support allows applications written for a UNIX-like operating system to be run on OS/390. C language programmers can access operating system services through a set of standard language bindings. C language programmers who install OS/390 UNIX System Services (OS/390 UNIX) and Language Environment for OS/390 & VM can call C language functions defined in the POSIX standard from their C applications and can run applications that conform to ISO/IEC 9945-1:1990, which is also ANSI-IEEE 1003.1-1990, is based on the POSIX.1 standard. C language programmers with OS/390 UNIX installed can also call a subset of the proposed programming interface for thread management (a subset of draft 6 of POSIX .4a). Through C interfaces, Language Environment functions conform to XPG4.2 specifications and are branded by X/Open. In addition, C POSIX-conforming applications may use all Language Environment services.

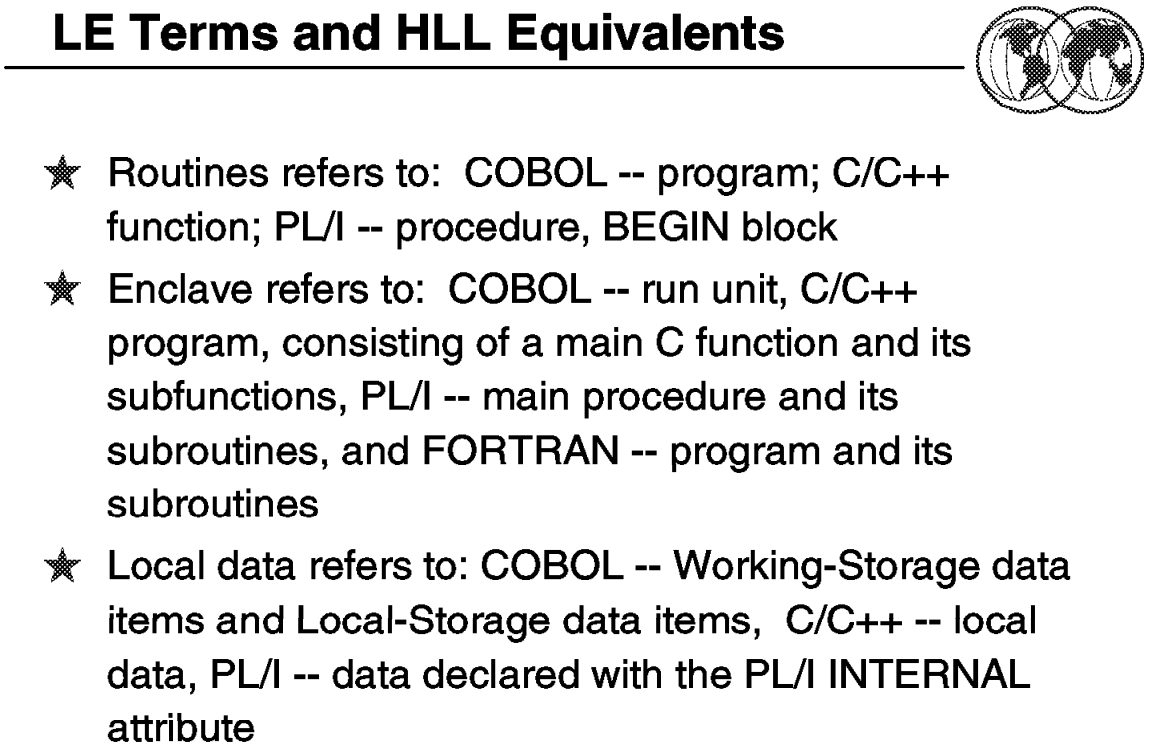


Figure 178. LE terms and HLL equivalents

4.1.6 LE terms and HLL equivalents

The Language Environment program management model provides a framework within which an application runs. It is the foundation of all of the component models--condition handling, run-time message handling, and storage management--that comprise the Language Environment architecture. The program management model defines the effects of programming language semantics in mixed-language applications and integrates transaction processing and multithreading.

Some terms used to describe the program management model are common programming terms; other terms are described differently in other languages. It is important that you understand the meaning of the terminology in a Language Environment context as compared to other contexts.

4.1.6.1 General programming terms

- Application program

  A collection of one or more programs cooperating to achieve particular objectives such as inventory control or payroll.

- Environment

  In Language Environment, normally a reference to the run-time environment of HLL the enclave level.

4.1.6.2 LE terms and HLL equivalents:

- Routine

  In Language Environment, refers to a procedure, or a function, or a subroutine.

  Equivalent HLL terms: COBOL--program; C/C++--function; PL/I--procedure, BEGIN block.

- Enclave

  The enclave defines the scope of HLL semantics. In Language Environment, it means a collection of routines, one of which is named as the main routine. The enclave contains at least one thread.

  Equivalent HLL terms: COBOL--run unit, C/C++--program, consisting of a main C function and its sub-functions, PL/I--main procedure and its subroutines, and FORTRAN--program and its subroutines.

- Process

  The highest level of the Language Environment program management model. A process is a collection of resources, both program code and data, and consists of at least one enclave.

- Thread

  An execution construct that consists of synchronous invocations and terminations of routines. The thread is the basic run-time path within the Language Environment program management model, and is dispatched by the system with its own run-time stack, instruction counter, and registers. Threads may exist concurrently with other threads.

Terminology for Data:

- Automatic data

  Data that does not persist across calls. It is allocated with the same value on entry and reentry into a routine.

- External data

  Data that can be referenced by multiple routines and data areas. External data is known throughout an enclave.

- Local data

  Data that is known only to the routine in which it is declared.

  Equivalent HLL terms: C/C++--local data, COBOL--WORKING-STORAGE data items and LOCAL-STORAGE data items, PL/I--data declared with the PL/I INTERNAL attribute.

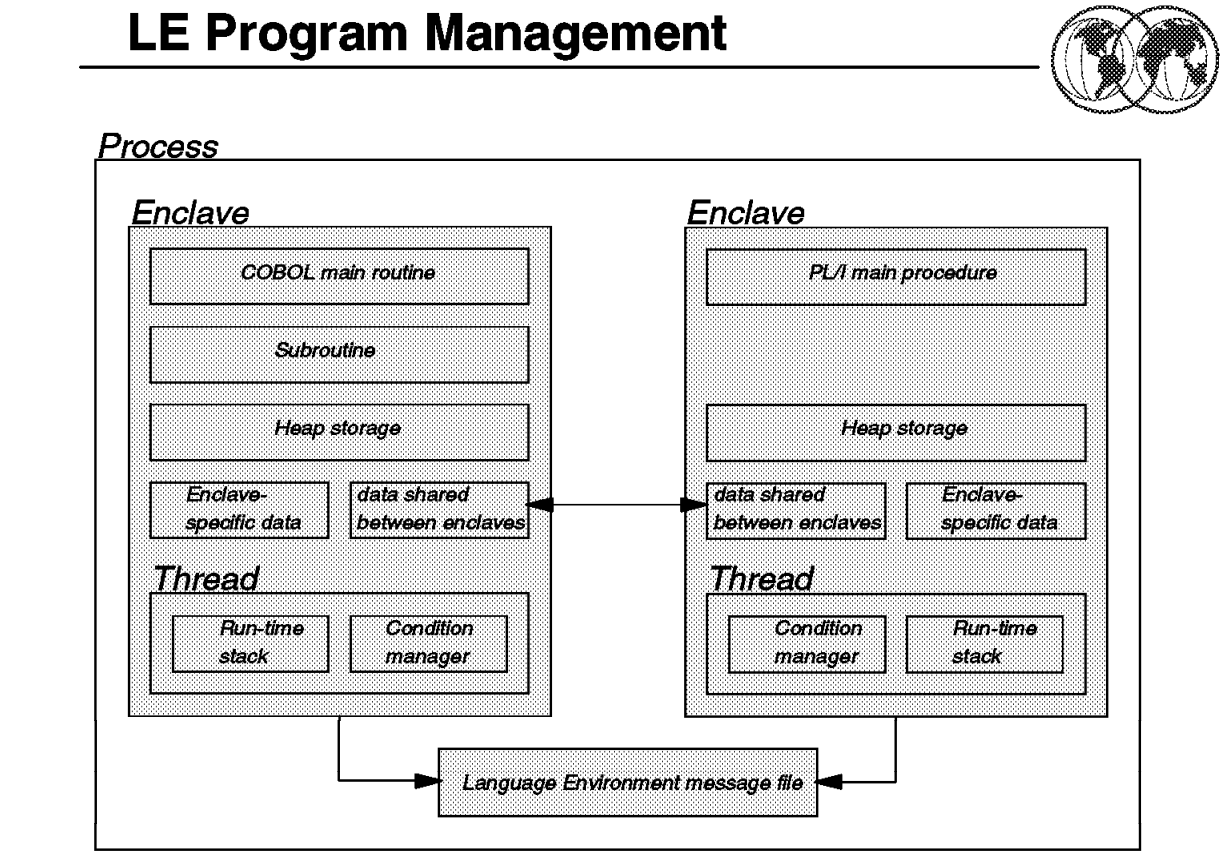


Figure 179. LE program management

4.1.7 LE program management

Three entities-- process, enclave, and thread--are at the core of the Language Environment program management model.

The visual shows the relationship between processes, enclaves, and threads. It illustrates the simplest form of the Language Environment program management model and how resources such as storage are managed.

4.1.7.1 Process

The highest level component of the Language Environment program model is the process. A process consists of at least one enclave and is logically separate from other processes. Processes do not share storage and are independent of and equal to each other; they are not hierarchically related.

Language Environment generally does not allow language file sharing across enclaves nor does it provide the ability to access collections of externally stored data. However, the PL/I standard SYSPRINT file may be shared across enclaves. The Language Environment message file also may be shared across enclaves, since it is managed at the process level. The Language Environment message file contains messages from all routines running within a process, making it a useful central location for messages generated during run time.

Processes can create new processes and communicate to each other by using Language Environment-defined communication, for such things as indicating when a created process has been terminated.

4.1.7.2 Enclaves

A key feature of the program management model is the enclave, a collection of the routines that make up an application. As mentioned in the terminology previously defined, the enclave is the equivalent of any of the following:

- A run unit, in COBOL
- A program, consisting of a main C function and its sub-functions, in C
- A main procedure and all of its subroutines, in PL/I
- A program and its subroutines, in FORTRAN

The enclave consists of one main routine and zero or more subroutines. The main routine is the first to execute in an enclave; all subsequent routines are named as subroutines.

4.1.7.3 Threads

Each enclave consists of at least one thread, the basic instance of a particular routine. A thread is created during enclave initialization with its own run-time stack, which keeps track of the thread′s execution, as well as a unique instruction counter, registers, and condition-handling mechanisms. Each thread represents an independent instance of a routine running under an enclave′s resources.

Threads share all of the resources of an enclave. A thread can address all storage within an enclave. All threads are equal and independent of one another and are not related hierarchically. A thread can create a new enclave. Because threads operate with unique run-time stacks, they can run concurrently within an enclave and allocate and free their own storage. Because they may execute concurrently, threads can be used to implement parallel processing applications and event-driven applications.

The figure illustrates the full Language Environment program model, with its multiple processes, enclaves, and threads.

It shows each process is within its own address space. An enclave consists of one main routine, with any number of subroutines. A main routine might not be active at all times in a POSIX application, if the thread in which the main routine executes terminates before the other threads that it created.

External data is available only within the enclave where it resides; notice that even though the external data may have identical names in different enclaves, the external data is unique to the enclave. The scope of external data, as described earlier, is the enclave. The threads can create enclaves, which can create more threads, and so on.

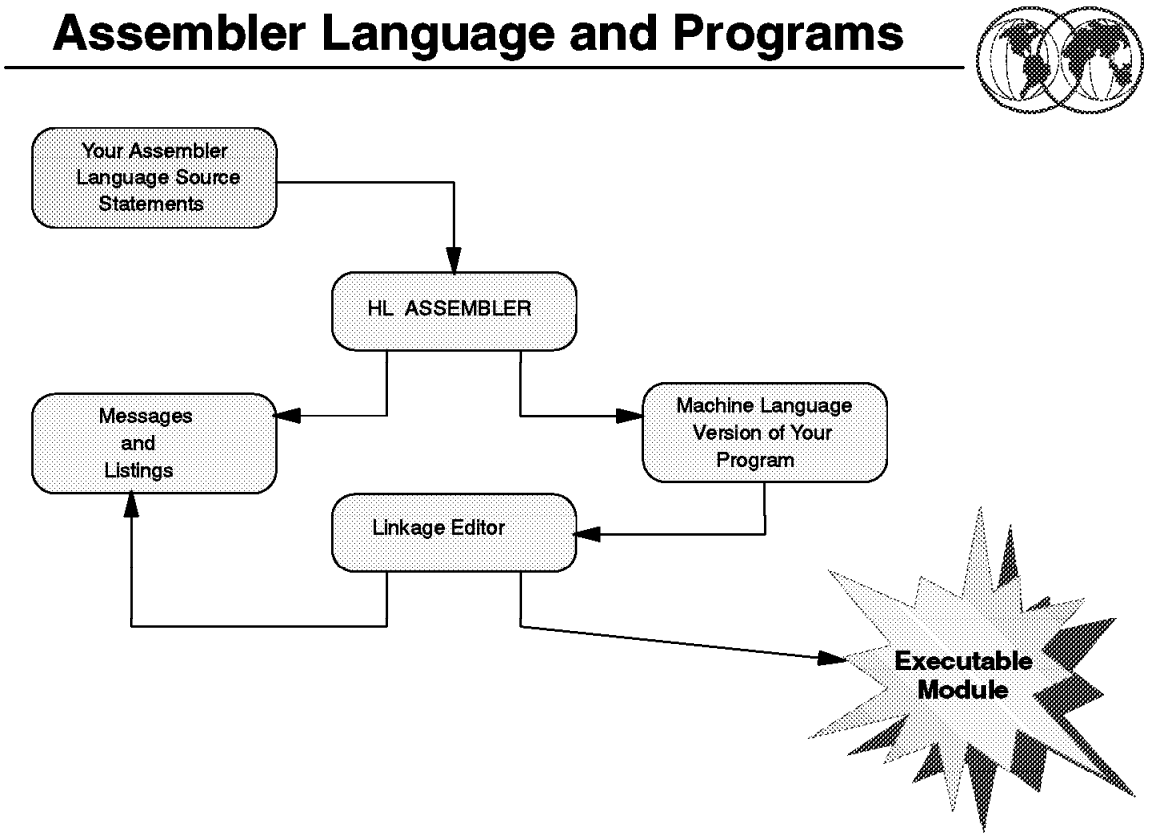


Figure 180. Assembler language and programs

4.1.8 Assembler language and programs

A computer can understand and interpret only machine language. Machine language is in binary form and, thus, is very difficult to write. The assembler language is a symbolic programming language that you can use to code instructions instead of coding in machine language.

Because the assembler language lets you use meaningful symbols made up of alphabetic and numeric characters, instead of just the binary digits 0 and 1 used in machine language, you can make your coding easier to read, understand, and change. Assembler is also enriched by a huge set of macros that make it a very powerful language; however, its major quality is the performance of its object code at running time.

One of the most key aspects of a language is its readability and and capacity to document itself. However, certain modern languages like C do not follow this rule. Then, we may say that if you write an assembler program using all the documenting capability of its features, chances are that you may have a more readable code than a HLL.

The Assembler must translate the symbolic assembler language into machine language before the computer can run your program. The specific procedures followed to do this may vary according to the system you are using. However, the method is basically the same for all systems and consists of the following:

Your program, written in the assembler language, becomes the *source module* that is input to the Assembler. The Assembler processes your source module and produces an *object module* in machine language (called object code). The object module can be used as input to be processed by the linkage

editor or the binder. The linkage editor or binder produces a *load module* that can be loaded later into the main storage of the computer. When your program is loaded, it can then be run. Your source module and the object code produced are printed, along with other information, on a program listing.

4.1.8.1 Assembler language

The assembler language is the symbolic programming language that lies closest to the machine language in form and content. You will, therefore, find the assembler language useful when:

- You need to control your program closely, down to the byte level and even to the bit level.
- You must write subroutines for functions that are not provided by other symbolic programming languages, such as COBOL, FORTRAN, or PL/I.
- You need good performance at execution time.

The assembler language is made up of statements that represent either instructions or comments. The instruction statements are the working part of the language and are divided into the following three groups:

- Machine instructions

  A machine instruction is the symbolic representation of a machine language instruction. It is called a machine instruction because the Assembler translates it into the machine language code that the computer can run.

- Assembler instructions

  An assembler instruction is a request to the Assembler to do certain operations during the assembly of a source module; for example, defining data constants, reserving storage areas, and defining the end of the source module. Except for the instructions that define constants, and the instruction used to generate no-operation instructions for alignment, the Assembler does not translate assembler instructions into object code.

- Macro instructions

  A macro instruction is a request to the Assembler program to process a predefined sequence of instructions called a macro definition. From this definition, the Assembler generates machine and assembler instructions, which it then processes as if they were part of the original input in the source module.

  IBM supplies macro definitions for input/output, data management, and supervisor operations that you can call for processing by coding the required macro instruction.

  You can also prepare your own macro definitions, and call them by coding the corresponding macro instructions. Rather than code all of this sequence each time it is needed, you can create a macro instruction to represent the sequence and then, each time the sequence is needed, simply code the macro instruction statement. During assembly, the sequence of instructions represented by the macro instruction is inserted into the source program.

4.1.8.2 Assembler program relationship to OS/390

The Assembler program, also referred to as the Assembler, processes: the machine, assembler instructions, and macro instructions you have coded (source statements) in the assembler language, and produces an object module in machine language.

The High Level Assembler operates under the OS/390 operating system. This operating system provides the Assembler with services for:

- Assembling a source module
- Running the assembled object module as a program

In writing a source module, you must include instructions that request any required service functions from the operating system. OS/390 provides the following services:

- For assembling the source module:
  - A control program
  - Sequential data sets to contain source code
  - Libraries to contain source code and macro definitions
  - Utilities
- For preparing for the execution of the Assembler program as represented by the object module:
  - A control program
  - Storage allocation
  - Input and output facilities

It can be very difficult to write an assembler language program using only machine instructions. The Assembler provides additional functions, not discussed here, that make this task easier.

Sample Assembler Routine

```
MAIN      CEEENTRY  PPA=MAINPPA
*
          LA        1,PARMLIST
          L         15,=V(CEEMOUT)
          BALR      14,15
*
*  Terminate the Language Environment environment
*
          CEETERM   RC=0,MODIFIER=0
* ================================================================
*           CONSTANTS AND WORKAREAS
* ================================================================
PARMLIST  DC        AL4(STRING)
          DC        AL4(DEST)
          DC        X'80000000'      Omitted feedback code
STRING    DC        AL2(STRLEN)
STRBEGIN  DC        CL19'In the main routine'
STRLEN    EQU       *-STRBEGIN
DEST      DC        F'2'
MAINPPA   CEEPPA                     Constants describing the code block
          CEEDSA                     Mapping of the dynamic save area
          CEECAA                     Mapping of the common anchor area
          END       MAIN             Nominate MAIN as the entry point
```

Figure 181. Sample assembler routine

4.1.9 Sample assembler routine

This visual shows a simple main assembler routine (source code) that brings up the environment, returns with a return code of 0, modifier of 0, and prints a message in the main routine.

Chapter 5. Infoprint Server

Infoprint Server is an optional feature of OS/390 Version 2 Release 8 that uses OS/390 UNIX System Services. This feature is the basis for a total print serving solution for the OS/390 environment. It lets you consolidate your print workload from many servers onto a central OS/390 print server

Note: Beginning with OS/390 Version 2 Release 5, this product was called OS/390 Print Server.

Infoprint Server delivers improved efficiency and lower overall printing cost with the flexibility for high-volume, high-speed printing from anywhere in the network. With Infoprint Server, you can reduce the overall cost of printing while improving manageability, data retrievability, and usability.

The IP PrintWay/NetSpool feature available in OS/390 Version 1 Release 3 and Version 2 Release 4 is now a part of Infoprint Server. IP PrintWay allows you fast and reliable access to Transmission Control Protocol/Internet Protocol (TCP/IP)-connected printers. NetSpool automatically directs Virtual Telecommunications Access Method (VTAM) application data to the job entry subsystem (JES) spool without requiring application changes.

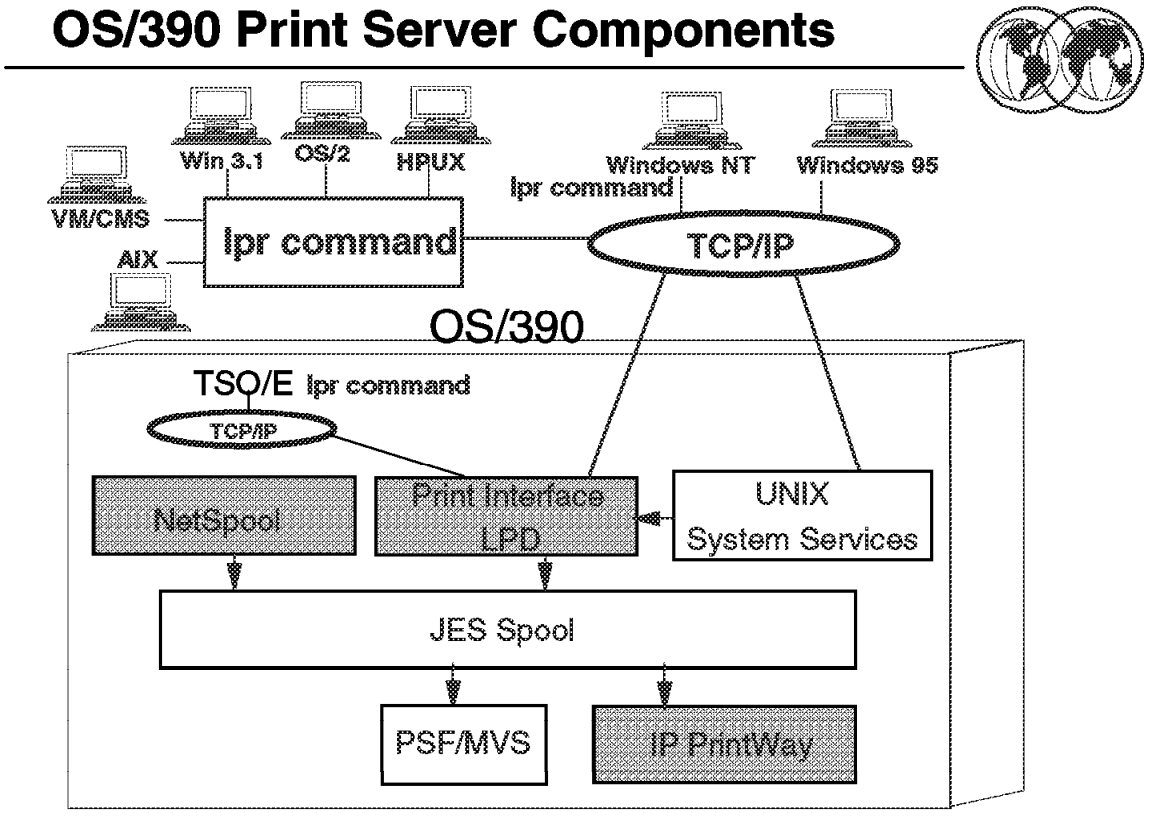


Figure 182. OS/390 Print Server components

5.1 OS/390 Print Server

With the introduction of the OS/390 Print Server, a new optional feature of OS/390 Release 5, users have the opportunity to consolidate their print workload on OS/390. This new function allows access to fast and reliable AFP printers, or TCP/IP-connected printers from OS/390, including UNIX services and LAN clients. Users can define their printers in a central repository, allowing clients in the network to use any printer in the enterprise that is registered to the OS/390 Print Server.

The OS/390 Print Server is the framework for a total print serving solution for the OS/390 system environment. It extends the functions provided by the optional IP PrintWay and NetSpool features in OS/390 Version 1 Release 3 and OS/390 Version 2 Release 4. IP PrintWay and NetSpool are now part of the OS/390 Print Server.

With the addition of a new Print Interface component, the OS/390 Print Server provides an end-to-end integrated solution from print submission to the printer. The IBM OS/390 Print Server consists of three components, as shown in the visual.

- OS/390 Print Interface

  The OS/390 Print Interface is the central server element of the IBM OS/390 Print Server. It consists of an LPD that allocates data sets on the JES spool using information from the printer definitions in the printer inventory.

  The OS/390 Print Interface address space, shown in the visual, runs as an LPD on the OS/390 system. It provides the following functions:

- It receives print requests and dynamically allocates a data set on the JES spool for each data set to be printed.
- It responds to query requests and returns the status of the data set on the JES spool or a list of printer names, locations, and descriptions.
- It removes data sets from the JES spool that have not been selected for printing.

The OS/390 Print Interface receives print requests that are submitted using TCP/IP protocol from:

- Remote systems in the TCP/IP LAN network
- OS/390 UNIX System Services (OpenEdition)
- A Windows 95 or Windows NT system
- A local OS/390 system

- IP PrintWay

  IP PrintWay can use standard LPR/LPD or direct socket printing protocol to route JES2 or JES3 print data from OS/390 to another system′s spool or to a printer in the TCP/IP network. Depending upon selected options, the print data is sent as is (binary format), or translated from EBCDIC into ASCII for the target system or printer. IP PrintWay is better than the TCP/IP Network Print Facility (NPF) for MVS in usability, performance, capacity and function, and is the strategic replacement for NPF.

- NetSpool

  NetSpool allows the user to automatically reroute VTAM application output (such as from CICS or IMS) to the JES spool without requiring application program changes. Application output can then be printed to any server or printer that is connected to the TCP/IP network using IP PrintWay, or to an AFP printer using Print Services Facility (PSF)/MVS.

  Note: This component of the IBM OS/390 Print Server is optional.

The following operating systems, shown in the visual, can use the LPR command to send print requests to the Print Interface:

- AIX 4.1.4 or 4.2.X
- OS/2
- VM/CMS
- HPUX
- Win 3.1

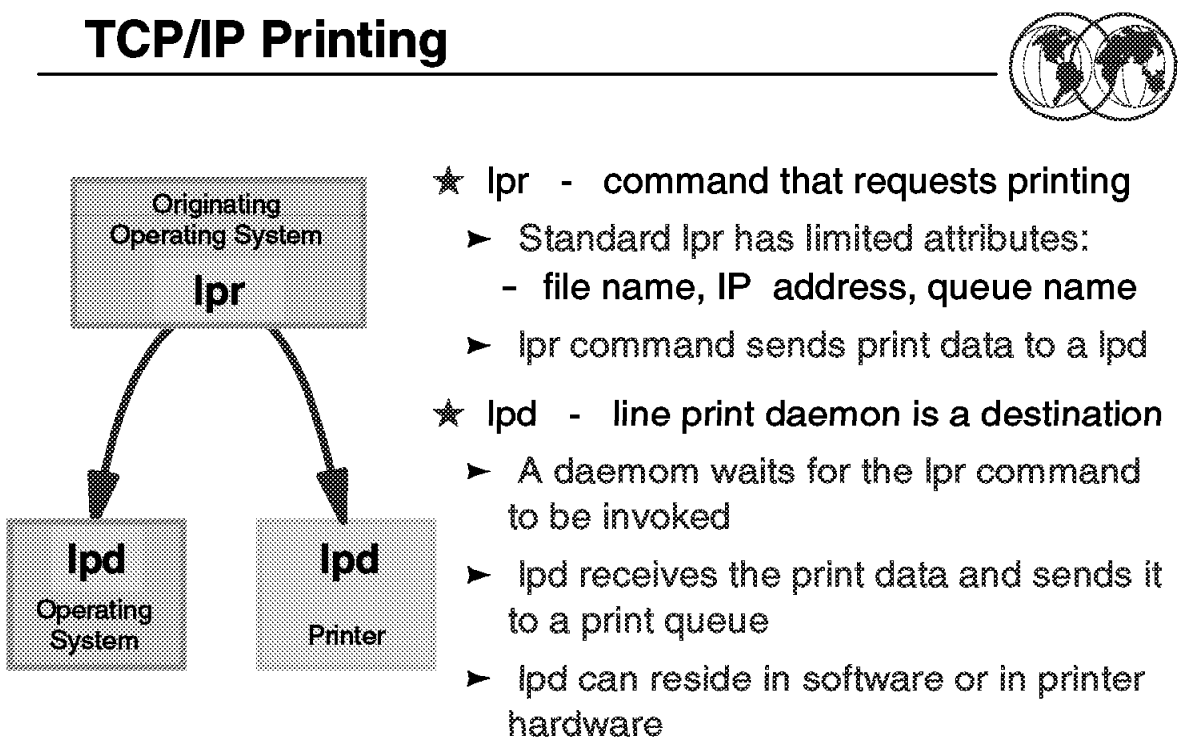


Figure 183. TCP/IP Print Protocol

5.1.1 TCP/IP Print Protocol

TCP/IP provides client and server support for remote printing by supporting the following commands:

LPR Line print requestor (LPR) is the command that requests printing. The standard LPR command has attributes such as file name, IP address, and queue name. The LPR command sends print data to an LPD.

LPD Line print daemon (LPD) is like a destination. An LPD daemon waits for the LPR command to be invoked. The LPD daemon receives the print data and sends it to a print queue. It can reside in the software or in the printer hardware, as shown in the visual.

The LPR command and the LPD function are part of IBM TCP/IP of the OS/390 base feature.

You can print from the following environments by using the TCP/IP LPR command:

- AIX 4.1.4 or 4.2.x
- OS/2
- VM/CMS
- A remote OS/390 system using TSO/E
- A local OS/390 system using TSO/E

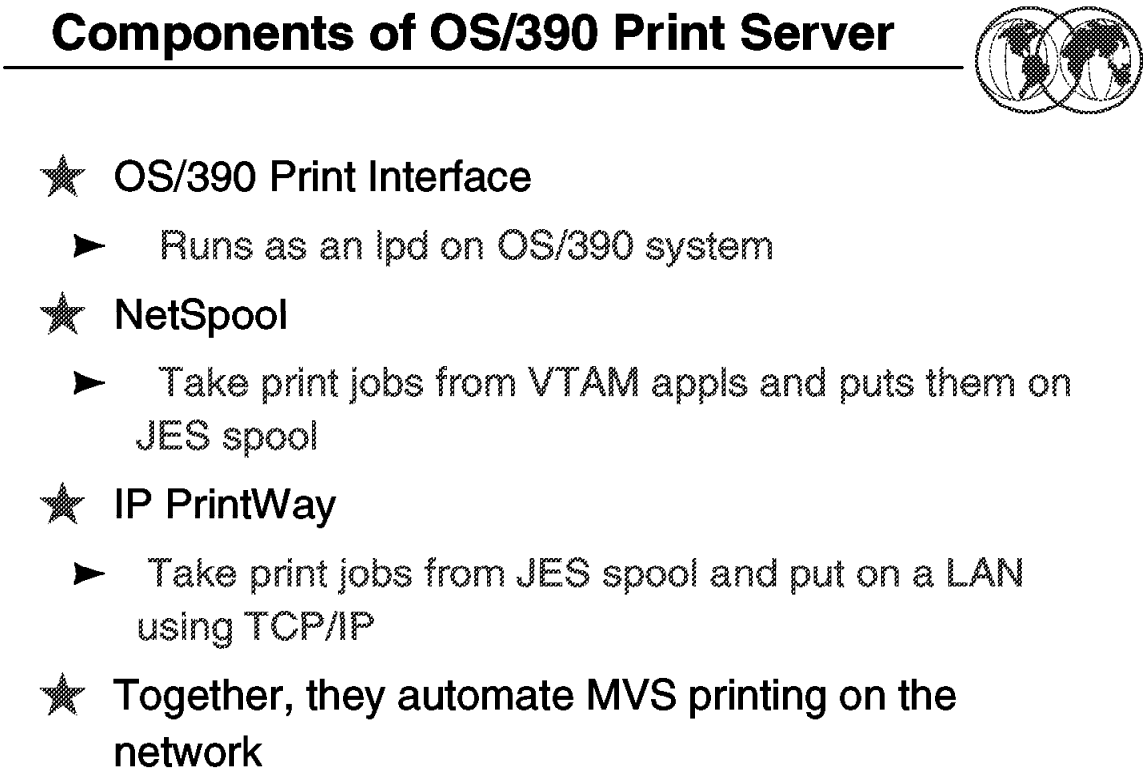


Figure 184. Components of OS/390 Print Server

5.1.2 Components of OS/390 Print Server

The OS/390 Print Server consists of three major components in OS/390 Releases 5, 6, and 7. They are as follows:

- Print Interface

  The Print Interface component of Infoprint Server runs on the OS/390 system. It provides an LPD that receives print requests from remote workstations that have TCP/IP access and from the OS/390 UNIX System Services shell printing commands on the local OS/390 system.

- NetSpool

  NetSpool intercepts print data from VTAM applications, such as CICS and IMS. NetSpool converts the data into S/390 line data and creates output data sets on the JES2 or JES3 spool. JES or PSF for OS/390 can print the output data sets or transmit them to another location for printing. Alternatively, IP PrintWay can transmit the data sets to a remote printer in your TCP/IP network.

- IP PrintWay

  IP PrintWay transmits output data sets from JES2 or JES3 to remote printers or to host systems in your TCP/IP network. A print server can be running on the host system. The remote printer or host system must support either the LPR/LPD protocol, the IPP protocol, or direct socket printing

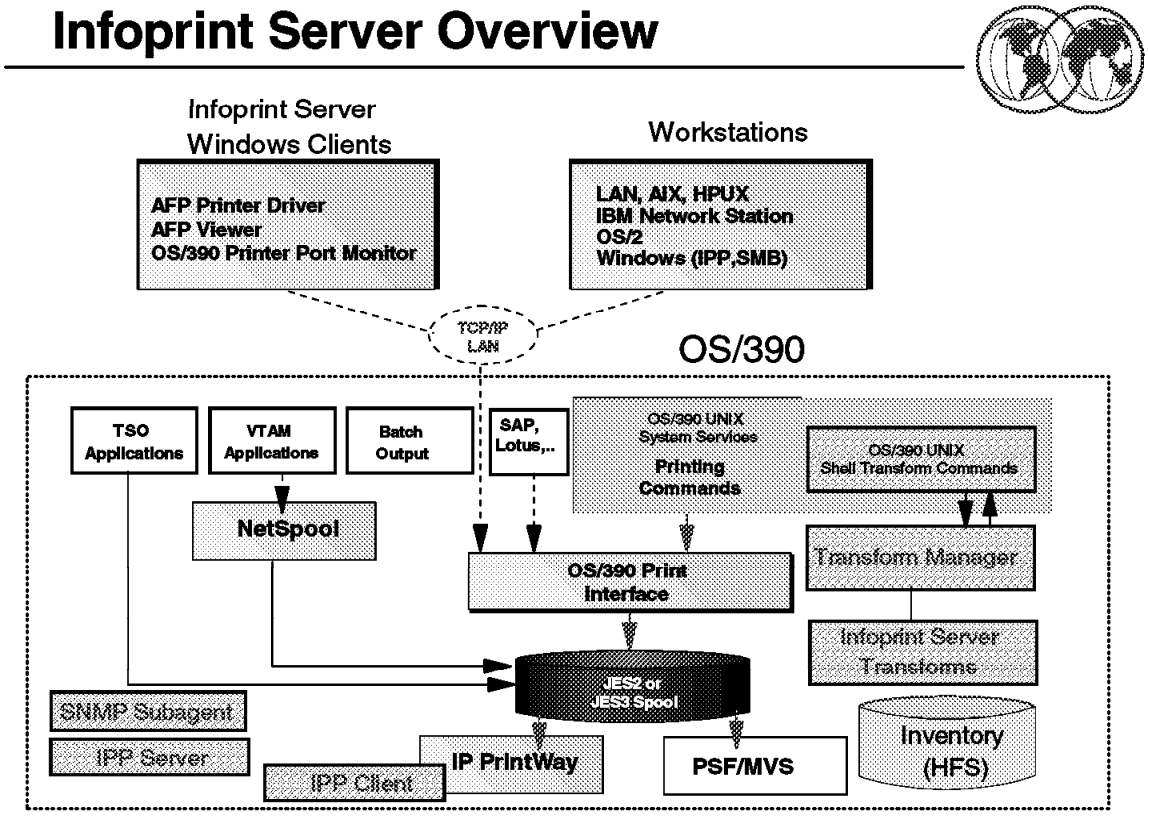


Figure 185. Infoprint Server overview

5.2 Infoprint Server overview

Infoprint Server for OS/390, an optional element of OS/390 Version 2 Release 8, provides support for LAN and host printing on OS/390. Infoprint Server consists of several components that work together to provide printing services. The visual shows some of the components and how they fit into your system. The components and features of Infoprint Server are shaded in the figure and are as follows:

- Printer Inventory and Printer Inventory Manager

  The Printer Inventory Manager controls the Printer Inventory, a set of files in the hierarchical file system (HFS) that contain information about each printer to which NetSpool, Print Interface, and IP PrintWay can print. The Printer Inventory also contains system configuration information for IP PrintWay and, optionally, for PSF for OS/390.

- Infoprint Server Windows client

  The Windows client provides (1) an AFP printer driver, (2) an AFP viewer plug-in, and (3) an OS/390 printer port monitor that sends print requests to the Print Interface component.

- Print Interface

  Print Interface processes print requests from remote clients that use any of the following TCP/IP printing protocols:

  – Line printer requester (LPR) to line printer daemon (LPD)
  – Internet Printing Protocol (IPP)

Print Interface also provides OS/390 UNIX shell commands (lp, lpstat, and cancel) and the AOPPRINT JCL procedure to let local users submit print requests to Print Interface.

Print Interface accepts any data format the target printer can print, converts data to EBCDIC or ASCII as required by the target printer, and allocates output data sets on the JES spool. Print Interface can also transform PCL, PDF, PostScript, and SAP data to AFP format prior to writing data to the JES spool, for printing on IBM AFP printers.

- Infoprint Server Transforms for OS/390 and the Transform Manager

  Infoprint Server Transforms for OS/390 is a Licensed Program Product (5697-F51) that provides PCL, PostScript, PDF, and SAP to AFP transforms for the OS/390 system. The Transform Manager component of Infoprint Server manages the PCL, PostScript, and PDF transforms.

  Infoprint Server Transforms also provides OS/390 UNIX shell commands (pcl2afp, ps2afp and its alias pdf2afp, and sap2afp) to let local users transform data without printing it.

- NetSpool NetSpool processes print requests from VTAM applications, such as CICS and IMS. NetSpool accepts SCS, 3270, and binary data and allocates output data sets on the JES spool.

- IP PrintWay IP PrintWay transmits data sets from the JES spool to printers or print servers using any of the following TCP/IP protocols:

  - Line printer requester (LPR) to line printer daemon (LPD)
  - Internet Printing Protocol (IPP)
  - Direct socket printing

- Simple Network Management Protocol (SNMP) subagent

  The SNMP subagent lets you use an SNMP manager to view printer characteristics and printer status for printers controlled by PSF for OS/390 that do not have internal SNMP agents or are not TCP/IP-attached to PSF.

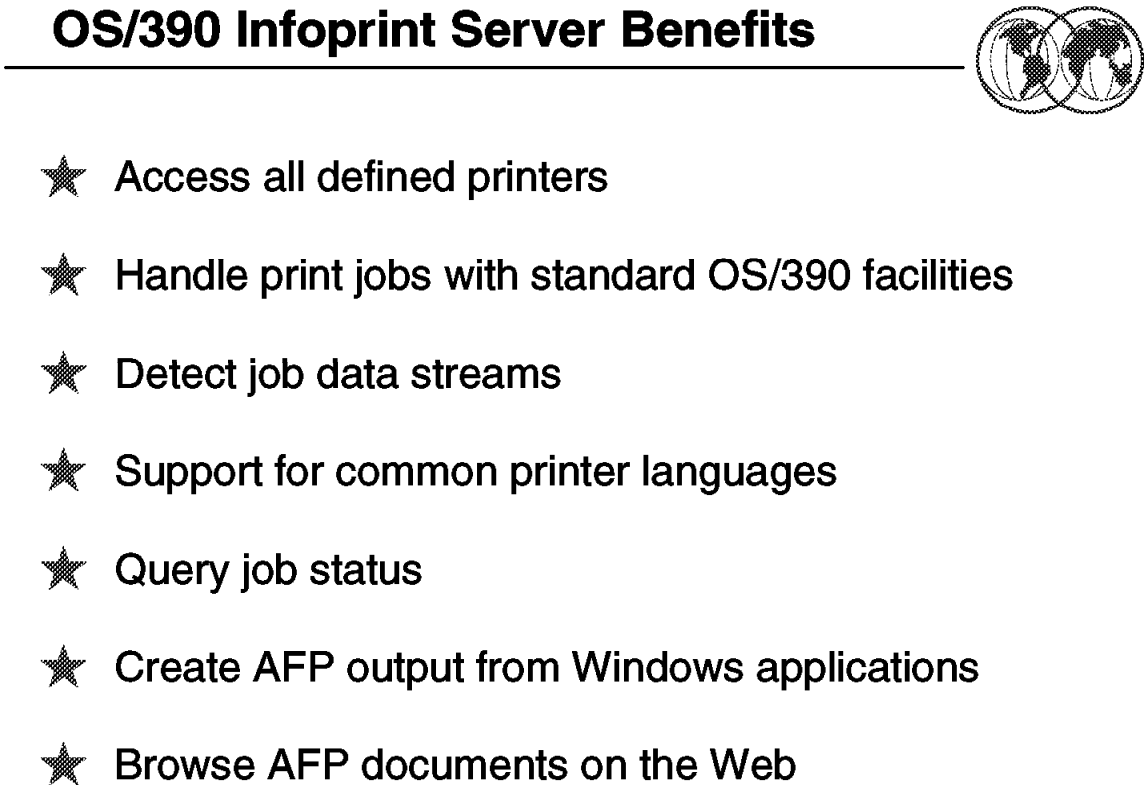


Figure 186. OS/390 Infoprint Server benefits

5.2.1 OS/390 Infoprint Server benefits

In today′s network environments, printers are often attached to a single workstation or are only available to users of a LAN. Infoprint Server lets you define all of your printers in a centralized repository. Any user in the network can send print jobs from OS/390 and LAN clients to any print that is defined to Infoprint Server. Because all components of Infoprint Server and PSF share the printer definitions, you only have to configure each printer in one place.

Because print jobs are managed by the OS/390 JES spool, they are secure and recoverable. OS/390 accounting information for print jobs is logged automatically.

The Print Interface automatically detects the data stream for jobs that LAN and OS/390 UNIX System Services clients submit. It can then ensure that the selected printer can print the data stream, thus saving paper and time.

Infoprint Server provides support for the most commonly used printer languages in the industry, including Postscript, PCL, and AFP or Mixed Object Document Content Architecture-Presentation (MO:DCA-P). With the optional Infoprint Server Transforms, Infoprint Server also supports PDF SAP OTF, and SAP ABAP. Infoprint Server protects your investment in printer hardware, while providing you with printing enhancements.

Users of network printing solutions today spend unnecessary time going to the printer to see if their jobs have printed. If they do not find them immediately, they may resubmit the jobs several times without knowing why they haven′t printed. With Infoprint Server, users in the LAN and UNIX System

Services environments can query the status of their print jobs to find out if a job is processing or is complete. Local system users can simply wait for the server to notify them that the job is complete. The users save time, and the business saves paper.

Because of its capabilities for automatic resource management, error recovery, integrated accounting, and printing from 10 to over 1000 pages per minute without application changes, AFP offers an outstanding solution for high-speed printing. Infoprint Server provides an AFP Printer Driver for Windows 95 and Windows 98 and an AFP Printer Driver for Windows NT and Windows 2000. Therefore, you can print output from any Windows application, such as Lotus WordPro or Freelance, on any of IBM′s AFP printers.

Many OS/390 applications generate documents that are formatted for AFP/Intelligent Printer Data Stream IPDS printers. You may need to view those documents in an archival system or on a Web server from your desktop. Infoprint Server includes an AFP Viewer plug-in for the Netscape Navigator and Microsoft Internet Explorer Web browsers so you can view AFP documents from your Web browser. You can also use Infoprint Server to print documents that you are viewing from a Web browser to any defined printer.

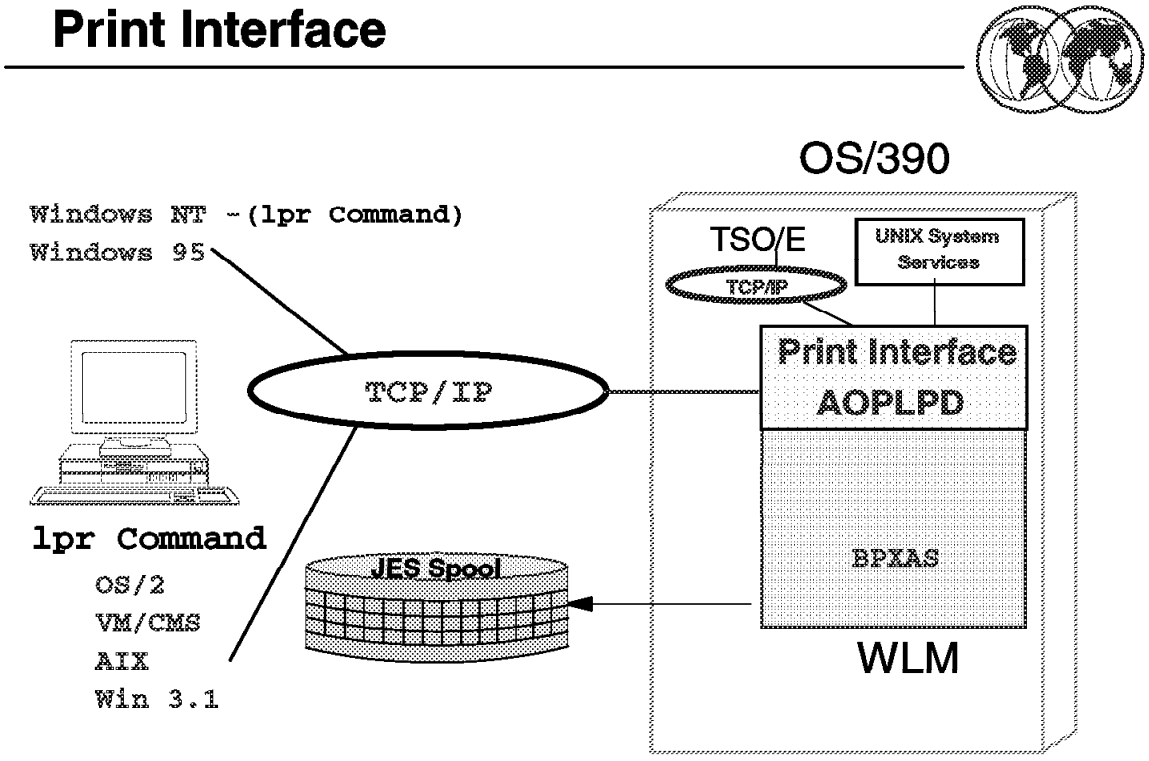


Figure 187. Print Interface

5.2.2 Print Interface

Print Interface runs as a UNIX application that uses the services of OS/390 UNIX System Services. Users can submit print requests from remote clients that use one of the following TCP/IP protocols:

- LPR to LPD. The OS/390 Printer Port Monitor for Windows and commands, such as lpr and lpq, use this protocol.
- Internet Printing Protocol (IPP) beginning with OS/390 Release 8.

Users can submit print requests from the local system with one of the following methods:

- OS/390 UNIX shell printing commands (lp, lpstat, and cancel). These commands, which adhere to the XPG4.2 standard, let users print HFS files and traditional data sets, query the status of a print job, and cancel a print job.
- The AOPPRINT JCL procedure, which lets users print HFS files and traditional data sets beginning with OS/390 Release 8.

The Print Interface performs these functions:

- It creates an output data set on the JES spool for each document to be printed. The Print Interface maps the printing options specified on lp commands and some of the printing options specified on lpr commands to JES output parameters. These parameters are the same parameters that you can specify on JCL statements when you submit batch jobs.
- It responds to query requests with the status of the output data set on the JES spool or a list of the printers known to the Print Interface.

- It removes data sets from the JES spool. The data sets must not yet have been selected for printing.

The Print Interface performs these functions:

- Printing of any data format that the printer supports

  The Print Interface allows users to submit print requests with any data format that the printer supports. These formats include, but are not limited to, PCL, PostScript, MO:DCA-P, S/390 line data, and text.

- The Print Interface detects the data format.
- Validation of print requests

  Before accepting print requests, the Print Interface validates, with some exceptions, that a document can print as requested on the selected printer. For example, the Print Interface rejects a document with a data format that the printer does not support.

- Notification of completion

  The Print Interface notifies users on the local OS/390 system when processing of a document is complete.

- Identification of printed output

  The Print Interface maintains the user ID of the job submitter for printing on separator pages. Both PSF for OS/390 and IP PrintWay allow installations to write an exit to print separator pages.

- Double-byte character set (DBCS) support

  The Print Interface converts DBCS data from one code page to another before writing the data to the JES spool.

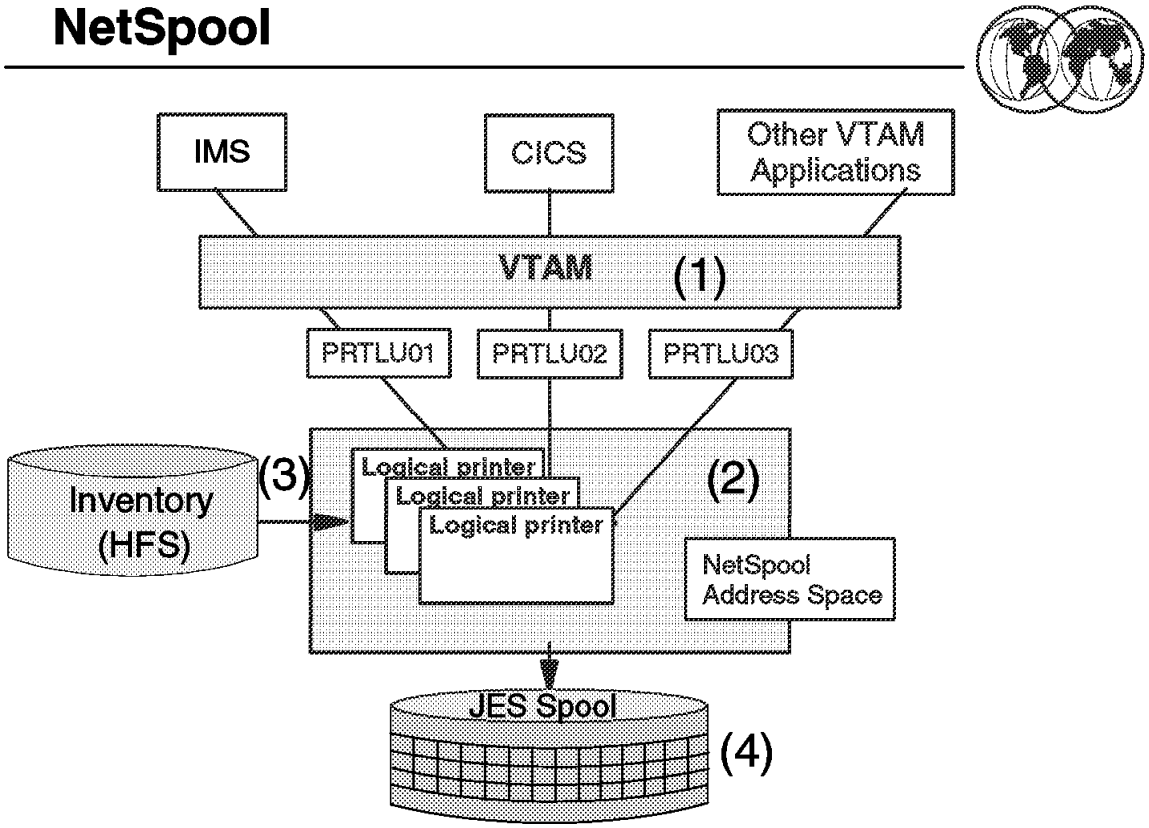


Figure 188. NetSpool

5.2.3 NetSpool

The visual shows the steps that occur from the time VTAM applications send print requests to NetSpool printer logical units (LUs) until NetSpool allocates output data sets on the JES spool. Each step is as follows:

1. VTAM applications, such as CICS or IMS, establish communication sessions with NetSpool printer logical units (LUs) instead of with SNA-network printers. Each NetSpool printer LU must be defined to VTAM as an application logical-unit (LU). NetSpool can process the following types of VTAM data streams:
   - SNA character string (SCS) data over an LU type 1 session
   - 3270 data over an LU type 3 or LU type 0 session
   - A binary data stream over an LU type 0, type 1, or type 3 session
2. NetSpool runs as a VTAM application on the same or different OS/390 system. Multiple instances of NetSpool can run simultaneously in separate address spaces; each instance of NetSpool can process VTAM print requests sent to different NetSpool printer LUs.
3. Each NetSpool printer LU must be defined in a printer definition in the Printer Inventory. NetSpool converts the data stream into S/390 line-data format and groups the data into output data sets using information in the printer definition.
4. NetSpool dynamically allocates output data sets on the JES2 or JES3 spool using JES allocation parameters specified in the printer definition, including:

- JES work-selection parameters, such as class, forms name, and destination. These parameters cause JES to direct the output data sets to the correct JES output writer or functional subsystem application (FSA), such as PSF for OS/390 or IP PrintWay.
- Advanced Function Presentation (AFP) parameters, such as the name of a form definition and page definition. PSF for OS/390 uses these parameters when printing data on IBM AFP printers.
- Distribution information, such as name and address, which can be printed on output header pages

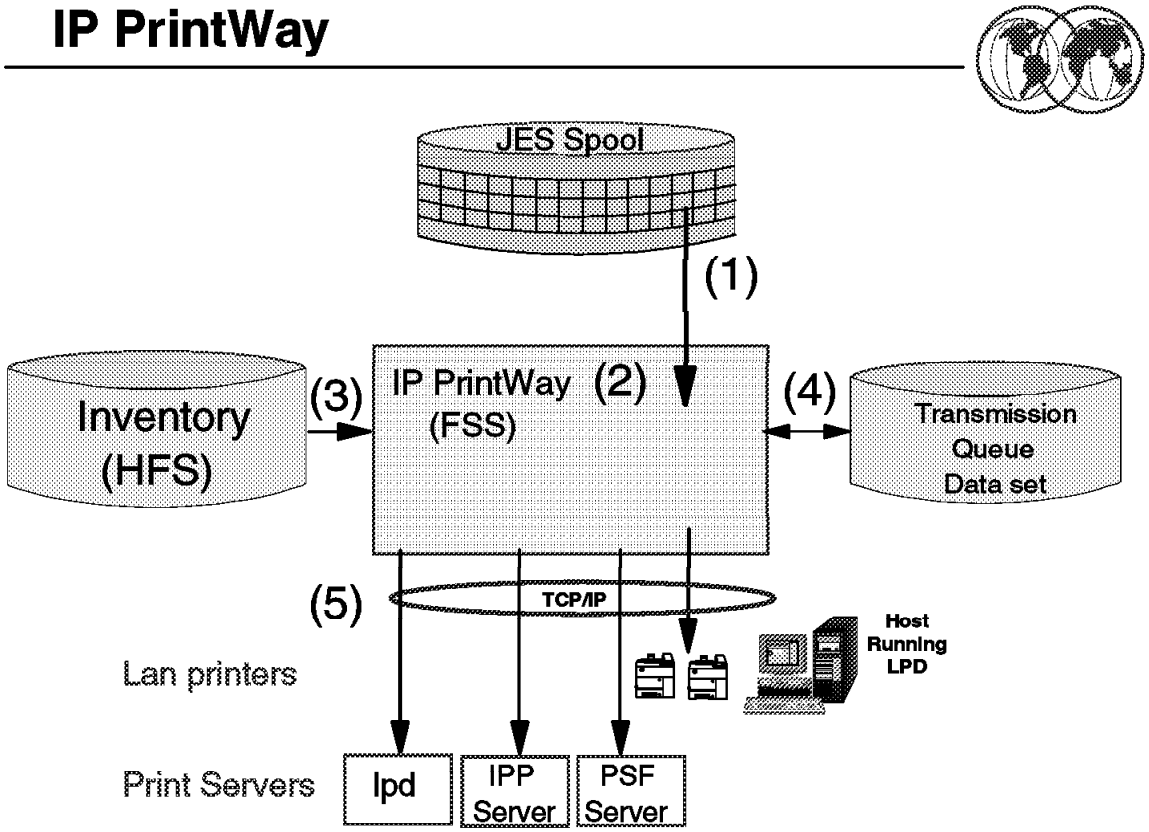


Figure 189. IP PrintWay

5.2.4 IP PrintWay

IP PrintWay transmits output data sets from the JES spool to printers or print servers in a TCP/IP network using one of the following TCP/IP protocols: LPR to LPD, IPP, or direct-socket printing.

The visual shows the steps that occur from the time IP PrintWay selects output data sets from the JES spool until IP PrintWay transmits the data sets to the target printer or print server and deletes the data sets fro the JES spool. An explanation of each step follows:

1. IP PrintWay selects output data sets from the JES spool according to the JES work-selection parameters defined for the IP PrintWay FSA. For example, IP PrintWay might select all data sets in JES output class J.

   IP PrintWay can select data sets that were allocated on the JES spool by NetSpool or Print Interface, or submitted from TSO or batch applications. The data sets can contain S/390 line data. ASCII text data, or formatted data, such as PCL, PostScript, SAP, or MO:DCA-P (AFP) data.

2. IP PrintWay runs as a functional subsystem application (FSA) of JES2 or JES3. Several IP PrintWay FSAs can run in one functional subsystem address space (FSS) to handle a high volume of data; however, one PrintWay FSA can transmit data sets to multiple printers or print servers.

3. IP PrintWay uses information in the printer definition in the Printer Inventory to process data sets, select the TCP/IP transmission protocol (LPR, IPP, or direct sockets), and obtain the address of the target printer. IP PrintWay can also use the IP address of a target printer specified directly on the OUTPUT JCL statement.

IP PrintWay recognizes data sets allocated on the JES spool by Print Interface and does not convert data from ASCII to EBCDIC or format the data; this is because Print Interface has already converted data to ASCII if necessary. For other data sets, IP PrintWay can convert data from EBCDIC to ASCII, can add a header to each page, and can format data using the carriage-control characters in S/390 line data, an FCB, or pagination attributes specified in the printer definition.

4. IP PrintWay maintains a transmission queue to keep track of data sets being processed. This transmission queue contains the status of each transmission, routing information, and so on. Using Infoprint Server ISPF panels, the system operator can monitor the status of transmissions, reroute data sets to another print queue or port, and change the transmission options.
5. IP PrintWay can use the LPR to LPD, IPP, or direct-socket TCP/IP protocol to transmit data sets to remote printers or print servers. IP PrintWay also transmits LPD options and IPP job attributes to the target LPDs and IPP servers. For example, IP PrintWay can transmit information for the LPD to print on a separator page.

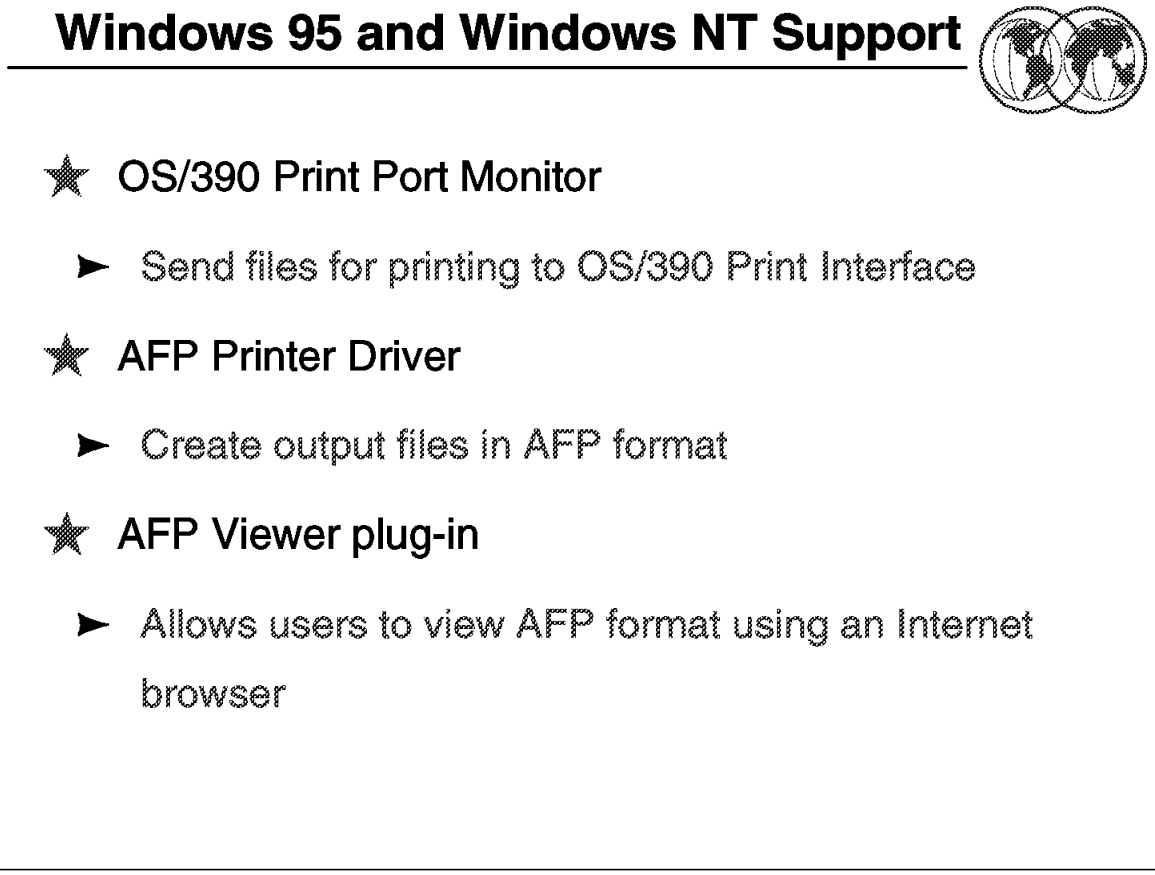


Figure 190. Windows 95 and Windows NT support

5.2.5 Windows 95 and Windows NT support

A job submitter can print documents from any Windows application using:

- Standard methods of print submission available with Windows applications
- A Windows (NT or 95) client provided with an IBM OS/390 Print Server

The job submitter's destination printer can be defined in the OS/390 Printer Inventory.

The Windows client passes job and document attributes to the OS/390 Print Interface. The job name, owner, and the requested printer name are passed to the server.

Note: The OS/390 Print Interface does not return error messages or other job process notifications to these clients. In addition, Windows clients cannot query the status of print requests or cancel a print request.

Note: The OS/390 Print Server Port Monitor must be installed.

5.2.5.1 Windows clients

The Infoprint Server Windows clients are as follows:

- AFP Printer Driver for Windows

  The AFP Printer Driver creates output files in AFP format, so that users can print documents to IBM AFP printers. The AFP Printer Driver can create output files containing documents, overlays, or page segments. It can also create inline form definitions for printing documents with special options, such as printing on both sides of the paper.

- AFP Viewer Plug-in for Windows

  The AFP Viewer plug-in lets users view documents in AFP format, for example documents downloaded from the OS/390 system or documents on the Web. The AFP Viewer plug-in also lets users print AFP documents to AFP as well as non-AFP printers.

- OS/390 Printer Port Monitor for Windows

  The OS/390 Printer Port Monitor lets users print documents using standard print-submission methods from any Windows application that supports printing. After the OS/390 Printer Port Monitor is installed and configured on the Windows system, the Port Monitor automatically sends documents to the Print Interface component of Infoprint Server.

5.2.5.2 Requirements for using the IBM-supplied clients

The IBM-supplied AFP clients require Windows 95 or Windows NT (Version 3.51 or later).

The OS/390 Print Server Port Monitor requires that Microsoft TCP/IP protocol be configured and operational.

The IBM AFP Plug-In Viewer requires Netscape Navigator (Version 3.01 or later) or Microsoft Internet Explorer (Version 3.01, Level 4.70.1215 or later).

The OS/390 Print Server Clients for Windows 95 and Windows NT can be downloaded from the OS/390 host. They are located and stored on OS/390 as self-extracting files in directory:

```
/usr/lpp/Printsrv/win/en_US/
```

Alternatively, you can download the latest software from the Web using the following site:

```
http://www.ibm.com/printers
```

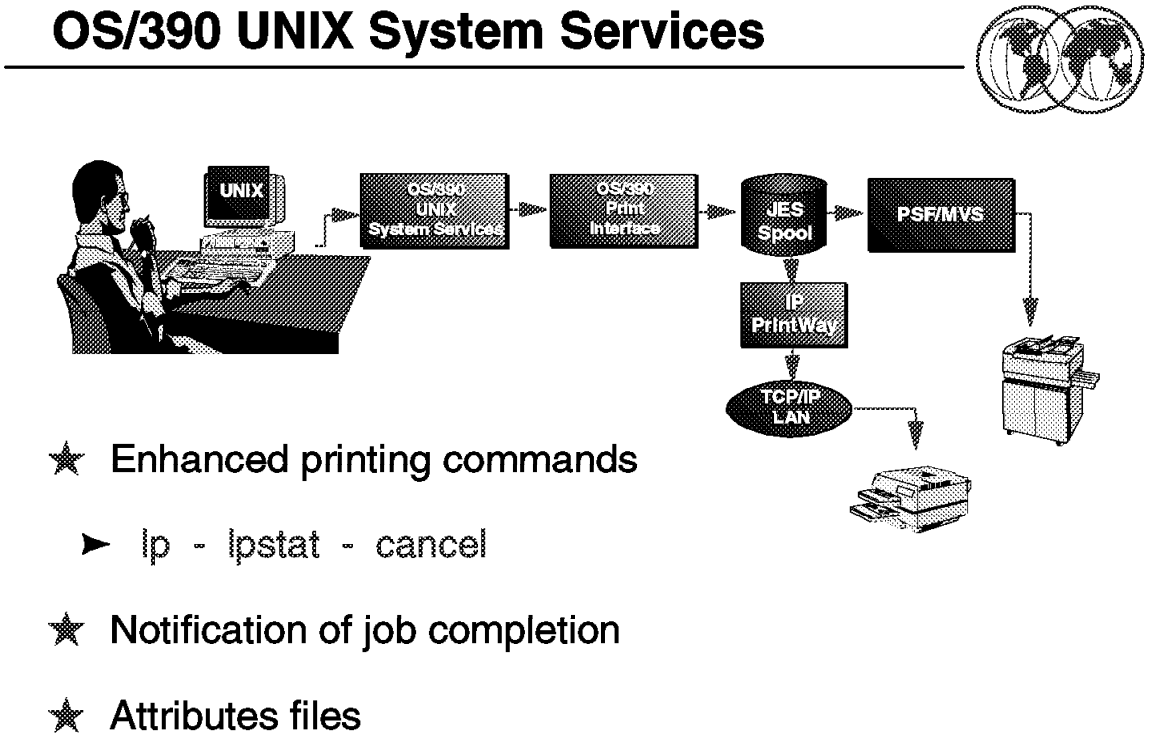


Figure 191. OS/390 UNIX System Services

5.2.6 OS/390 UNIX System Services

The Print Interface provides printing support for users and application programs in the OS/390 UNIX System Services environment. Users and applications can print to OS/390-controlled printers, including these printers:

- JES-controlled printers
- JES- controlled Advanced Function Presentation (AFP) printers that are attached to OS/390 and that use PSF
- LAN-attached ASCII printers that use IP PrintWay

When printing from UNIX System Services, you can print the following types of data:

- Hierarchical File System (HFS) files
- Partitioned data sets
- Sequential data sets

A job submitter can use the following commands:

lp Print documents

lpstat Query the status of print requests

cancel Cancel print requests

Using these commands, you can print jobs on any printer that your system administrator has defined to Infoprint Server. These printing commands provide enhanced function over the commands of the same

name that are described in OS/390 UNIX System Services Command Reference. For example, when printing on AFP printers, you can specify options such as duplexing or a special overlay. You can also display the status of your print request, and you can cancel a print request. These printing commands adhere to the UNIX standards in XPG4.2. You do not need to change your UNIX applications when you port them to OS/390.

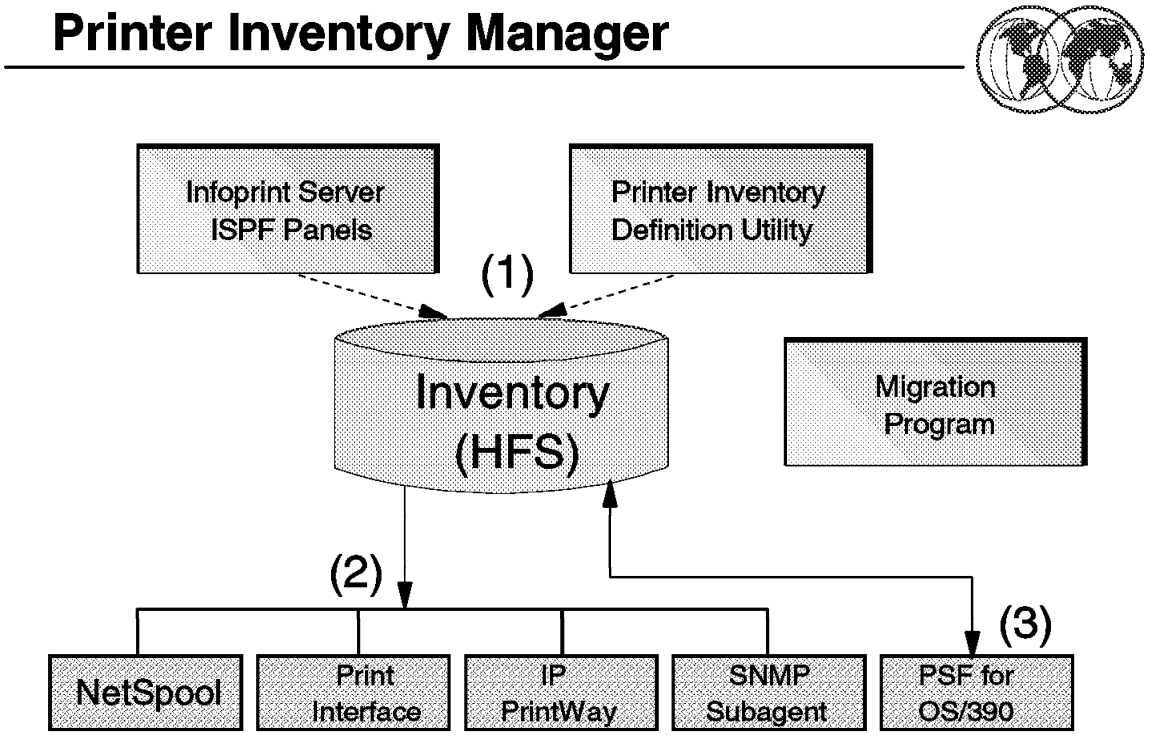


Figure 192. Printer Inventory Manager

5.3 Printer Inventory Manager

The Printer Inventory Manager controls the Printer Inventory, HFS files that contain information about the OS/390 printing environment. The administrator must create and manage information in the Printer Inventory.

Note: The Printer Inventory Manager is new with OS/390 Release 8. OS/390 Releases 5, 6, and 7 with the Print Server have separate ISPF applications to define the printing environment.

The administrator can create the following objects:

- Printer definitions, which contain information about printers to which Print Interface, NetSpool, or IP PrintWay can print
- Printer pool definitions, which contain information about groups of printers to which NetSpool can broadcast data
- FSA definitions, which contain configuration information for IP PrintWay and PSF for OS/390 functional subsystem applications (FSAs)
- FSS definitions, which contain configuration information for IP PrintWay and PSF for OS/390 functional subsystems (FSSs)

The visual shows how the administrator can create definitions in the Printer Inventory and which components of Infoprint Server use the Printer Inventory, as follows:

1. The administrator can use Infoprint Server ISPF panels and the Printer Inventory Definition Utility (PIDU) to create and maintain the Printer Inventory. The PIDU is useful for creating many printer definitions at the same time and for backing up the Printer Inventory.
2. The following Infoprint Server components use information in the Printer Inventory:
   - NetSpool uses information in printer definitions and in printer pool definitions.
   - Print Interface uses information in printer definitions.
   - IP PrintWay uses information in printer definitions. IP PrintWay also can use IP PrintWay configuration information in FSS and FSA definitions.
   - The SNMP subagent uses printer information that PSF for OS/390 stores in the Printer Inventory about PSF printers.
3. PSF for OS/390, although not a component of Infoprint Server, can use configuration information that the administrator specifies in FSS and FSA definitions. PSF for OS/390 can also store printer information in the Printer Inventory for use by the Infoprint Server SNMP subagent. For information about how to customize PSF for OS/390 to use the Printer Inventory, refer to *Print Services Facility for OS/390: Customization*, S544-5622.

5.3.1 Migration program

The Infoprint Server migration program helps the administrator migrate from previous releases of IP PrintWay, NetSpool, and the OS/390 Print Server. The migration program merges printer information currently specified in NetSpool print characteristics data sets, NetSpool tables, NetSpool startup procedures, IP PrintWay routing and options data sets, and the Print Interface printer inventory to create entries (such as printer definitions and printer pool definitions) in the new Infoprint Server Printer Inventory.

The migration program can also move printer information in PSF startup procedures to FSS and FSA definitions in the Printer Inventory.

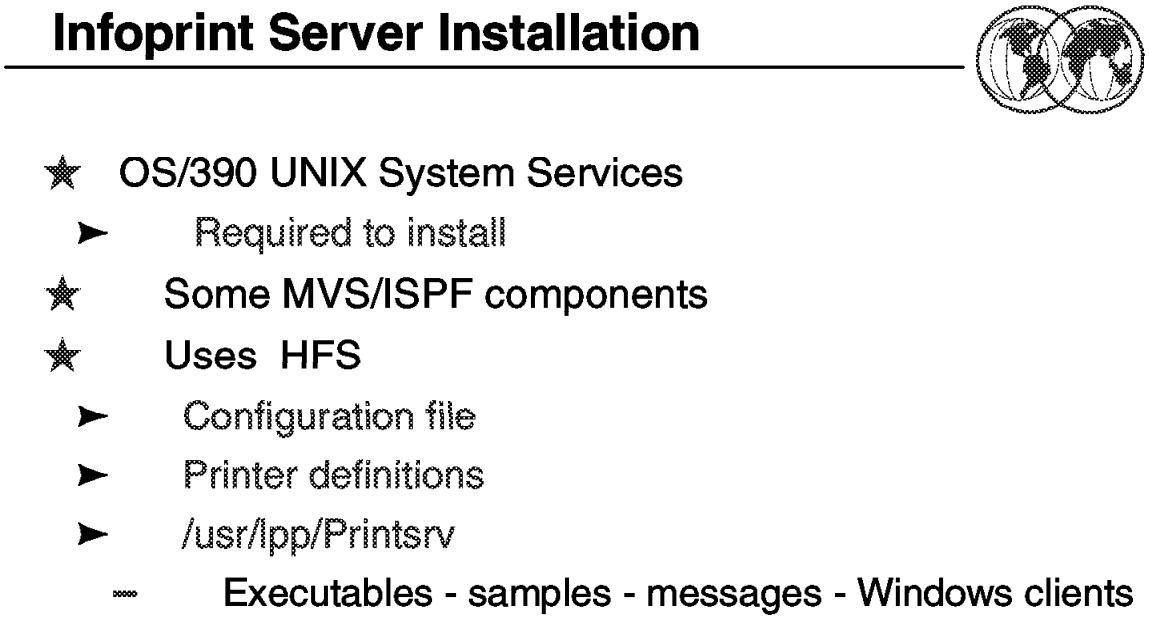


Figure 193. Infoprint Server installation

5.4 Infoprint Server installation

OS/390 UNIX Services must be installed to use the OS/390 Print Server.

Following your system install, a configuration file called *aopd.conf* contains configuration information to be used by the OS/390 Print Interface The aopd.conf configuration file lets you customize the Printer Inventory Manager and other components of Infoprint Server. This file is optional; if the configuration file does not exist or if a statement in the configuration file is omitted, default values are used. The configuration file is stored in an HFS.

The administrator can use Infoprint Server ISPF panels to add, browse, copy, edit, and delete printer definitions and other objects in the Printer Inventory.

During the install of the Infoprint Server, the executables, samples, messages, and Windows clients are installed into the HFS in the directory, /usr/lpp/Printsrv.

Appendix A. Network Management

This appendix includes references from Chapter 1, “Network Management.”

A.1 Major node definitions

This is an example of a local non-SNA major node to define co-ax-attached 3270 terminals connected to a channel-attached 3174-type controller.

```
*/* LIB: SYS1.VTAMLST(EXLOCAL)
*/*
*/* DOC: THIS MEMBER CONTAINS ACF/VTAM DEFINITION STATEMENTS
*/*      FOR 3270-TYPE SCREENS ON A NON-SNA CONTROL UNIT
*/*
EXLOCAL LBUILD
*
EXL001   LOCAL CUADDR=6020,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
EXL002   LOCAL CUADDR=6021,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
EXL003   LOCAL CUADDR=6022,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
EXL004   LOCAL CUADDR=6023,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
EXL005   LOCAL CUADDR=6024,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
EXL006   LOCAL CUADDR=6025,                                          X
               DLOGMOD=D4B32782,                                     X
               TERM=3277,                                            X
               FEATUR2=MODEL2,                                       X
               ISTATUS=ACTIVE,                                       X
               USSTAB=USSPAC
*
```

This example will define six terminals, EXL001-EXL006, on the device address range 6020-6025.

For a complete description of the parameters and their values refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference* SC31-8565.

EXLOCAL EXLOCAL is the name of this major node. The name coded here should be the same as the member name, in this example, SYS1.VTAMLST(**EXLOCAL**). This is the name by which this major node will be known to VTAM.

LBUILD LBUILD specifies that this major node is for LOCAL NON-SNA devices.

EXL001 The LUNAME (Logical Unit NAME) associated with this terminal.

LOCAL Defines this LU as type LOCAL.

CUADDR Specifies the physical device address of this terminal.

DLOGMOD Specifies the Default Logmode associated with this terminal. The logmode is used by VTAM to tailor the 3270 datastream to match the terminal's capabilities. The D4B32782 logmode used in this example will incorporate parameters that give VTAM such information as the number of rows and columns that this terminal can support and whether or not it can handle extended highlighting (colors, reverse video, etc.).

TERM Specifies a specific of terminal device type.

FEATUR2 Specifies some additional features that this non-SNA local device can support.

ISTATUS Specifies whether this terminal is to be activated when the major node is activated.

USSTAB Specifies the table to use when this terminal is not connected to an application. It can include a "Welcome to the VTAM Network" USSMSG10 as well as *shortcut* commands for connecting to other applications. For example, a USS table can have an entry PROD so that instead of entering the command `LOGON APPLID(CICSPROD)` to connect to the CICSPROD application, you can simply type PROD.

This member can be activated by the following VTAM Operator command:

```
V NET,ACT,ID=EXLOCAL
```

A.2 XCA Major Node

The XCA major node is used to define an External Communication Adapter to VTAM. The following example is for an OSA device attached to a token ring network.

```
OSA1     VBUILD TYPE=XCA
*
OSAPORT  PORT  ADAPNO=0,          OSA ADAPTER 0                      X
               CUADDR=D02,        OSA DEVICE ADDRESS                 X
               MEDIUM=RING,       TOKEN RING                         X
               SAPADDR=4,         SERVICE ACCESS POINT               X
               TIMER=30
*
OSAGRP   GROUP ANSWER=ON,         PU DIAL TO VTAM CAPABILITY         X
               CALL=INOUT,        BOTH WAYS SESSION SETUP ALLOWED    X
               DIAL=YES,          SWITCHED CONNECTION                X
               AUTOGEN=(255,L,P), AUTOGEN 8 LINE/PU PAIRS            X
               DYNPU=YES,         PU's DYNAMICALLY ALLOCED           X
               ISTATUS=ACTIVE     INITIAL STATUS ACTIVE
```

This example will define a token ring SNA port on device address D02.

For a complete description of the parameters and their values refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference* SC31-8565.

OSA1 OSA1 is the name of this major node.

VBUILD TYPE=XCA Defines this as an XCA type major node.

OSAPORT The name VTAM will use to refer to the port being defined.

CUADDR The device address of the port.

MEDIUM The type of connection. Ring for token ring.

SAPADDR Service Access Point address for this connection to the LAN. It must be a multiple of 4.

OSAGRP The group name by which VTAM will refer to this set of logical Line and Physical Unit pairs. When subsequent Switched major nodes are defined for token ring-attached devices they will establish a connection via a logical Line and PU pair.

AUTOGEN Instead of coding 255 Line and PU pairs, the autogen parameters specifies that 255 pairs are to be defined, Line names starting with L and PU names starting with P. This will be sufficient for the simultaneous connection of up to 255 downstream LAN-attached PUs.

ISTATUS Specifies that these devices are to be activated on startup.

This member can be activated by the VTAM Operator command:

```
V NET,ACT,ID=OSA1
```

A.3 Switched major node

The Switched major node is used to define switched communication links to VTAM.

```
SWITCH1        VBUILD TYPE=SWNET,                                    X
               MAXGRP=1,           ** NO. OF UNIQUE GROUP NAMES     ** X
               MAXNO=1             ** NO. OF UNIQUE TELEPHONE NOS.  **
*
SWPUA     PU   ADDR=C1,            ** PHYSICAL UNIT ADDRESS         ** X
               IDBLK=017,          ** CONTROLLER                    ** X
               IDNUM=00001,        ** USER 3174 SERIAL NUMBER HERE  ** X
               DISCNT=NO,          ** VTAM DOES NOT HANG UP         ** X
               IRETRY=YES,         ** RE-POLL AFTER TIMEOUT         ** X
               PUTYPE=2,           ** PU TYPE 2                     ** X
               MAXOUT=7,           ** MAXIMUM PIUS = RESPONSE       ** X
               MAXDATA=2042,       ** DATA PLUS HEADER              ** X
               MODETAB=AMODE,      ** LOGON MODE TABLE NAME         ** X
               USSTAB=AUSSTAB,     ** USS TABLE NAME                ** X
               DLOGMOD=SNX32702,   ** LOGON MODE ENTRY 3270 EABs    ** X
               PACING=0,           ** SECONDARY RECEIVES            ** X
               VPACING=0           ** PRIMARY SENDS                 **
SWLUA2    LU   LOCADDR=2
SWLUA3    LU   LOCADDR=3
SWLUA4    LU   LOCADDR=4
SWLUA5    LU   LOCADDR=5
SWLUA6    LU   LOCADDR=6
SWLUA7    LU   LOCADDR=7
```

This example will define a Switched Physical Unit and 6 Logical Units.

For a complete description of the parameters and their values refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference* SC31-8565.

SWITCH1 The name by which VTAM will identify this major node.

VBUILD TYPE=SWNET Defines this as a Switched Major Node.

SWPUA PU The name by which VTAM will identify this PU

IDBLK A 3-digit hexadecimal number that identifies the device type. This value will be provided in the documentation for the actual device. All devices of the same type, such as a 3274, will have the same IDBLK value.

IDNUM A 5-digit hexadecimal value that uniquely identifies this device connection. This value will be provided in the documentation for the actual device. All devices of the same type, such as a 3274, will have the same IDBLK value. When this device attempts to establish a connection with VTAM, the information provided on the setup request will include this IDNUM value. VTAM will then make a one-to-one match between the physical device and this PU definition.

SWLUA2 LU Logical Unit definition

LOCADDR The physical device will be configured to support a number of logical units at given addresses. These addresses will correspond to the LOCADDR addresses in the VTAM definition.

A.4 Sample FTP start procedure

```
//FTPD   PROC PARMS=''
//*********************************************************************
//*                                                                   *
//*         FTP for MVS TCP/IP Version 3 Release 1 Level 0            *
//*                                                                   *
//*********************************************************************
//FTPD   EXEC PGM=FTPD,PARM='POSIX(ON) ALL31(ON)/&PARMS'
//STEPLIB DD DSN=TCPIP.SEZALINK,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*           /etc/ftp.data
//*           SYSFTPD DD statement
//*           jobname.FTP.DATA
//*           SYS1.TCPPARMS(FTPDATA)
//*           tcpip.FTP.DATA
//*
//*       If no FTP.DATA file is found, FTP default values are used.
//*       For information on FTP defaults, see the Customization
//*       and Administration Guide and TCP/IP OE MVS Applications
//*       Feature Guide.
//SYSFTPD DD DISP=SHR,DSN=SYS1.TCPIP.PARMS(FTPDATA)   1
//*
//*      SYSTCPD explicitly identifies which file is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the JCL of
//*      the server.  The file can be any sequential data set,
//*      member of a partitioned data set (PDS), or HFS file.
//SYSTCPD  DD DISP=SHR,DSN=SYS1.TCPIP.PARMS(TCPDATA)  2
```

A.5 Sample OAT

```
/*********************************************************************/
/*      File created 15:16:33 on 04/26/1999                          */
/*********************************************************************/

/*********************************************************************/
/*               Start of OSA Address Table for CHPID 5C             */
/*********************************************************************/
/* All entries below that are preceeded by 's-' indicate that the    */
/* field is settable during Put_OAT processing                       */
/*********************************************************************/
oathdr.1 = IOA_OAT_HDR                /*   Eyecatcher-Do not change */
oathdr.2 = 5C                         /*   CHPID                    */
oathdr.3 = 8                          /* s-Number of entries        */
/*********************************************************************/
/* Start of OAT entry 1                                              */
/*********************************************************************/
oat.1.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.1.2 = All data is valid           /*   Valid data indicator     */
oat.1.3 =                             /*   Partition name           */
oat.1.4 = 0                           /* s-Partition number         */
oat.1.5 = 04                          /* s-Unit address             */
oat.1.6 = 0D04                        /*   Device number            */
oat.1.7 = 5C                          /*   Chpid                    */
oat.1.8 = 0D00                        /*   Control unit number      */
oat.1.9 = configured                  /*   Channel state            */
oat.1.10 = yes                        /*   Device accessible        */
oat.1.11 = 02                         /*   Group size               */
oat.1.12 = passthru                   /* s-Entry type. One of:      */
                                      /*     Passthru               */
                                      /*     SNA                    */
                                      /*     Unassigned             */
oat.1.13 = started and in use         /*   Entry descriptor         */
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
passthru.1.1 = 1                      /* s-Port number              */
passthru.1.2 = no                     /* s-Default LP (yes/no)      */
passthru.1.3 = 195.183.66.11          /* s-home IP address          */
/*********************************************************************/
/* Start of OAT entry 2                                              */
/*********************************************************************/
oat.2.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.2.2 = All data is valid           /*   Valid data indicator     */
oat.2.3 =                             /*   Partition name           */
oat.2.4 = 0                           /* s-Partition number         */
oat.2.5 = 05                          /* s-Unit address             */
oat.2.6 = 0D05                        /*   Device number            */
oat.2.7 = 5C                          /*   Chpid                    */
oat.2.8 = 0D00                        /*   Control unit number      */
oat.2.9 = configured                  /*   Channel state            */
oat.2.10 = yes                        /*   Device accessible        */
oat.2.11 = 02                         /*   Group size               */
oat.2.12 = passthru                   /* s-Entry type. One of:      */
                                      /*     Passthru               */
                                      /*     SNA                    */
                                      /*     Unassigned             */
oat.2.13 = started and in use         /*   Entry descriptor         */
```

```
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
passthru.2.1 = 1                      /* s-Port number               */
passthru.2.2 = no                     /* s-Default LP (yes/no)       */
passthru.2.3 = 195.183.66.11          /* s-home IP address           */
/*********************************************************************/
/* Start of OAT entry 3                                              */
/*********************************************************************/
oat.3.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.3.2 = All data is valid           /*   Valid data indicator      */
oat.3.3 =                             /*   Partition name            */
oat.3.4 = 0                           /* s-Partition number          */
oat.3.5 = 02                          /* s-Unit address              */
oat.3.6 = 0D02                        /*   Device number             */
oat.3.7 = 5C                          /*   Chpid                     */
oat.3.8 = 0D00                        /*   Control unit number       */
oat.3.9 = configured                  /*   Channel state             */
oat.3.10 = yes                        /*   Device accessible         */
oat.3.11 = 01                         /*   Group size                */
oat.3.12 = SNA                        /* s-Entry type. One of:       */
                                      /*     Passthru                */
                                      /*     SNA                     */
                                      /*     Unassigned              */
oat.3.13 = started and in use         /*   Entry descriptor          */
/*********************************************************************/
/* Start of Extended OAT entry                                       */
/*********************************************************************/
sna.3.1 = 00                          /* s-Port number               */
/*********************************************************************/
/* Start of OAT entry 4                                              */
/*********************************************************************/
oat.4.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.4.2 = Channel subsystem valid     /*   Valid data indicator      */
oat.4.3 =                             /*   Partition name            */
oat.4.4 = 0                           /* s-Partition number          */
oat.4.5 = 00                          /* s-Unit address              */
oat.4.6 = 0D00                        /*   Device number             */
oat.4.7 = 5C                          /*   Chpid                     */
oat.4.8 = 0D00                        /*   Control unit number       */
oat.4.9 = configured                  /*   Channel state             */
oat.4.10 = yes                        /*   Device accessible         */
oat.4.11 = N/A                        /*   Group size                */
oat.4.12 = N/A                        /* s-Entry type. One of:       */
                                      /*     Passthru                */
                                      /*     SNA                     */
                                      /*     Unassigned              */
oat.4.13 = N/A                        /*   Entry descriptor          */
/*********************************************************************/
/* Start of OAT entry 5                                              */
/*********************************************************************/
oat.5.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.5.2 = Channel subsystem valid     /*   Valid data indicator      */
oat.5.3 =                             /*   Partition name            */
oat.5.4 = 0                           /* s-Partition number          */
oat.5.5 = 01                          /* s-Unit address              */
oat.5.6 = 0D01                        /*   Device number             */
oat.5.7 = 5C                          /*   Chpid                     */
oat.5.8 = 0D00                        /*   Control unit number       */
```

```
oat.5.9 = configured                  /*   Channel state           */
oat.5.10 = yes                        /*   Device accessible       */
oat.5.11 = N/A                        /*   Group size              */
oat.5.12 = N/A                        /* s-Entry type. One of:     */
                                      /*     Passthru              */
                                      /*     SNA                   */
                                      /*     Unassigned            */
oat.5.13 = N/A                        /*   Entry descriptor        */
/*************************************************************************/
/* Start of OAT entry 6                                                  */
/*************************************************************************/
oat.6.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.6.2 = Channel subsystem valid     /*   Valid data indicator    */
oat.6.3 =                             /*   Partition name          */
oat.6.4 = 0                           /* s-Partition number        */
oat.6.5 = 03                          /* s-Unit address            */
oat.6.6 = 0D03                        /*   Device number           */
oat.6.7 = 5C                          /*   Chpid                   */
oat.6.8 = 0D00                        /*   Control unit number     */
oat.6.9 = configured                  /*   Channel state           */
oat.6.10 = yes                        /*   Device accessible       */
oat.6.11 = N/A                        /*   Group size              */
oat.6.12 = N/A                        /* s-Entry type. One of:     */
                                      /*     Passthru              */
                                      /*     SNA                   */
                                      /*     Unassigned            */
oat.6.13 = N/A                        /*   Entry descriptor        */
/*************************************************************************/
/* Start of OAT entry 7                                                  */
/*************************************************************************/
oat.7.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.7.2 = Channel subsystem valid     /*   Valid data indicator    */
oat.7.3 =                             /*   Partition name          */
oat.7.4 = 0                           /* s-Partition number        */
oat.7.5 = 06                          /* s-Unit address            */
oat.7.6 = 0D06                        /*   Device number           */
oat.7.7 = 5C                          /*   Chpid                   */
oat.7.8 = 0D00                        /*   Control unit number     */
oat.7.9 = configured                  /*   Channel state           */
oat.7.10 = yes                        /*   Device accessible       */
oat.7.11 = N/A                        /*   Group size              */
oat.7.12 = N/A                        /* s-Entry type. One of:     */
                                      /*     Passthru              */
                                      /*     SNA                   */
                                      /*     Unassigned            */
oat.7.13 = N/A                        /*   Entry descriptor        */
/*************************************************************************/
/* Start of OAT entry 8                                                  */
/*************************************************************************/
oat.8.1 = IOA_OATENTRY                /*   Eyecatcher- Do not delete*/
oat.8.2 = Channel subsystem valid     /*   Valid data indicator    */
oat.8.3 =                             /*   Partition name          */
oat.8.4 = 0                           /* s-Partition number        */
oat.8.5 = 07                          /* s-Unit address            */
oat.8.6 = 0D07                        /*   Device number           */
oat.8.7 = 5C                          /*   Chpid                   */
oat.8.8 = 0D00                        /*   Control unit number     */
oat.8.9 = configured                  /*   Channel state           */
oat.8.10 = yes                        /*   Device accessible       */
```

```
oat.8.11 = N/A                          /*   Group size              */
oat.8.12 = N/A                          /* s-Entry type. One of:     */
                                        /*     Passthru              */
                                        /*     SNA                   */
                                        /*     Unassigned            */
oat.8.13 = N/A                          /*   Entry descriptor        */
/*************************************************************************
/******************    End of OAT entries    ************************
/*************************************************************************

/*-----------------------------------------------------------------------*/
/* Examples of Extended OAT entries start here                           */
/*************************************************************************/
/* Passthru extended entry example                                       */
/* --change IP address to match TCPIP profile                            */
/*************************************************************************/
/* passthru.n.1 = PORT_NUMBER          /* s-Port number (1 hex digit)*/
/* passthru.n.2 = yes or no            /* s-Default LP (yes/no)      */
/* passthru.n.3 = IP address           /* s-home IP address (w.x.y.z)*/
/*************************************************************************/
/* SNA extended entry WITHOUT Network managment example                  */
/*************************************************************************/
/* sna.n.1 = PORT_NUMBER               /* s-Port number (1 hex digit)*/
/*************************************************************************/
/* SNA extended entry WITH Network managment example                     */
/* --change VTAM IDNUM to match value on XCA node definition             */
/*************************************************************************/
/* sna.n.1 = FF                        /* s-Port number (must be FF) */
/* sna.n.2 = VTAM IDNUM                /* s-VTAM IDNUM (5 hex chars) */
/*************************************************************************
/*************    END of Extended OAT entry examples    ***************
/*************************************************************************
```

Appendix B. Special Notices

This publication is intended to help new system programmers who need to understand S/390 and the OS/390 operating system. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 Versions. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 Version 2 Release 8, Program Number 5647-A01 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ACF/VTAM
AFP
CICS
CICS/MVS
Advanced Function Printing
AnyNet
CICS/ESA
DB2

DFSMS
DFSMSdfp
DFSMShsm
DFSORT
FICON
IMS
Intelligent Printer Data Stream
IP PrintWay
MVS (block letters)
NetSpool
OpenEdition
Parallel Sysplex
RACF
RMF
S/390 Parallel Enterprise Server
VTAM
DFSMS/MVS
DFSMSdss
DFSMSrmm
ESCON
IBM
InfoPrint
IPDS
Language Environment
MVS/DFP
NetView
OS/390
PrintWay
RAMAC
S/390
Sysplex Timer

The following terms are trademarks of other companies:

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjobenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

SET, SET Secure Electronic Transaction, and the SET logo are trademarks owned by Secure Electronic Transaction LLC.

UNIX is a registered trademark in the United States and other countries licensed exclusively through the Open Group.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these ITSO publications see “How to get IBM Redbooks” on page 355.

- *OS/390 Release 5 Implementation*, SG24-5151
- *OS/390 Release 4 Implementation*, SG24-2089
- *OS/390 Release 3 Implementation*, SG24-2067
- *OS/390 Release 2 Implementation*, SG24-4834
- cit.OS/390 Security Server 1999 Updates Technical Presentation Guide, SG24-5627
- *Security in OS/390-based TCP/IP Networks*, SG24-5383
- *Hierarchical File System Usage Guide*, SG24-5482
- *Enhanced Catalog Sharing and Management*, SG245594
- *Integrated Catalog Facility Backup and Recovery*, SG24-5644
- *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customiztion*, SG24-5178
- *IBM S/390 FICON Implementation Guide*, SG24-5169
- *Exploiting S/390 Hardware Cryptography with Trusted Key Entry*, SG24-5455
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Introduction to Storage Area Network SAN*, SG2-45470
- *TCP/IP in a Sysplex*, SG24-5235
- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *ADSM Server-to-Server Implementation and Operation*, SG24-5244
- *Stay Cool on OS/390: Installing Firewall Technology*, SG24-2046
- *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268
- *TCP/IP OpenEdition Implementation Guide*, SG24-2141
- *IMS/ESA Version 5 Performance Guide*, SG24-4637
- *Parallel Sysplex Configuration: Overview*, SG24-2075
- *Parallel Sysplex Configuation: Cookbook*, SG24-2076
- *Parallel Sysplex Config.: Connectivity*, SG24-2077
- *DFSMS Optimizer Presentation Guide Update*, SG24-4477
- *MVS Parallel Sysplex Capacity Planning*, SG24-4680

- *Getting the Most Out of a Parallel Sysplex*, SG24-2073
- *OS/390 eNetwork Communication Server TCP/IP Implementation Guide Volume 2*, SG24-5228

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 Initialization and Tuning Guide*, SC28-1751
- *OS/390 Initialization and Tuning Reference*, SC28-1752
- *OS/390 Introduction and Release Guide*, GC28-1725
- *OS/390 MVS JCL User's Guide*, SC28-1758
- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS System Diagnosis: Tools and Service Aids*, LY28-1085, (available to IBM licensed customers only)
- *Interactive System Productivity Facility Getting Started*, SC34-4440
- *OS/390 Security Server (RACF) System Programmer's Guide*, SC28-1913
- *OS/390 TSO/E Customization*, SC28-1965
- *OS/390 TSO/E Primer*, GC28-1967
- *OS/390 TSO/E User's Guide*, SC28-1968
- *OS/390 SMP/E Reference*, SC28-1806
- *OS/390 SMP/E User's Guide*, SC28-1740
- *OS/390 SMP/E Commands*, SC28-1805
- *Standard Packaging Rules for MVS-Based Products*, SC23-3695
- *OS/390 MVS System Commands*, GC28-1781
- *OS/390 MVS IPCS Commands*, GC28-1754
- *OS/390 MVS IPCS User's Guide*, GC28-1756
- *DFSMS/MVS Using Data Sets*, SC26-4922
- *OS/390 Planning for Installation*, GC28-1726
- *OS/390 MVS System Data Sets Definition*, GC28-1782

- *ICKDSF R16 Refresh, User's Guide*, GC35-0033
- *OS/390 MVS System Management Facilities (SMF)*, GC28-1783
- *EREP V3R5 Reference*, GC35-0152
- *OS/390 JES2 Commands*, GC28-1790
- *OS/390 Hardware Configuration Definition User's Guide*, SC28-1848
- *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929
- *IBM ServerPac for OS/390 Using the Installation Dialog*, SC28-1244
- *OS/390 Hardware Configuration Definition Planning*, GC28-1750
- *OS/390 MVS Using the Subsystem Interface*, SC28-1502
- *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914
- *DFSMS/MVS Version 1 Release4: Access Method Services for Integrated Catalog Facility*, SC26-4906
- *DFSMS/MVS: DFSMShsm Implementation and Customization Guide*, SH21-1078
- *DFSMS/MVS Access Method Services for ICF Catalogs*, SC26-4500
- *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920
- *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*, SC31-8565
- *OS/390 eNetwork Communications Server SNA Resource Definition Samples*, SC31-8566
- *OS/390 eNetwork Communications Server: SNA Operation*, SC31-8567
- *OS/390 V2R7.0 eNetwork CS IP Configuration*, SC31-8513
- *eNetwork Communications Server: IP User's Guide* GC31-8514
- *OS/390 UNIX System Services Planning*, SC28-1890
- *OS/390 TCP/IP OpenEdition: Configuration Guide* SC31-8304
- *OS/390 Open Systems Adapter Support Facility User's Guide*, SC28-1855.
- *OS/390 V2R6.0 MVS Planning: APPC/MVS Management*, GC28-1807
- *Print Services Facility for OS/390: Customization*, S544-5622
- *DFSMS/MVS Planning for Installation*, SC26-4919
- *DFSMS/MVS Implementing System-Managed Storage*, SC26-3123
- *DFSMS/MVS Remote Copy Administrator's Guide and Reference*, SC35-0169
- *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913
- *DFSMS/MVS DFSMSdfp Diagnosis Guide*, SY27-9605
- *DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921
- *DFSMS/MVS Using Magnetic Tapes*, SC26-4923
- *DFSMS/MVS Utilities*, SC26-4926
- *Service Level Reporter User's Guide: Reporting*, SH19-6530
- *DFSMS/MVS Object Access Method Application Programmer's Reference*, SC26-4917
- *DFSMS/MVS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC26-4918
- *DFSORT Installation and Customization*, SC26-7041
- *DFSORT Getting Started with DFSORT R14*, SC26-4109
- *DFSMS/MVS Network File System Customization and Operation*, SC26-7029

- *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930
- *DFSMShsm Storage Administration Guide*, SH21-1076
- *DFSMShsm Storage Administration Reference*, SH21-1075
- *DFSMS/MVS Network File System User's Guide*, SC26-7028
- *DFSMS/MVS DFSMSrmm Guide and Reference*, SC26-4931
- *DFSMS/MVS DFSMSrmm Implementation and Customization Guide*, SC26-4932
- *MVS/ESA Storage Management Library Managing Data*, SC26-3124
- *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125
- *MVS/ESA Storage Management Library Leading a Storage Administration Group*, SC26-3126.
- *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911
- *ADSTAR Distributed Storage Manager for MVS Administrator's Guide*, GC35-0277
- *OS/390 MVS Programming: Assembler Services Guide*, GC28-1762
- *OS/390 MVS Programming: Resource Recovery*, GC28-1739
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779
- *OS/390 MVS Sysplex Services Guide*, GC28-1771
- *OS/390 Parallel Sysplex Systems Management*, GC28-1861
- *OS/390 MVS Systems Codes*, GC28-1780
- *OS/390 MVS System Messages Volume 1*, GC28-1784
- *OS/390 MVS System Messages Volume 2*, GC28-1785
- *OS/390 MVS System Messages Volume 3*, GC28-1786
- *OS/390 MVS System Messages Volume 4*, GC28-1787
- *OS/390 MVS System Messages Volume 5*, GC28-1788
- *OS/390 MVS Installation Exits*, SC28-1753
- *OS/390 MVS Diagnosis Reference*, SY28-1084
- *CICS User's Handbook*, SX33-1188
- *CICS Diagnosis Guide*, LX33-6093
- *MQSeries for MVS/ESA Messages and Codes*, GC33-0819

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbook fax order form to:

In United States:	e-mail address: usib6fpl@ibmmail.com
Outside North America:	Contact information is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

abend. Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task executing.

ACB. Access method control block.

access. A specific type of interaction between a subject and an object that results in the flow of information from one to the other.

access authority. An authority that relates to a request for a type of access to protected resources. In RACF, the access authorities are NONE, READ, UPDATE, ALTER, and EXECUTE.

access list. A list within a profile of all authorized users and their access authorities.

access method control block (ACB).. A control block that links an application program to VTAM.

ACDS. Active control data set.

ACF/VTAM. An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. VTAM runs under MVS (OS/VS1 and OS/VS2), VSE, and VM/SP and supports direct control application programs and subsystems such as VSE/POWER.

ACIF. (1) AFP conversion and indexing facility. (2) A PSF utility program that converts a print file into AFP, MO:DCA-P, creates an index file for input data, and collects resources used by an AFP document into separate file.

action message retention facility (AMRF). A facility that, when active, retains all action messages except those specified by the installation in the MPFLSTxx member in effect.

action message sequence number. A decimal number assigned to action messages.

Advanced Function Presentation (AFP). A set of licensed programs, together with user applications, that use the all-points-addressable concept to print on presentation devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

Advanced Program-to-Program Communications (APPC). A set of inter-program communication services that support cooperative transaction processing in a SNA network.

AFP. Advanced Function Presentation.

AFP Printer Driver for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and creates output in AFP format, for printing on AFP printers.

AFP Viewer plug-in for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and allows you to view files in AFP format.

AIX operating system. IBM′s implementation of the UNIX operating system. The RS/6000 system, among others, runs the AIX operating system.

allocate. To assign a resource for use in performing a specific task.

alphanumeric character. A letter or a number.

amode. Addressing mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. AMODE states the addressing mode that is expected to be in effect when the program is entered.

AMRF. action message retention facility

AOR. Application-owning region

APPC. Advanced Program-to-Program Communications

APPN. Advanced Peer-to-Peer Networking.

ASCII (American Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bit including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

audit. To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data accuracy.

authority. The right to access objects, resources, or functions.

authorization checking. The action of determining whether a user is permitted access to a RACF-protected resource.

Authorized Program Analysis Report (APAR). A request for correction of problem caused by a defect in a current unaltered release of a program.

authorized program facility (APF). A facility that permits identification of programs authorized to use restricted functions.

automated operations. Automated procedures to replace or simplify actions of operators in both systems and network operations.

AVR. Automatic volume recognition.

B

banner page. A page printed before the data set is printed.

basic mode. A central processor mode that does not use logical partitioning. Contrast with logically partitioned (LPAR) mode.

batch message processing (BMP) program. An IMS batch processing program that has access to online databases and message queues. BMPs run online, but like programs in a batch environment, they are started with job control language (JCL).

batch-oriented BMP program. A BMP program that has access to online databases and message queues while performing batch-type processing. A batch-oriented BMP does not access the IMS message queues for input or output. It can access online databases, GSAM databases, and MVS files for both input and output.

BMP. Batch message processing (BMP) program.

broadcast. (1) Transmission of the same data to all destinations. (2) Simultaneous transmission of data to more than one destination.

binary data. (1) Any data not intended for direct human reading. Binary data may contain unprintable characters, outside the range of text characters. (2) A type of data consisting of numeric values stored in bit patterns of 0s and 1s. Binary data can cause a large number to be placed in a smaller space of storage.

BIND. In SNA, a request to activate a session between two logical units (LUs).

buffer. A portion of storage used to hold input or output data temporarily.

buffered device. A device where the data is written to a hardware buffer in the device before it is placed on the paper (for example, IBM 3820).

burst. To separate continuous-forms paper into single sheets.

C

cache structure. A coupling facility structure that enables high-performance sharing of cached data by multisystem applications in a sysplex. Applications can use a cache structure to implement several different types of caching systems, including a store-through or a store-in cache.

carriage control character. An optional character in an input data record that specifies a write, space, or skip operation.

carriage return (CR). (1) A keystroke generally indicating the end of a command line. (2) In text data, the action that indicates to continue printing at the left margin of the next line. (3) A character that will cause printing to start at the beginning of the same physical line in which the carriage return occurred.

CART. Command and response token.

case-sensitive. Pertaining to the ability to distinguish between uppercase and lowercase letters.

catalog. (1) A directory of files and libraries, with reference to their locations. (2) To enter information about a file or a library into a (3) The collection of all data set indexes that are used by the control program to locate a volume containing a specific data set.

CBPDO. Custom Built Product Delivery Offering.

CEC. Synonym for central processor complex (CPC).

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

central processor complex (CPC). A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

CFRM. Coupling facility resource management.

channel-to-channel (CTC). Refers to the communication (transfer of data between programs on opposite sides of a channel-to-channel adapter (CTCA

channel-to-channel adapter (CTCA). An input/output device that is used a program in one system to communicate with a program in another system.

checkpoint. (1) A place in a routine where a check, or a recording of data for restart purposes, is performed. (2) A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.

checkpoint write. Any write to the checkpoint data set. A general term for the primary, intermediate, and final writes that update any checkpoint data set.

CICS. Customer Information Control System.

CICSplex. A group of connected CICS regions.

CICSPlex SM. CICSPlex System Manager

client. A functional unit that receives shared services from a server. See also client-server.

client-server. In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

CMOS. Complementary metal-oxide semiconductor.

CNGRPxx. The SYS1.PARMLIB member that defines console groups for the system or sysplex.

code page. (1) A table showing codes assigned to character sets. (2) An assignment of graphic characters and control function meanings to all code points. (3) Arrays of code points representing characters that establish ordinal sequence (numeric order) of characters. (4) A particular assignment of hexadecimal identifiers to graphic elements.

code point. A 1-byte code representing one of 256 potential characters.

coexistence. Two or more systems at different levels (for example, software, service or operational levels) that share resources. Coexistence includes the ability of a system to respond in the following ways to a new function that was introduced on another system with which it shares resources: ignore a new function, terminate gracefully, support a new function.

command and response token (CART). A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

command prefix facility (CPF). An MVS facility that allows you to define and control subsystem and other command prefixes for use in a sysplex.

COMMDS. Communications data set.

complementary metal-oxide semiconductor (CMOS). A technology that combines the electrical properties of positive and negative voltage requirements to use considerably less power than other types of semiconductors.

connection. In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In Internet communications, a connection extends from a TCP application on one syste system to a TCP application on another system.

console. That part of a computer used for communication between the operator or user and the computer.

console group. In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

CONSOLxx. The SYS1.PARMLIB member used to define message handling, command processing, and MCS consoles.

control unit. Synonymous with device control unit.

conversation. A logical connection between two programs over an LU type 6.2 session that allows them to communicate with each other while processing a transaction.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

copy group. One or more copies of a page of paper. Each copy can have modifications, such as text suppression, page position, forms flash, and overlays.

couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. See also sysplex couple data set.

coupling facility. A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

coupling facility channel.. A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

coupling services. In a sysplex, the functions of XCF that transfer data and status between members of a group residing on one or more MVS systems in the sysplex.

CP. Central processor.

CPC. Central processor complex.

CPF. Command prefix facility.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

cryptography. The transformation of data to conceal its meaning.

cryptographic key. A parameter that determines cryptographic transformations between plaintext and ciphertext.

CTC. Channel-to-channel.

Customer Information Control System (CICS). An IBM licensed program tha that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

D

DAE. Dump analysis and elimination.

daemon. A program that runs unattended to perform a standard service.

DASD. Direct access storage device.

data definition name. The name of a data definition (DD) statement, which corresponds to a data control block that contains the same name. Abbreviated as ddname.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data integrity. The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set label. (1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

data set separator pages. Those pages of printed output that delimit data sets.

data sharing. The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data while maintaining data integrity.

data stream. (1) All information (data and control commands) sent over a data link usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

DBCS. Double-byte character set.

DBCTL. IMS Database Control.

DBRC. Database Recovery Control.

DB2. DATABASE 2 for MVS/ESA.

DB2 data sharing group. A collection of one or more concurrent DB2 subsystems that directly access and change the same data while maintaining data integrity.

DB2 PM. DB2 Performance Monitor.

deallocate. To release a resource that is assigned to a specific task.

default. A value, attribute, or option that is assumed when no alternative is specified by the user.

destination node. The node that provides application services to an authorized external user.

device control unit. A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices or terminals.

device number. The unique number assigned to an external device.

device type. The general name for a kind of device; for example, 3330.

DFSMS. Data Facility Storage Management Subsystem.

direct access storage device (DASD). A device in which the access time effectively independent of the location of the data.

directory. (1) A type of file containing the names and controlling information for other files or other directories. Directories can also contain subdirectories, which can contain subdirectories of their own. (2) A file that contains directory entries. No two directory entries in the same directory can have the same name. (POSIX.1). (3) A file that points to files and to other directories. (4) An index used by a control program to locate blocks of data that are stored in separate areas of a data set in direct access storage.

display console. In MVS, an MCS console whose input/output function you can control.

DLL filter. A filter that provides one or more of these functions in a dynamic load library - init(), prolog(), process(), epilog(), and term(). See cfilter.h and cfilter.c in the /usr/lpp/Printsrv/samples/ directory for more information. See also filter. Contrast with DLL filter.

DOM. An MVS macro that removes outstanding WTORs or action messages that have been queued to a console end-of-tape-marker. A marker on a

magnetic tape used to indicate the end of the permissible recording area, for example, a photo-reflective strip a transparent section of tape, or a particular bit pattern.

dotted decimal notation. The syntactical representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. It is used to represent IP addresses.

double-byte character set (DBCS). A set of characters in which each character is represented by a two-bytes code. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

drain. Allowing a printer to complete its current work before stopping the device.

E

entry area. In MVS, the part of a console screen where operators can enter commands or command responses.

EMIF. ESCON Multiple Image Facility.

Enterprise Systems Connection (ESCON). A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

EPDM. IBM SystemView Enterprise Performance Data Manager/MVS.

ESCD. ESCON Director.

ESCM. ESCON Manager. The licensed program System Automation for OS/390 includes all of the function previosuly provided by ESCM.

ESCON. Enterprise Systems Connection.

ETR. External Time Reference. See also Sysplex Timer.

extended MCS console. In MVS, a console other than an MCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

F

FMID. Function modification identifier. The IBM release-specific product identifier such as HJE6610 for OS/390 Release 1 JES2.

FOR. File-owning region.

frame. For a System/390 microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

FSS. functional subsystem. An address space uniquely identified as performing a specific function related to the JES. An example of an FSS is the program Print Services Facility that operates the 3800 Model 3 an 38xx printers.

functional subsystem (FSS). An address space uniquely identified as performing a specific function related to the JES.

functional subsystem application (FSA). The functional application program managed by the functional subsystem.

functional subsystem interface (FSI). The interface through which JES2 JES3 communicate with the functional subsystem.

G

gateway node. A node that is an interface between networks.

generalized trace facility (GTF). Like system trace, gathers information used to determine and diagnose problems that occur during system operation. Unlike system trace, however, GTF can be tailored to record very specific system and user program events.

global access checking. The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources.

global resource serialization. A function that provides an MVS serialization mechanism for resources (typically data sets) across multiple MVS images.

global resource serialization complex. One or more MVS systems that use global resource serialization to serialize access to shared resources (such as data sets on shared DASD volumes).

group. A collection of RACF users who can share access authorities for protected resources.

GTF. Generalized trace facility.

H

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

hardware. Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

hardware configuration dialog. In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

Hardware Management Console. A console used to monitor and control hardware such as the System/390 microprocessors.

HCD. Hardware Configuration Definition.

highly parallel. Refers to multiple systems operating in parallel, each of which can have multiple processors. See also n-way.

I

ICMF. Integrated Coupling Migration Facility.

IMS. Information Management System.

IMS DB. Information Management System Database Manager.

IMS DB data sharing group. A collection of one or more concurrent IMS DB subsystems that directly access and change the same data while maintaining data integrity.

IMS TM. Information Management System Transaction Manager.

initial program load (IPL). The initialization procedure that causes an operating system to begin operation.

instruction line. In MVS, the part of the console screen that contains messages about console control and input errors.

internal reader. A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

IOCDS. Input/output configuration data set.

IOCP. Input/output configuration program.

IODF. Input/output definition file.

IPL. Initial program load.

IRLM. Internal resource lock manager.

ISPF. Interactive System Productivity Facility.

J

JES common coupling services. A set of macro-driven services that provide the communication interface between JES members of a sysplex. Synonymous with JES XCF.

JESXCF. JES cross-system coupling services. The MVS component, common to both JES2 and JES3, that provides the cross-system coupling services to either JES2 multi-access spool members or JES3 complex members, respectively.

JES2. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

JES2 multi-access spool configuration. A multiple MVS system environment that consists of two or more JES2 processors sharing the same job queue and spool

JES3. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In complexes that have several loosely-coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

JES3 complex. A multiple MVS system environment that allows JES3 subsystem consoles and MCS consoles with a logical association to JES3 to receive messages and send commands across systems.

job entry subsystem (JES). A system facility for spooling, job queuing, and managing the scheduler work area.

job separator page data area (JSPA). A data area that contains job-level information for a data set. This information is used to generate job header, job trailer or data set header pages. The JSPA can be used by an installation-defined JES2 exit routine to duplicate the information currently in the JES2 separator page exit routine.

job separator pages. Those pages of printed output that delimit jobs.

K

keyword. A part of a command operand or SYS1.PARMLIB statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

L

LIC. Licensed Internal Code.

list structure. A coupling facility structure that enables multisystem applications in a sysplex to share information organized as a set of lists or queues. A list structure consists of a set of lists and an optional lock table, which can be used for serializing resources in the list structure. Each list consists of a queue of list entries.

lock structure. A coupling facility structure that enables applications in a sysplex to implement customized locking protocols for serialization of application-defined resources. The lock structure supports shared, exclusive, and application-defined lock states, as well as generalized contention management and recovery protocols.

logical partition (LP). A subset of the processor hardware that is defined to support an operating system. See also logically partitioned (LPAR) mode.

logically partitioned (LPAR) mode. A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, central storage, expanded storage, and channel paths) among logical partitions. Contrast with basic mode.

logical unit (LU). In SNA, a port through which an end user accesses th SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs).

logical unit type 6.2. The SNA logical unit type that supports general communication between programs in a cooperative processing environment.

loosely coupled. A multisystem structure that requires a low degree of interaction and cooperation between multiple MVS images to process a workload. See also tightly coupled.

LP. Logical partition.

LPAR. Logically partitioned (mode).

M

MAS. Multi-access spool.

master console. In an MVS system or sysplex, the main console used for communication between the operator and the system from which all MVS commands can be entered. The first active console with AUTH(MASTER) defined becomes the master console in a system or sysplex.

master console authority. In a system or sysplex, a console defined with AUTH(MASTER) other than the master console from which all MVS commands can be entered.

master trace. A centralized data tracing facility of the master scheduler, used in servicing the message processing portions of MVS.

MCS. Multiple console support.

MCS console. A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

member. A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

message processing facility (MPF). A facility used to control message retention, suppression, and presentation.

message queue. A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message text. The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

microprocessor. A processor implemented on one or a small number of chips.

mixed complex. A global resource serialization complex in which one or more of the systems in the global resource serialization complex are not part of a multisystem sysplex.

MP. Multiprocessor.

MPF. Message processing facility.

MPFLSTxx. The SYS1.PARMLIB member that controls the message processing facility for the system.

MRO. Multiregion operation.

multiple console support (MCS). The operator interface in an MVS system.

multi-access spool (MAS). A complex of multiple processors running MVS/JES2 that share a common JES2 spool and JES2 checkpoint data set.

multiprocessing. The simultaneous execution of two or more computer programs or sequences of instructions. See also parallel processing.

multiprocessor (MP). A CPC that can be physically partitioned to form two operating processor complexes.

multisystem application. An application program that has various functions distributed across MVS images in a multisystem environment.

multisystem console support. Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

multisystem environment. An environment in which two or more MVS images reside in one or more processors, and programs on one image can communicate with programs on the other images.

multisystem sysplex. A sysplex in which two or more MVS images are allowed to be initialized as part of the sysplex.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

MVS router. The MVS router is a system service that provides an installation with centralized control over system security processing.

MVS system. An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.

MVS/ESA. Multiple Virtual Storage/ESA.

MVSCP. MVS configuration program.

N

n-way. The number (n) of CPs in a CPC. For example, a 6-way CPC contains six CPs.

NIP. Nucleus initialization program.

NJE. Network job entry.

no-consoles condition. A condition in which the system is unable to access any full-capability console device.

nonstandard labels. Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

nucleus initialization program (NIP). The stage of MVS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

O

offline. Pertaining to equipment or devices not under control of the processor.

OLTP. Online transaction processing.

online. Pertaining to equipment or devices under control of the processor.

OPC/ESA. Operations Planning and Control.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

operations log. In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

OPERLOG. The operations log.

OPERPARM. In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

OS/390. OS/390 is a network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

OS/390 Network File System. A base element of OS/390, that allows remote access to MVS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

OS/390 UNIX System Services (OS/390 UNIX). The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the OS/390 operating system that allow users to write and run application programs that conform to UNIX standards.

P

parallel processing. The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch). See also highly parallel.

Parallel Sysplex. A sysplex that uses one or more coupling facilities.

partitionable CPC. A CPC that can be divided into 2 independent CPCs. See also physical partition, single-image mode, MP, side.

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it.

permanent data set. A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with temporary data set.

PFK. Program function key.

PFK capability. On a display console, indicates that program function keys are supported and were specified at system generation.

PFKTABxx. The SYS1.PARMLIB member that controls the PFK table settings for MCS consoles in a system.

physical partition. Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

physically partitioned (PP) configuration. A system configuration that allows the processor controller to use both central processor complex (CPC) sides as individual CPCs. The A-side of the processor controller controls side 0; the B-side of the processor controller controls side 1. Contrast with single-image (SI) configuration.

PR/SM. Processor Resource/Systems Manager.

Print Services Facility (PSF). The access method that supports the 3800 Printing Subsystem Models 3 and 8. PSF can interface either directly to a user′s application program or indirectly through the Job Entry Subsystem (JES) of MVS.

printer. (1) A device that writes output data from a system on paper or other media.

processor controller. Hardware that provides support and diagnostic functions for the central processors.

Processor Resource/Systems Manager (PR/SM). The feature that allows the processor to use several MVS images simultaneously and provides logical partitioning capability. See also LPAR.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program function key (PFK). A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

program status word (PSW). A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

pseudo-master console. A subsystem-allocatable console that has system command authority like that of an MCS master console.

PSW. Program status word.

R

RACF. See Resource Access Control Facility.

RAID. See redundant array of independent disk.

RAMAC Virtual Array (RVA) system. An online, random access disk array storage system composed of disk storage and control unit combined into a single frame.

read access. Permission to read information.

recording format. For a tape volume, the format of the data on the tape, for example, 18, 36, 128, or 256 tracks.

recovery. The process of rebuilding data after it has been damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a log.

redundant array of independent disk (RAID). A disk subsystem architecture that combines two or more physical disk storage devices into a single logical device to achieve data redundancy.

remote operations. Operation of remote sites from a host system.

Resource Access Control Facility (RACF). An IBM-licensed program or a base element of OS/390, that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system and logging the detected accesses to protected resources.

restructured extended executor (REXX). A general-purpose, procedural language for end-user personal programming, designed for ease by both casual general users and computer professionals. It is also useful for application macros. REXX includes the capability of issuing commands to the underlying operating system from these macros and procedures. Features include powerful character-string manipulation, automatic data typing, manipulation of objects familiar to people, such as words, numbers, and names, and built-in interactive debugging.

REXX. See restructured extended executor.

RMF. Resource Measurement Facility.

rmode. Residency mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. RMODE states the virtual storage location (either above 16 megabytes or anywhere in virtual storage) where the program should reside.

roll mode. The MCS console display mode that allows messages to roll of off the screen when a specified time interval elapses.

roll-deletable mode. The console display mode that allows messages to roll off the screen when a specified time interval elapses. Action messages remain at the top of the screen where operators can delete them.

routing. The assignment of the communications path by which a message will reach its destination.

routing code. A code assigned to an operator message and used to route the message to the proper console.

RVA. See RAMAC Virtual Array system.

S

SCDS. Source control data set.

SDSF. System Display and Search Facility.

shared DASD option. An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

side. A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

single point of control. The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

single system image. The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

single-image (SI) mode. A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with physically partitioned (PP) configuration.

single-system sysplex. A sysplex in which only one MVS system is allowed to be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system but does not provide signalling services between MVS systems. See also multisystem sysplex, XCF-local mode.

SLR. Service Level Reporter.

small computer system interface (SCSI). A standard hardware interface that enables a variety of peripheral devices to communicate with one another.

SMF. System management facilities.

SMP/E. System Modification Program Extended.

SMS. Storage Management Subsystem.

SMS communication data set. The primary means of communication among systems governed by a single SMS configuration. The SMS communication data set (COMMDS) is a VSAM linear data set that contains the current utilization statistics for each system-managed volume, which SMS uses to help balance space usage among systems.

SMS configuration. The SMS definitions and routines that the Storage Management Subsystem uses to manage storage.

SMS system group. All systems in a sysplex that share the same SMS configuration and communications data sets, minus any systems in the sysplex that are defined individually in the SMS configuration.

software. (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) Contrast with hardware. A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library

routines, manuals, circuit diagrams. Contrast with hardware.

spanned record. A logical record contained in more than one block.

status-display console. An MCS console that can receive displays of system status but from which an operator cannot enter commands.

storage administrator. A person in the data processing center who is responsible for defining, implementing, and maintaining storage manageme policies.

storage class. A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage group. A collection of storage volumes and attributes, defined the storage administrator. The collections can be a group of DASD volume or tape volumes, or a group of DASD, optical, or tape volumes treated as single object storage hierarchy. See tape storage group.

storage management. The activities of data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These can be done either manually or by using automated processes. cThe Storage Management Subsystem automates these processes for you, while optimizing storage resources. See also Storage Management Subsystem.

Storage Management Subsystem (SMS). A DFSMS/MVS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

storage subsystem. A storage control and its attached storage devices. See also tape subsystem.

structure. A construct used by MVS to map and manage storage on a coupling facility. See cache structure, list structure, and lock structure.

subsystem-allocatable console. A console managed by a subsystem like JES3 or NetView used to communicate with an MVS system.

subsystem interface (SSI). An MVS component that provides communication between MVS and JES.

supervisor call instruction (SVC). An instruction that interrupts a program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

support element. A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

SVC routine. A control program routine that performs or begins a contro program service specified by a supervisor call instruction.

symmetry. The characteristic of a sysplex where all systems, or certain subsets of the systems, have the same hardware and software configurations and share the same resources.

synchronous messages. WTO or WTOR messages issued by an MVS system during certain recovery situations.

SYSLOG. The system log data set.

sysplex. A set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads. See also MVS system, Parallel Sysplex.

sysplex couple data set. A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All MVS systems in a sysplex must have connectivity to the sysplex couple data set. See also couple data set.

Sysplex Timer. An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the MVS generic name for the IBM Sysplex Timer (9037).

system control element (SCE). Hardware that handles the transfer of data and control information associated with storage requests between the elements of the processor.

system console. In MVS, a console attached to the processor controller used to initialize an MVS system.

system log (SYSLOG). In MVS, the system log data set that includes all entries made by the WTL (write-to-log) macro as well as the hardcopy log. SYSLOG is maintained by JES in JES SPOOL space.

system management facilities (SMF). An optional control program feature of OS/390 and MVS that provides the means for gathering and recording information that can be used to evaluate system usage.

System Modification Program Extended (SMP/E). In addition to providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog

interface, and supports dynamic allocation of data sets.

Systems Network Architecture (SNA). A description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of networks.

system trace. A chronological record of specific operating system events. The record is usually produced for debugging purposes.

T

temporary data set. A data set that is created and deleted in the same job.

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

terminal user. In systems with time-sharing, anyone who is eligible to log on.

tightly coupled. Multiple CPs that share storage and are controlled by a single copy of MVS. See also loosely coupled, tightly coupled multiprocessor.

tightly coupled multiprocessor. Any CPU with multiple CPs.

Time Sharing Option (TSO). An option on the operating system; for OS/390 the option provides interactive time sharing from remote terminals.

TOR. Terminal-owning region.

transaction. In APPC/MVS, a unit of work performed by one or more transaction programs, involving a specific set of input data and initiating a specific process or job.

transaction program (TP). For APPC/MVS, any program on MVS that issues APPC/MVS or CPI Communication calls, or is scheduled by the APPC/MVS transaction scheduler.

U

undelivered message. An action message or WTOR that cannot be queued for delivery to the expected console. MVS delivers these messages to any console with the UD console attribute in a system or sysplex.

uniprocessor (UP). A CPC that contains one CP and is not partitionable.

UP. Uniprocessor.

V

VM. Virtual Machine.

virtual telecommunications access method (VTAM). A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2 operating systems.

volume. (1) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

volume serial number. A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents (VTOC). A table on a direct access volume that describes each data set on the volume.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method.

VTOC. Volume table of contents.

W

wait state. Synonymous with waiting time.

waiting time. (1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

WLM. MVS workload management.

wrap mode. The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

write-to-log (WTL) message. A message sent to SYSLOG or the hardcopy log.

write-to-operator (WTO) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

write-to-operator-with-reply (WTOR) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

WTL message. Write-to-log message

WTO message. Write-to-operator message

WTOR message. Write-to-operator-with-reply message.

X

XCF. Cross-system coupling facility.

XCF PR/SM policy. In a multisystem sysplex on PR/SM, the actions that XCF takes when one MVS system in the sysplex fails. This policy provides high availability for multisystem applications in the sysplex.

XCF-local mode. The state of a system in which XCF provides limited services on one system and does not provide signalling services between MVS systems. See also single-system sysplex.

XRF. Extended recovery facility.

IBM Redbooks evaluation

ABCs of OS/390 System Programming Volume 4
SG24-5654-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

_____

SG24-5654-00
Printed in the U.S.A.

SG24-5654-00