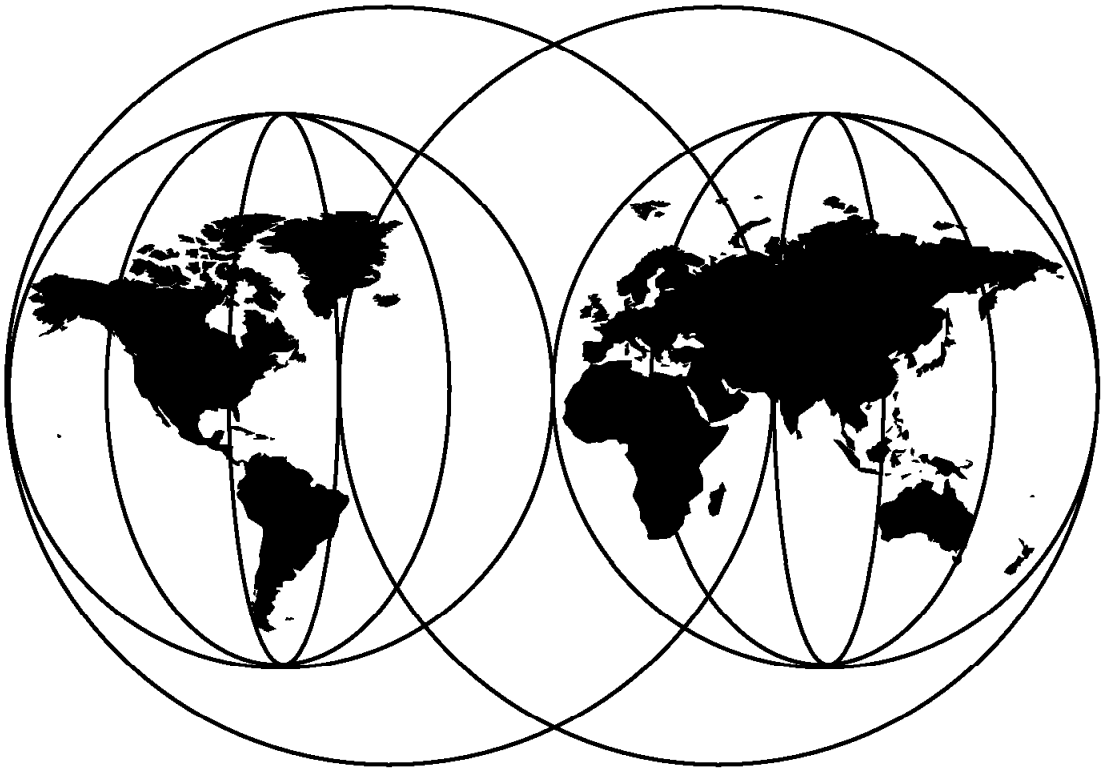
Connecting Auto UNIX System to VM and VSE

Walter Bieg, Leif-Erik Kristiansen, R. Werner Schröter, Stan Weeks



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-5377-00

International Technical Support Organization

SG24-5377-00

Connecting Auto UNIX System to VM and VSE

January 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special Notices” on page 159.

First Edition (January 1999)

This edition applies to Version 1, Release 1 of OS/390 Automated UNIX System Option for VM, VSE and OS/390, Program Number 5655-A97.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 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 will help you install, tailor and configure the new OS/390 Automated UNIX System Option for VM, VSE and OS/390 (Auto UNIX System). With Auto UNIX System, customers now have the flexibility to exploit new UNIX application opportunities and access existing VM, VSE or OS/390 applications and data.

Guidance and samples are provided to help you with the installation, configuration and connectivity of Auto UNIX System, as well as showing how you can use the product in a normal installation with VSE/ESA or with VM/ESA. Special considerations are given to the usage of FTP, Network File System and MQSeries.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working for the International Technical Support Organization Poughkeepsie Center in the IBM Germany Development location in Böblingen.

Leif-Erik Kristiansen, who was the project leader.

Walter Bieg, IBM Germany.

R. Werner Schröter, who works for an IBM Partner Company in Brazil.

Stan Weeks, from Stan Weeks, Inc. in Seattle.

The four gentlemen represent together approximately 100 years of VM and VSE experience, working in areas such as development, systems programming and so forth in various organizations inside and outside IBM.

Thanks to the following people for their invaluable contributions to this project:

Franz-Peter Boley
IBM Germany Development

Ulrich Hild
IBM Germany Development

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 "ITSO Redbook Evaluation" on page 189 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. Introduction to Auto UNIX System

OS/390 Automated UNIX System Option for VM, VSE and OS/390 (Auto UNIX System) Release 1 is a pre-configured and pre-built OS/390 Version 2 Release 6 system, that provides all the system programming required to support an OS/390 UNIX System Services based application environment.

Auto UNIX System works as an automated OS/390 application server, and enables its customers to implement OS/390 UNIX applications without requiring OS/390 system or programming skills.

You can perform system administration tasks from the UNIX shell interface of Auto UNIX System and Control Center. The Auto UNIX System Control Center is described in 1.3, “Auto UNIX System Control Center” on page 5. All necessary system setup tasks have already been performed. This includes customization of the base operating system components:

Data Facility Storage Management Subsystem (DFSMS)
Advanced Program-to-Program Communication (APPC)
Job Entry Subsystem (JES)
System Management Facility (SMF).

The definition of a hierarchical file system (HFS) for OS/390 UNIX System Services, and a default customization of the OS/390 UNIX shell is also done. In addition, an extensive security structure is set up using the Resource Access Control Facility (RACF) component in the OS/390 Security Server.

Auto UNIX System provides integration with existing OS/390, VM/ESA and VSE/ESA applications and data. Auto UNIX System uses communications facilities such as TCP/IP, APPC (SNA) and MQSeries to enable application integration.

This is particularly valuable for VM/ESA and VSE/ESA customers who do not have OS/390 skills, and to OS/390 customers looking for a departmental or remote distributed S/390 UNIX solution where no OS/390 skills exist.

With the growing number of UNIX applications, many organizations with VM/ESA or VSE/ESA systems may want to explore these new applications, but without having to invest in specific UNIX hardware. For existing VM or VSE installations the new Auto UNIX System will be an interesting offer to start exploring UNIX applications.

Auto UNIX System either runs in a standalone processor, or together with VM/ESA and/or VSE/ESA on the same processor, as a VM/ESA guest system or in an LPAR.

With Auto UNIX System you:

- Do not have direct access to traditional OS/390 applications such as Time Sharing Option (TSO), Customer Information Control System (CICS), Information Management System (IMS), or batch processing.
- Can not modify the system structure.
- Are not necessarily licensed for all included products and features in the system.

1.1 Auto UNIX System Optional Features

The following features can be ordered when you place the order for Auto UNIX System:

- Auto UNIX System C/C++
- Auto UNIX System DB2
- Auto UNIX System MQSeries

The Auto UNIX System C/C++ code is shipped disabled, as a part of the Auto UNIX System. It can be enabled if you order the optional feature Auto UNIX System C/C++.

1.1.1 Auto UNIX System C/C++

Auto UNIX System C/C++ feature consists of the C/C++ compiler and debugger, and can be used to create, modify and debug C or C++ UNIX based applications.

Auto UNIX System C/C++ consists of:

- An OS/390 Version 2 Release 6 C compiler
- An OS/390 Version 2 Release 6 C++ compiler
- A starter set of class libraries
- An OS/390 interactive debug tool
- Some C/C++ application development utilities

1.1.2 Auto UNIX System DB2

The Auto UNIX System DB2 feature provides DB2 (Version 5) as a database management system.

Some of the tasks that can be accomplished using the Auto UNIX System DB2 feature are:

- Starting, stopping and recycling DB2
- Issuing DB2 commands from the OS/390 shell
- Invoking DB2 utilities from the OS/390 shell, with OS/390 UNIX files for SYSIN and SYSPRINT
- Controlling DASD space available for DB2 databases
- Issuing SQL statements to DB2

1.1.3 Auto UNIX System MQSeries

Auto UNIX System MQSeries feature supports connectivity to VSE/ESA systems; VM/ESA is currently not supported. You can use the OS/390 shell to access MQSeries commands and utilities. Sample programs are provided with Auto UNIX System to show how to do this.

1.2 Auto UNIX System Hardware Requirements

Hardware for Auto UNIX System must be ESA capable. The Auto UNIX System requires the following hardware features:

- S/390 processor
- DASD devices and volumes
- Tape device
- Network connections

For required main storage, please see Table 1 for detailed information.

1.2.1 S/390 Processor

The S/390 processor must be ESA capable. Auto UNIX System installation is **not** supported on P/390, R/390 or S/390 Integrated Server processors because the Control Center functions are not supported on those processors.

Table 1. Storage Required on the Host

Storage type	Size	Comment
Main storage	128 MB	Base system without optional products or features and configured for only a few users
Additional storage for DB2	128 MB	If you plan to install and use DB2
Additional storage for Lotus Domino	256 MB	If you plan to install and use Lotus Domino

1.2.2 DASD Devices and Volumes

Auto UNIX System supports the following DASD devices:

- 3390
- 3380
- 9345

The two system volumes must be the same DASD device type, and they require a minimum of 1.5GB each (for example 3390-2 or larger, 3380-K or 9345 B22).

The absolute minimum DASD requirement is about 16.5GB, excluding optional feature requirements. This number can only be realized by VM/ESA users who can optimize the size of their minidisks. For users using real DASD, the total amount of required DASD will be larger depending on the actual device types. For information on DASD requirements please see Table 2 on page 4, and Table 3 on page 4.

Note: The following two tables only have DASD information for 3390. If you need information for 9345 or 3380 please refer to *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

Table 2. Required Volumes, Cylinders Needed on 3390					
Volume	**Used for**	**GB**	**No of 3390 cyl**	**No of 3390 Mod 2**	**No of 3390 Mod 3**
EZM001	System residency volume	1.5	2225	1	1
EZM003	System residency volume	1.5	2225	1	1
EZM004	Diagnostic data	1.5	2225	1	1
EZM005 - EZM099	IBM defined HFSs	1.8	2118	1	1
EZMT01 - EZMT99	Target libraries	5.1	6000	3	2
EZMS01 - EZMS99	Temporary data	5.1	6000	3	2
Total		16.5	20793	10	8
EZMU01 - EZMU99	User provided HFSs	Depends on user data			

Table 3. Required DASD Volumes for Auto UNIX System Features					
Volume	**Used for**	**GB**	**No of 3390 cyl**	**No of 3390 Mod 2**	**No of 3390 Mod 3**
EZMA01 - EZMA99	DB2 MQSeries (code)	1.8	2118	1	1
EZMD01 - EZMD99	DB2 MQSeries (data)	0.9	1059	1	1
Total		2.7	3177	2	2

Note: The features as described in Table 3 are DB2 and MQSeries.

1.2.3 Tape Device

Tapes are used for Auto UNIX System installation, service and recovery. Auto UNIX System is shipped on either 3480 or 4mm DAT tapes.

1.2.4 Network Connections

For installation of Auto UNIX System at least one TCP/IP dedicated LAN attachment port must be defined, and configured for the TCP/IP Passthru mode. This attachment can either be an OSA-2 adapter card or a 3172 control unit. There is also the option of defining CTC connections.

Auto UNIX System is also pre-configured for an SNA connection. You can use SNA connection via CTC to VM/ESA or VSE/ESA.

Note: When installing Auto UNIX System it is **strongly** recommended to define all unit addresses for all communication devices you ever plan to use, since they can not be added after installation.

1.3 Auto UNIX System Control Center

The Auto UNIX System Control Center allows the Auto UNIX System administrator to do several administration tasks without having direct access to the OS/390 console, or having specific OS/390 skills. These tasks include:

- Install Auto UNIX System (inclusive features)
- Start and shutdown of Auto UNIX System
- Apply service to Auto UNIX System
- Restart TCP/IP
- Display the event log
- Create a standalone dump
- Recover the system

The Auto UNIX System Control Center is a Graphical User Interface (GUI) that can run on either:

- The Hardware Management Console (HMC)
- A Standalone Support Element (SA-SE)
- An OS/2 workstation (only supported when running Auto UNIX System under VM/ESA)

Note: Auto UNIX System can only be controlled by a Control Center running on an HMC or SA-SE when Auto UNIX System is running in an LPAR or a standalone environment. An HMC or SA-SE is not supported for running the Control Center as a VM/ESA guest system.

1.3.1 OS/2 Workstation Requirements

The Control Center requires a Personal Computer with OS/2 Warp Release 4 or higher connected to the VM/ESA where Auto UNIX System is the guest system, and PCOM/3270 Version 4.1 or higher.

1.4 Graphical Representation of Auto UNIX System

You can see a simple graphical representation of Auto UNIX System in Figure 1 on page 6. Since Auto UNIX System is pre-built for you, you only need to develop skills in those applications and application enablers you choose to implement.

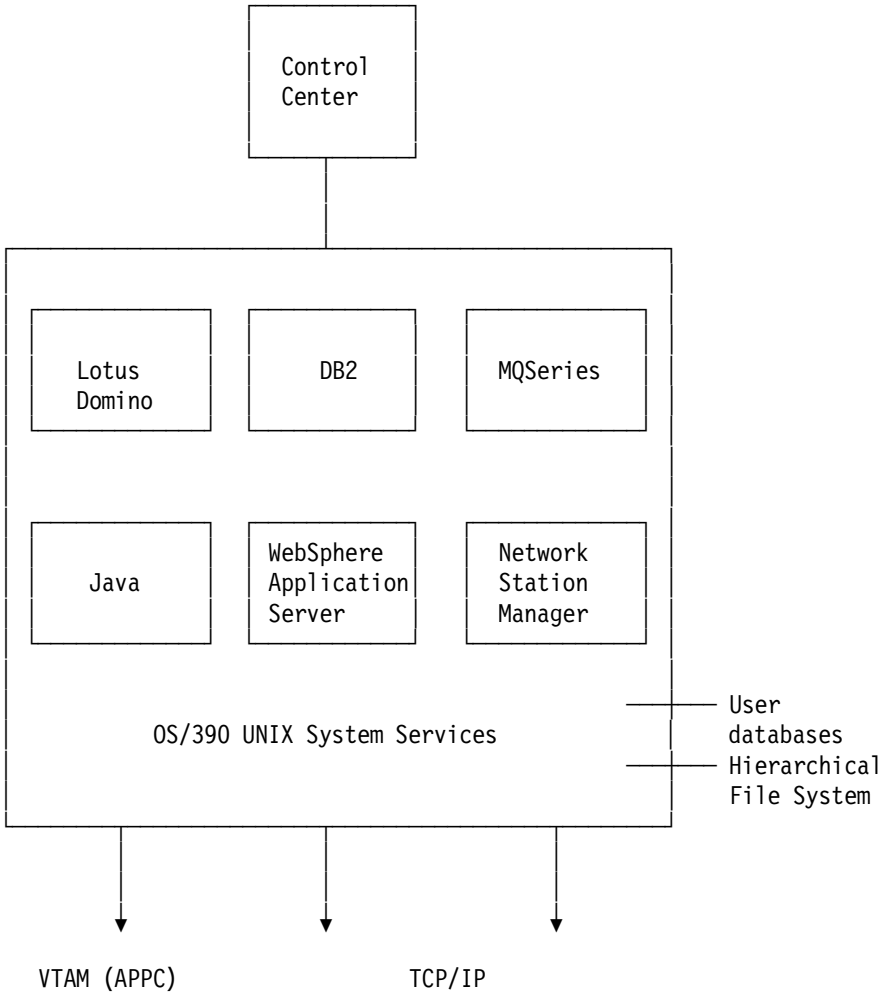


Figure 1. Graphical Representation of Auto UNIX System

Chapter 2. Installing Auto UNIX System

This chapter describes:

- Hardware features required for Auto UNIX System.
- Overview of installation steps.
- Installation steps in detail.

2.1 Overview of Hardware Requirements

The following hardware features are required for Auto UNIX System:

S/390 Processor
Processor storage as follows:

- Auto UNIX System requires a dedicated minimum of 128MB of storage. This is the base system without optional products and configured for only a few users.
- If you plan to install and use the Auto UNIX System DB2 feature, an additional 128MB of storage is required.
- If you plan to install and use Lotus Domino, an additional 256MB of storage is required.

Tape drive
Auto UNIX System supports only one tape drive. Tapes are used for installation, service and recovery. Supported tape drives are:

- 3490 includes 3490E and 4 mm DAT tape, which emulates the 3490
- 3480 includes all 3480 devices and 3490 non-E models.

Disk volumes
There are three types of disk volumes supported: 3380, 3390 and 9345. The capacity of a volume depends on the type. Disk volumes in Auto UNIX System are divided into groups. The volumes in a group must always be the same type. For each group a minimum disk capacity is required, and can vary from one release to the next. For detailed information see Table 2 on page 4, and Table 3 on page 4.

Network connections
You need to be able to do an **rlogin** or **telnet** from a workstation to Auto UNIX System. Therefore you need a PC or a UNIX workstation for the TCP/IP Auto UNIX System connection. Optionally, you can connect to another system, for example, VM/ESA or VSE/ESA. For this, you may use channel to channel (CTC) or LAN connection.

Also supported is the **Networkstation**. For setting up and customization refer to *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993.

Note: It is not possible to add any communication device numbers after the installation is complete. This means that for all communication devices you plan to use, you must define the device numbers when you install Auto UNIX System.

2.2 Overview of Installation Steps

Auto UNIX System is installed from a set of tapes.

You can install Auto UNIX System as:

- A logical partition (LPAR) of an S/390 processor.
- A VM/ESA Guest System.
- A standalone system on an S/390 processor.

2.2.1 Logical Partition

When running Auto UNIX System on an LPAR, the LPAR must have a minimum size of 128MB.

Auto UNIX System comes with an initial hardware configuration defined. That configuration is described in the Input/Output Definition File (IODF). Auto UNIX System needs three predefined unit addresses, two for the system packs, and one for the Auto UNIX System console. When installing Auto UNIX System on an LPAR, you must update your IOCDS to include these predefined addresses, as well as the other addresses you plan to use. The default addresses are 120 and 121 for system volumes, and 700 for the OS/390 console. The system volumes use the device type you specified when you order Auto UNIX System.

2.2.2 VM/ESA Guest System

Auto UNIX System can be installed as a single, or as multiple VM/ESA guest systems. Each guest system image is controlled by its own dedicated Control Center.

The Control Center requires two host sessions for the VM/ESA guest system to be controlled. The first session is used as the Auto UNIX System guest system console from which the Auto UNIX System start can be performed. The second session is used by the local terminal for accessing the OS/2 console.

2.2.3 Installation Steps

During installation you will be guided with information from the Control Center. After the Control Center is up and running, you will define the hardware, do a physical restore from two tapes, do a logical restore of the rest of the tapes, and then do an IPL of the Auto UNIX System.

The following is a short list of the installation steps you have to go through. All the steps will be described in more detail in 2.3, “Installing Auto UNIX System in Detail” on page 9.

1. Install the Control Center on your workstation.
2. Start the Control Center.
3. Start the Installation dialog.
4. Go through the installation dialog tabs:
   - On the Devices tab you can enter I/O configuration data (tape, DASD volumes and terminals).
   - On the Communication tab you can enter data for your network communication.

- On the Features tab you may decide to install the Auto UNIX System DB2 feature or the Auto UNIX System MQSeries feature, or both.
- On the Trace tab you can turn on tracing. However, do not turn on tracing unless explicitly instructed by IBM support personnel.

5 To start the restore process click on the installation button and follow the dialog.

Note: If for any reason, the installation process is interrupted, you can resume the process. To do this, on the Control Center menu, Click on **Install ===> Install Auto UNIX System**. The Installation saves the data you have entered, and tries to recover up to the point where processing stopped.

2.3 Installing Auto UNIX System in Detail

The following will briefly take you through the installation of the Control Center, and the installation of Auto UNIX System under VM/ESA.

If you do not use VM/ESA the Control Center is part of the Hardware Management Console (HMC), or a Standalone Support Element (SA-SE), and no separate installation is required. For configuration of HMC or SA-SE, please see *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

2.3.1 Installing the Control Center

The code for the Control Center is delivered on a diskette.

From an OS/2 Window start the: [**a:**]**EZMCC.EXE** program from the diskette. After loading, the picture in Figure 2 on page 10 will pop up.

If you are not running as a guest under VM/ESA, for example LPAR or standalone system, refer to the corresponding chapters in *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

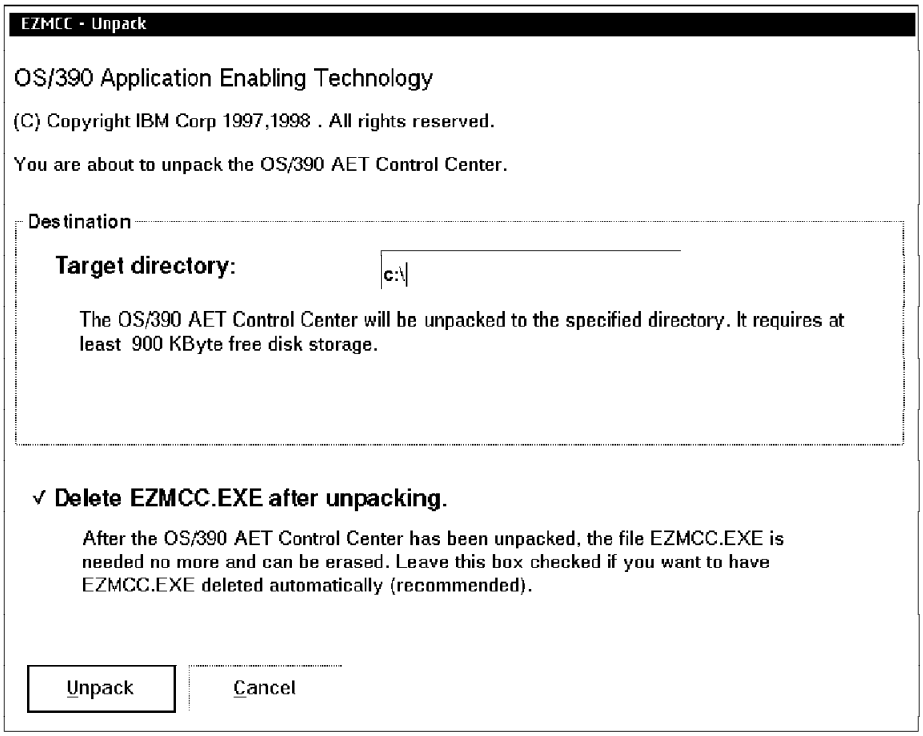


Figure 2. Unpacking the Control Center to OS/2 Hard Disk

You may change the Target directory to your requirements. Click on the **Unpack** button to unpack the code to the directory you specified.

2.4 Start and Configure the Control Center

Note: It is recommended to start the Control Center from the directory where you installed it. The Control Center stores important information in two initialization files (EZMAET.INI and EZMFLEX.INI). To ensure that these files are accessible you must always start the Control Center from the same OS/2 directory where you installed it.

Make sure that both emulator sessions you are going to use are up and running and that you have set up your VM correctly.

We have included two figures in Appendix A, “Sample VM/ESA Directory and Profile EXEC” on page 109. See Figure 49 on page 109 for an example of a VM Directory for a system with three CPUs, and Figure 50 on page 110 for an example of a PROFILE EXEC for the VM System.

1. Start the Control Center by running the **EZMEHMPR.EXE** from an OS/2 window. The first time the Control Center is started the **Options Notebook** opens.

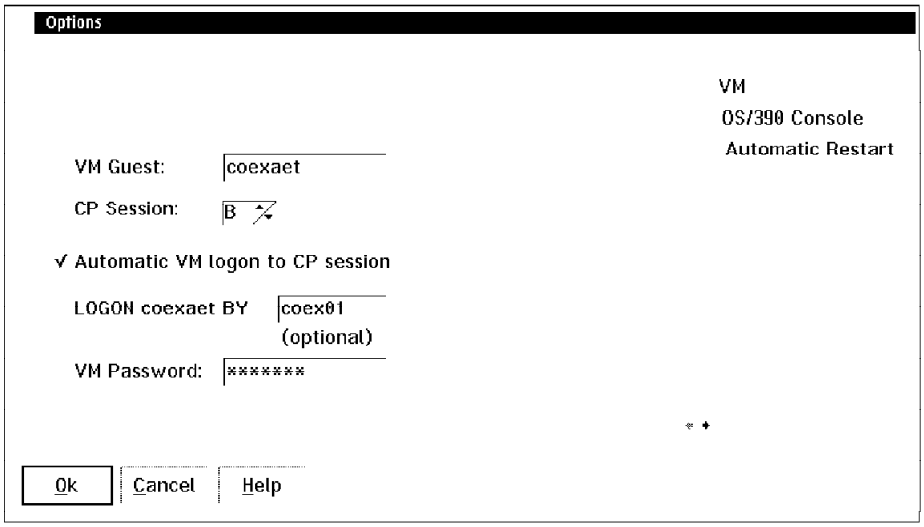


Figure 3. The VM Tab in Options Notebook for the Control Center

2 Enter the requested values in the Options notebook.

- **VM Guest**
  Specify the user ID of the VM guest where Auto UNIX System will be installed.
- **CP Session**
  Select the emulator session you want to use for logging on to the Auto UNIX System.
- **Automatic VM logon to CP session**
  Optional, if you check this box the Control Center will attempt to log on to the VM guest user ID given. You also need to enter a password in the password field, otherwise you need to log on to the CP session manually.
- **LOGON** VM-Guest **BY**
  Optional, enter a user ID for the VM logon facility.
- **VM Password**
  Enter the password for the VM guest, or, if you are using the logon by facility, the password of the logon by user ID.

  Note: **LOGON coexaet BY** means the use of a shared password with another VM guest.

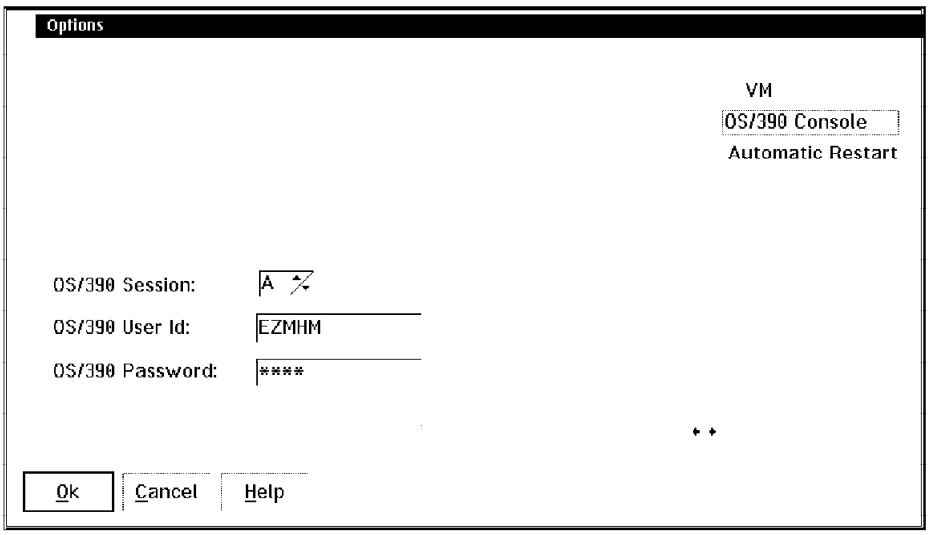


Figure 4. OS/390 Console Tab in Options Notebook for Control Center

- **OS/390 Session**
  Select the emulator session you want to use as the OS/390 operator console.
- **OS/390 User ID**
  This is the Control Center user ID. Do not change the default EZMHM.
- **OS/390 Password**
  Password of the Control Center user ID. Do not change the default RHEP.

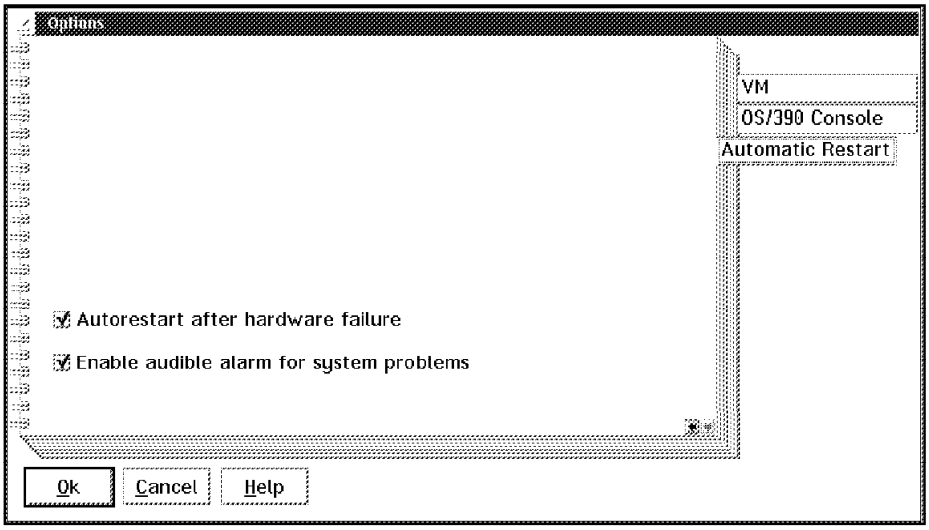


Figure 5. The Automatic Restart Tab in Options Notebook for the Control Center

- **Autorestart after hardware failure**
  Optional, but recommended. If this option is checked the Control Center attempts to restart Auto UNIX System if the system enters an unexpected disabled wait-status.
- **Enable audible alarm for system problems**
  Optional. When checked, a signal sounds when problems occur.
- Press the **OK** button to end the options dialog.

The Control Center is now prepared to start the installation dialog to install Auto UNIX System.

2.5 Control Center Operation

When you install Auto UNIX System as a VM guest, the Control Center controls the VM guest through two 3270 emulator sessions. The CP session is used to control system startup, system shutdown and system status. The Control Center uses the OS/390 console session to watch for system messages and the status of the subsystem.

2.6 Disconnecting and Reconnecting the VM Guest Machine

It is possible to disconnect and reconnect the virtual machine where Auto UNIX System is running. To disconnect:

1. Exit the Control Center by pressing F3.
2. In the emulator session window, disconnect the CP session by issuing: **CP Disc**
3. Disconnect the console session: In the emulator session window for the operator console, click **Communication = = = > Disconnect** as shown on Figure 6.

Figure 6. Emulator Session Window, Disconnecting the Operator Console

To reconnect, restart the Control Center. The Control Center logs on to the CP session, issues a reset to Graf 700, and dials to Graf 700 in the console session.

If no operator console appears in the dialed session the Control Center issues an external interrupt (#cp ext) in the CP session. This retrieves the operator console.

2.7 Starting the Installation Dialog

If you run the Control Center on **SA-SE**, click **OK** on the appropriate icon to start Control Center.
If you are using the HMC console, there is no **icon**, you must start **EZMEHMPR.EXE** from an OS/2 window.

1. Start the Control Center from the directory in OS/2 where you installed the Control Center. On the command line enter **EZMEHMPR.EXE**
2. On the message box that pops up

   EZM421D - System needs initial load (IPL). Do you want to start an IPL immediately?

   Click **NO.**
3. If you are installing on an LPAR, select the image name from the list.
4. Click **Install ===> Install Auto UNIX System** on the Control Center menu bar. For the installation dialog see Figure 7.

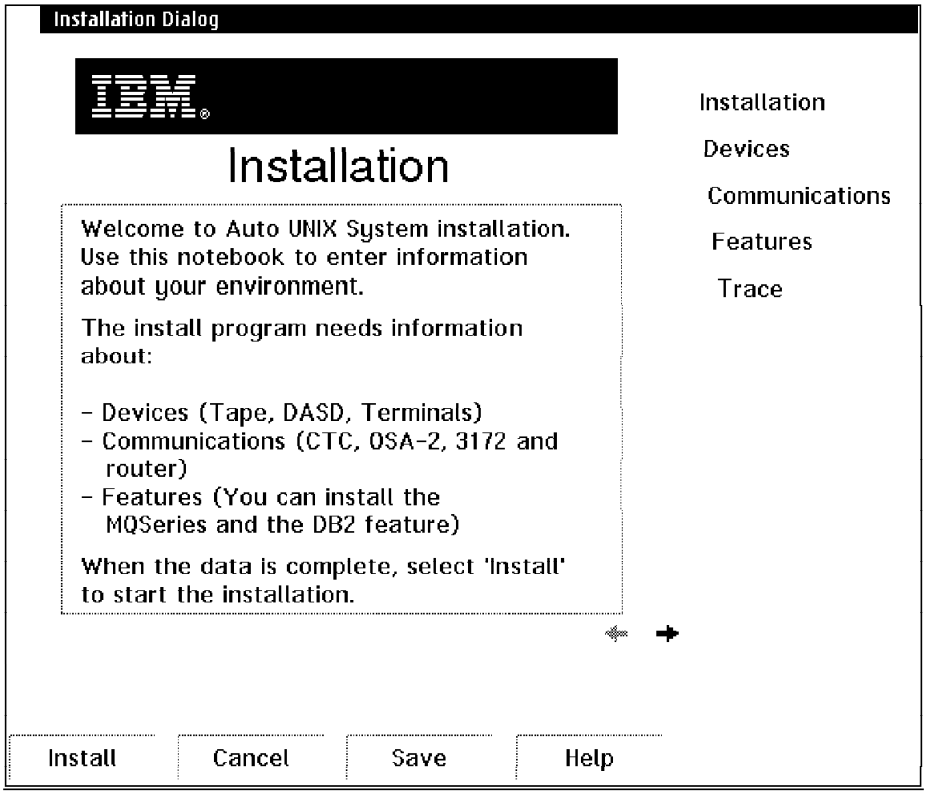


Figure 7. The Installation Dialog

On the installation dialog, there are four buttons:

- **Install** performs some checking of the data you have entered, and starts the installation dialog.
- **Cancel** leaves the dialog **without** saving your data.

- **Save** saves the data you have entered so far, and leaves the dialog.
- **Help** calls the Control Center on-line help for details about the fields.

Continue now by going through the installation dialog tabs.

2.8 Defining Devices

On the Devices tab you can see three pages that let you define:

- Tape
- DASD volumes
- Terminal addresses

2.9 Tape

Auto UNIX System supports only one tape drive. Ensure that the tape drive is accessible to the system where you are installing Auto UNIX System. You need to define the following for the tape drive:

- Device address. The device address is a four digit hexadecimal number.
- Device type. Auto UNIX System supports following device types for installation:
  - use **3490** for 3490E devices and 4mm DAT tape, which emulates the 3490 device.
  - use **3480** for 3480 devices and non-E models of 3490.

2.10 DASD Volumes

Use the DASD page to add required system DASD volumes and optional user-specific DASD volumes. Refer to Table 2 on page 4 and Table 3 on page 4 for the device types, device addresses and number of cylinders needed. The DASD page looks like Figure 8 on page 16.

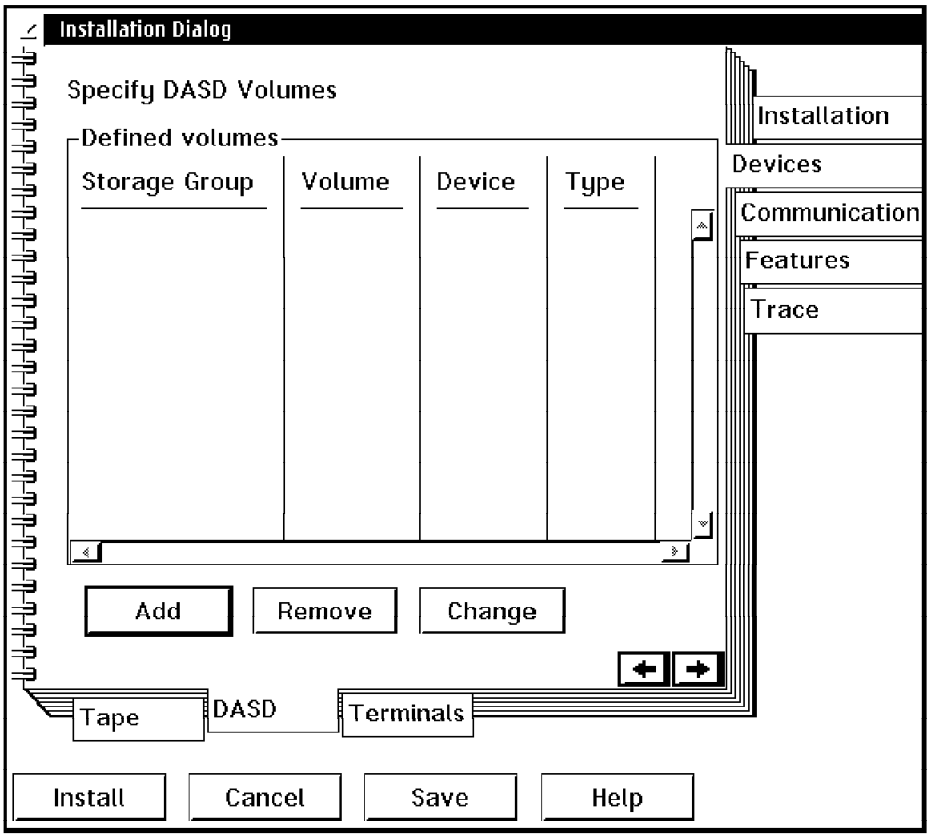


Figure 8. The DASD Page

2.11 Adding DASD Volumes

For example, assume that Diagnostic data EZM004 should go to a Device Type 3390 with address 122.

See predefined storage groups in Table 2 on page 4.

To add this DASD volume to the table in the installation dialog, follow this procedure:

1. On the DASD page click **Add**.
2. Select a predefined storage group, for example Diagnostic Data.
3. Enter the device Address.
4. Enter the number of volumes needed for the amount of cylinders required for this storage group, see Table 2 on page 4.
5. Select a device type from the predefined types. Refer to Figure 9 on page 17.

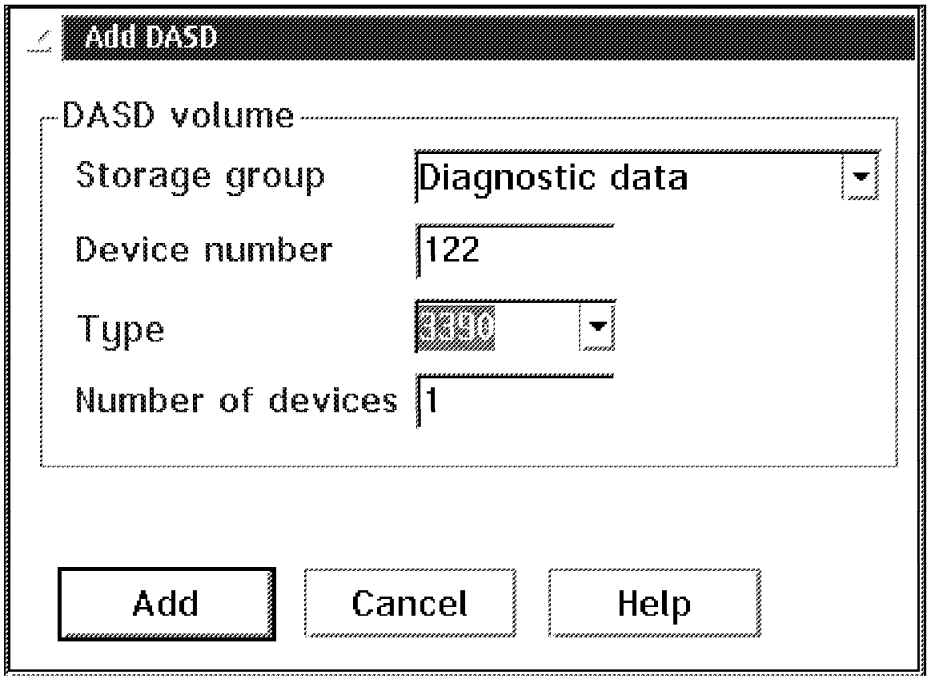


Figure 9. The Add DASD Panel

6 Click **Add**.

7 Continue this step for each predefined storage group, type and number of devices.

After defining all storage groups the DASD page should look like the picture in Figure 10.

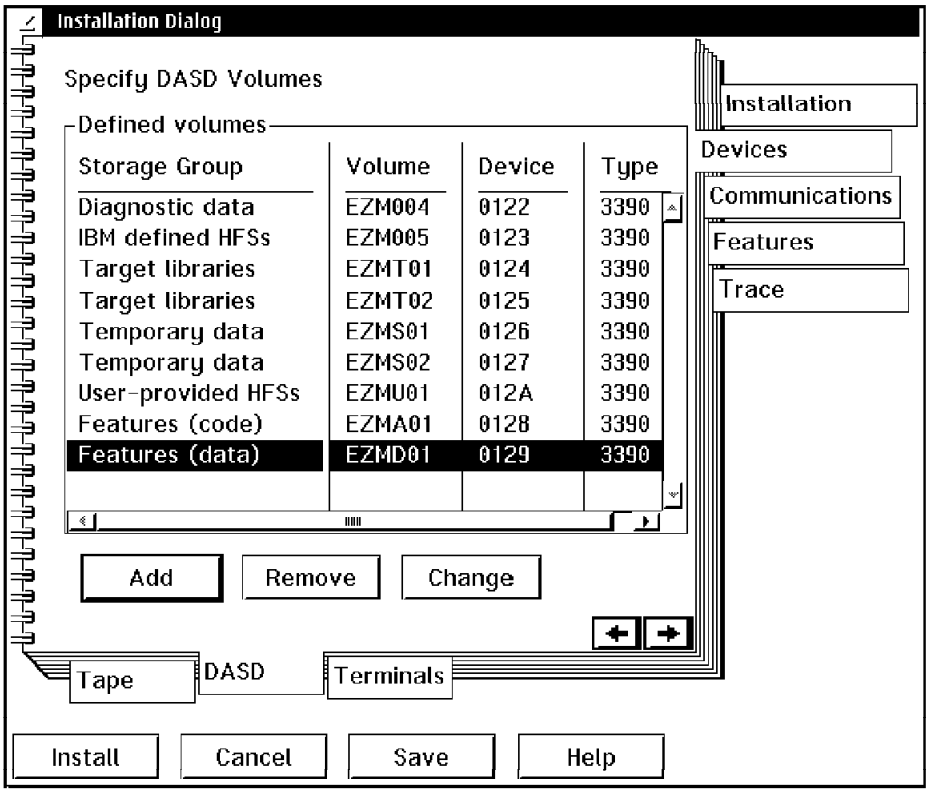


Figure 10. The DASD Page with Defined Volumes

Note: The system volumes (EZM001 and EZM003 are automatically added by Auto UNIX System), with the type you specified when you ordered Auto UNIX System, at addresses 0120 and 0121. The two system volumes are not shown in Figure 10.

2.12 Removing a DASD Volume

To remove a DASD volume do the following:

1. Select the row of the volume you want to remove in the Defined volumes table.
2. Click on **Remove**.

2.13 Changing Details of a DASD Volume

Change the device number, DASD type, or storage group for an existing DASD volume as below:

1. Select the volume in the table.
2. Click on **Change**.
3. Make the changes you want to the data displayed on the Change DASD panel.
4. Click on **Change**.

The volume appears in the Defined volumes list with its changed data.

2.14 Terminals for Remote Service

Switch to the Terminal page to specify up to four device numbers for local terminals. IBM service can use the terminals for remote support service. You can use the terminals to access OS/390 if you later decide to transfer to a full OS/390. IBM service personal requires at least one terminal in order to access the system.

The device number for a local terminal is a four-digit number. Typical values are 0701, 0702, 0703, and 0704. See the example in Figure 11.

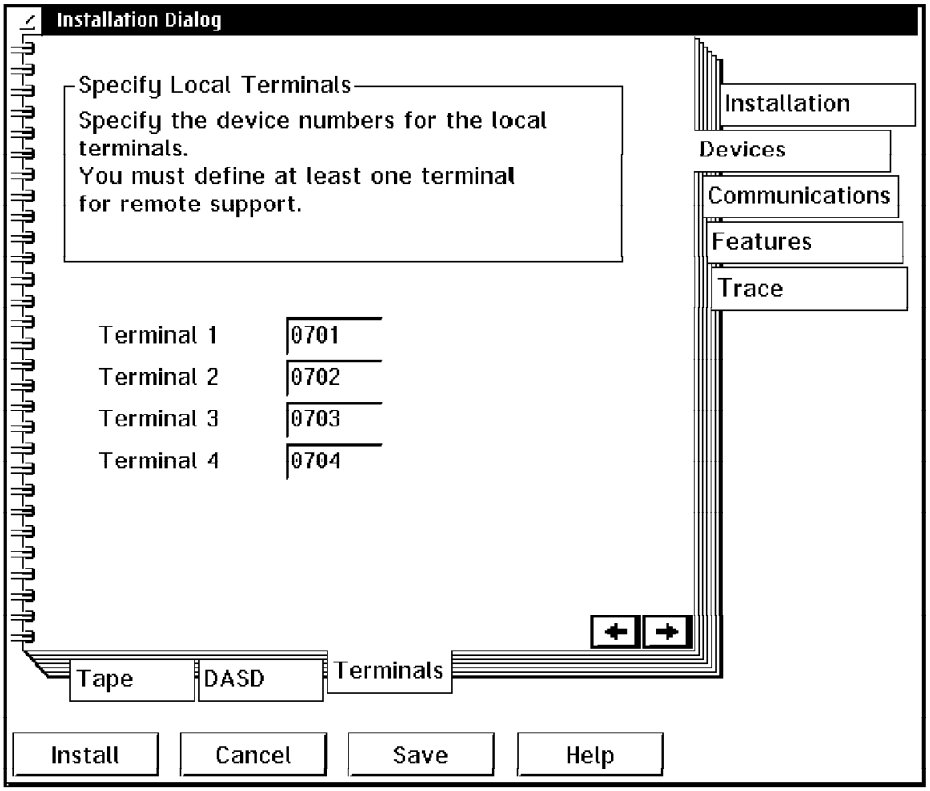


Figure 11. The Terminal Page

Note: Device number 0700 is reserved.

2.15 Configuring Communications

Note: You cannot change the I/O configuration later. So even if you do not need the channels now, you should define them for later use.

Switch to the Communications tab to define access to:

- A workstation from which you **rlogin** or **telnet** to Auto UNIX System.

  To connect to a workstation, you can use TCP/IP through any of the following alternatives:

- Ethernet Token-Ring (ENTR) OSA-2 adapter card (two ports).
- 3172 control unit (four ports).
- CTC channel pair, when using VM/ESA or VSE/ESA as a router.

You must have either an OSA-2 or a 3172 connection for native TCP/IP passthrough mode or

- A VM/ESA or VSE/ESA system.

  To connect to VM/ESA or VSE/ESA, you use CTC for which there are two channel pairs:

  - One channel pair is for Virtual Telecommunications Access Method (VTAM), and you need this pair for advanced program-to-program communication (APPC). On VSE/ESA you can use this channel pair for MQSeries communication in addition to APPC communication.
  - The second channel pair is for TCP/IP communication with a VM/ESA or VSE/ESA system. Use a CTC channel for TCP/IP if you need a high-speed connection, or if you need to take some load off the LAN.

All of the different possibilities are described in detail in the manual *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

2.16 Configuring VTAM and TCP/IP Connection

In order to:

- Enable VTAM communication with another system VM/ESA or VSE/ESA.
- Configure CTC for TCP/IP to enable communication with another system such as VM/ESA or VSE/ESA.

Go to the CTC page as shown in Figure 12 on page 21.

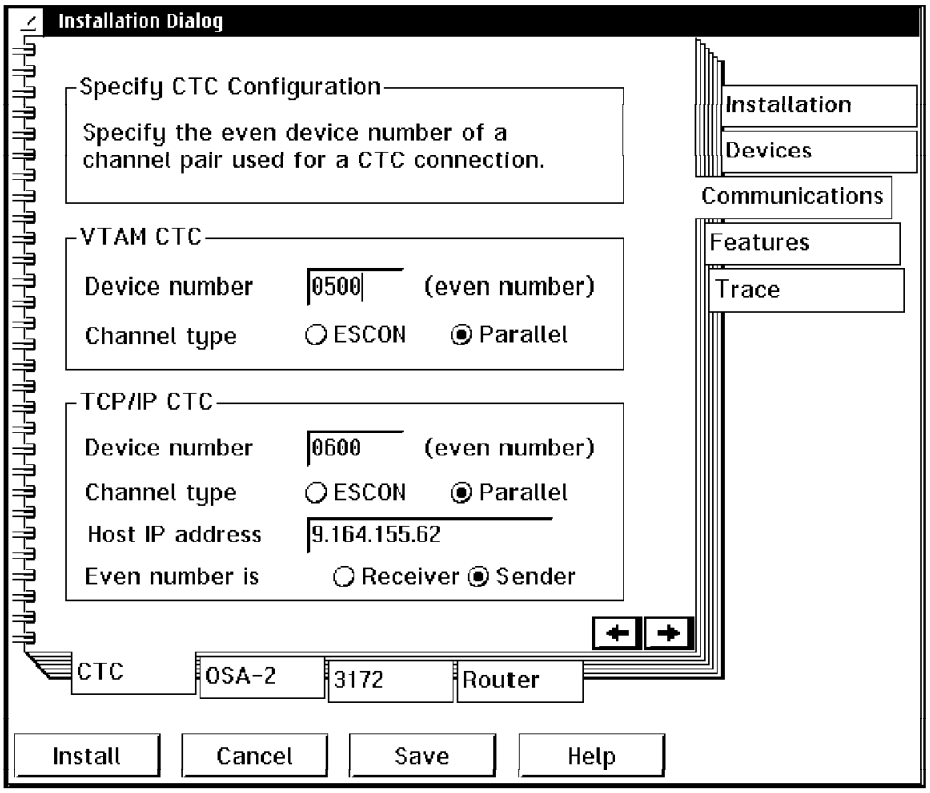


Figure 12. The CTC Page for VTAM and TCP/IP Connection

Do the following steps:

1 Specify the CTC channel pair for:

- VTAM connection
- TCP/IP connection

You only specify the even numbered address, and the next higher value automatically becomes the second channel address.

2 Specify if the channels are ESCON or Parallel:

- If you are using an ESCON CTC-CNC (high speed serial orange cable) specify ESCON.
- If you are using a 3088 (blue cable) or a VM (virtual CTCA), specify parallel.

To activate the VTAM connection you must connect the systems, and set up the VM/ESA or VSE/ESA system. Refer to *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993 for information on how to do this.

In addition, for TCP/IP specify the following:

3 The Auto UNIX System IP address. You need at least one IP address for either:

- OSA-2
- 3172
- CTCA

in order to have a default interface. The IP host address of the default interface becomes the Auto UNIX System IP address that you use when you do an **rlogin** or **telnet** to the system.

4 Specify if the device number is receiver or sender.

Note: Keep the default as sender but remember this when configuring TCP/IP.

2.17 Specifying a Default Interface and a Router for TCP/IP

Switch to the Router page to specify a default interface and a router, we used VM as a router. The Router page looks like Figure 13.

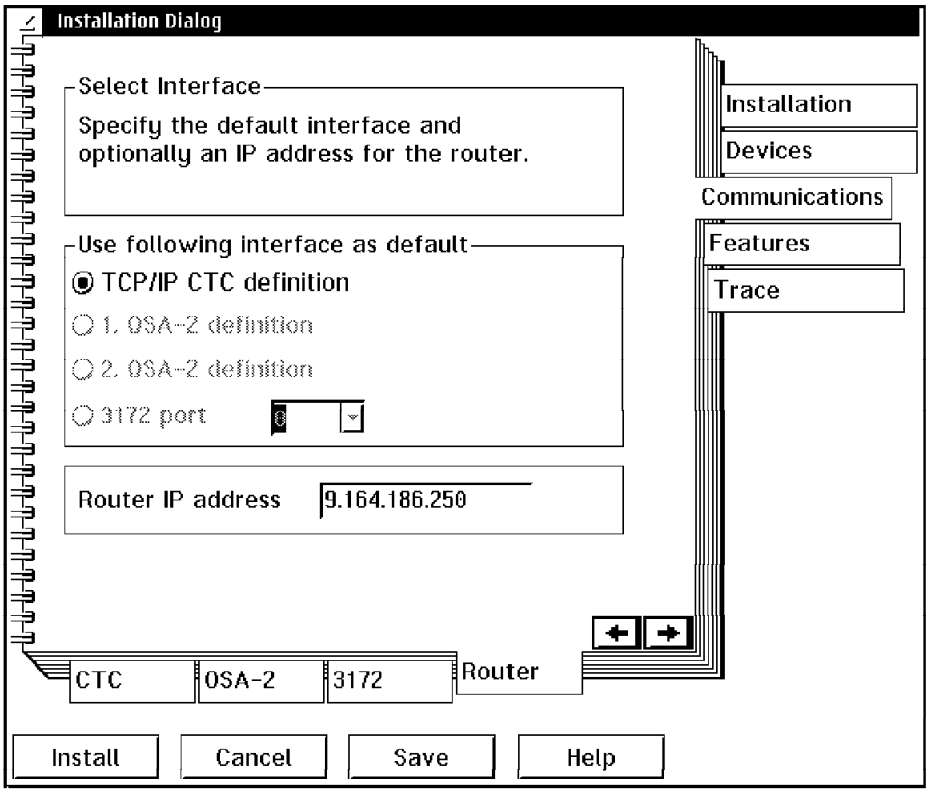


Figure 13. The Router Page

To specify the default interface:

Click the radio button of the interface you want to use for default routing. If you specified an IP host address for an interface, the radio button for that interface is enabled.

To specify a router:

Enter the IP address in the Router address field. You need a Router to go outside the local subnet. If you do not use a router leave the Router IP address field empty.

Note: If you do not have a router IP address now, you can add one later from Auto UNIX System. How to do this is described in the manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993.

2.18 Installing Auto UNIX System DB2 and MQSeries Features

If you ordered Auto UNIX System DB2 or Auto UNIX System MQSeries feature, these features are provided on separate tapes, in addition to the Auto UNIX System tapes.

Note: You can install the features now, or later, using the **install** command.

If you decide to install the features later, click the **Install** button now and proceed with installation.

For feature installation at a later time refer to ″Chapter 11: Installing Auto UNIX System DB2 and Auto UNIX System MQSeries on a Running system″ in the manual *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

If you install the features now do the following:

1 Switch to the feature Tab in the Installation Dialog

2 Select the features you want to install

3 Click the Install button.

2.19 Installing Auto UNIX System from Tapes

The actual installation is started by clicking the **Install** push button on the Installation Dialog. You will be prompted for the Auto UNIX System installation tapes. Information about the progress of the installation is displayed on the Control Center. The last stage is stage 6 if you selected to install features, otherwise stage 5. After completion of the last stage the system is automatically IPLed and ready to use.

Note: Wait until the subsystem status for all subsystems that are listed in the Control Center window is **UP** as shown in Figure 14 on page 24.

In our case, the installation of all 34 3480 tapes took about three hours, but this can vary depending on your system.

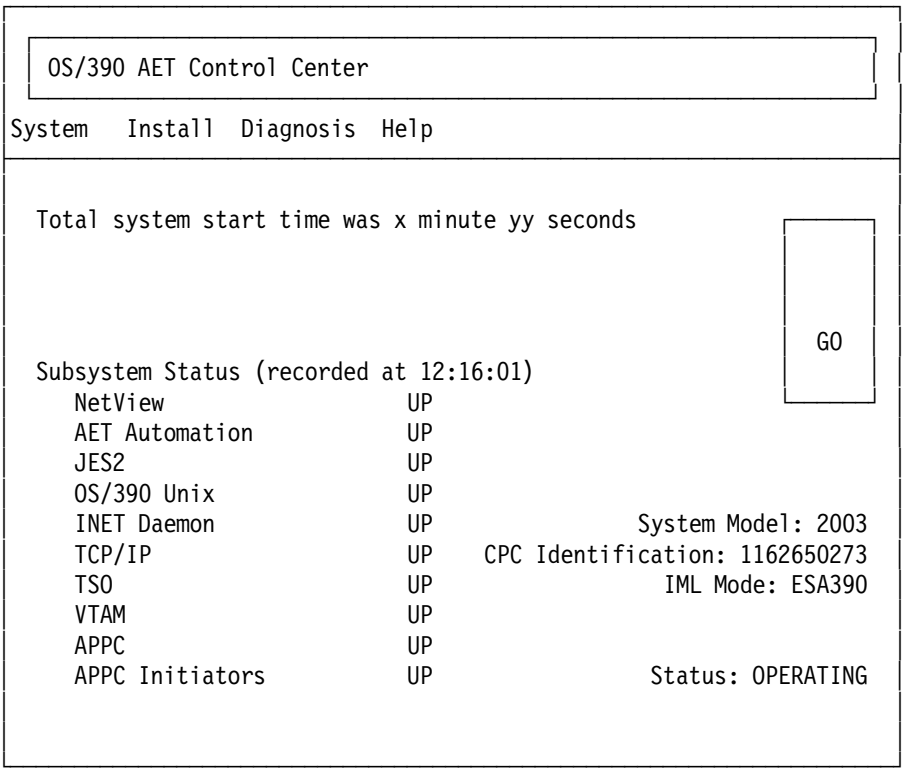


Figure 14. Subsystem Status

Continue with Customizing Auto UNIX System.

2.20 Initial Logon to Auto UNIX System

After installation completes, and the system comes up, you should check that you can log in to the system. Use **rlogin** or **telnet** to log in for the first time and change the administrator password.

The procedure below describes what you need to do in more detail:

1. Directly attach a workstation that supports TCP/IP **telnet** or **rlogin** to the same network segment as the Auto UNIX System S/390 processor is attached.
2. At this attached workstation, open a window and enter the **telnet** or **rlogin** command with the IP address for Auto UNIX System and the user ID EZMSUPR (administrator ID), for example: **telnet** *host IP*

   Where *host IP* is the address for Auto UNIX System, for example: **telnet 9.164.155.62**
3. At the prompt for the user ID enter **EZMSUPR**.
4. At the prompt for the password, enter **EZMSUPR**, and at the prompt for a new password enter a new password as shown on Figure 15 on page 25.

```
   EZMSUP's Password: ezmsupr
   Password expired
   Enter a new password: ******
   Re-Enter new password:
     .
     .
     .
     .
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*             OS/390 Application Enabling Technology             *
*                                                                *
*                      Powered by S/390(R)                       *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

   EZMSUPR /etc/profile active
   EZMSUPR:/:>
```

Figure 15. Logging in Example

Note: Do not forget your new password. **Write it down**

If the **rlogin** or **telnet** fails do a **ping** to the host and check:

- Is the workstation TCP/IP access set up correctly?
- Is the host IP address correct?
- If there is a router between a workstation and the host, did you enter the right router IP address? Try connecting without routers.

Note: If you entered the wrong host IP address or router during installation, you may have to re-install Auto UNIX System with the correct IP addresses. Temporarily configuring another workstation with the suitable IP address may also help to fix this problem.

To continue with tailoring the Auto UNIX System refer to manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993 Chapter 3. Getting started on Auto UNIX System.

For how to set up your TCP/IP refer to *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993, Chapter 12.

Chapter 3. VSE/ESA Setup for Auto UNIX System

This chapter will describe what we did to our VSE/ESA system after Auto UNIX System installation, before we could begin to use it.

3.1 Connecting Auto UNIX System with VSE/ESA

The Auto UNIX System installation is described in Chapter 2, “Installing Auto UNIX System” on page 7. The configuration for our redbook is described in Figure 16. We installed Auto UNIX System as a VM/ESA guest system. The user ID of the guest machine was COEXAET. The VSE/ESA system was also installed as a VM/ESA guest system. The user ID of the VSE/ESA guest machine was COEXVSE. The TCP/IP network was defined and controlled by the VM/ESA TCP/IP system. The user ID of the VM/ESA TCP/IP system was TCPIP.

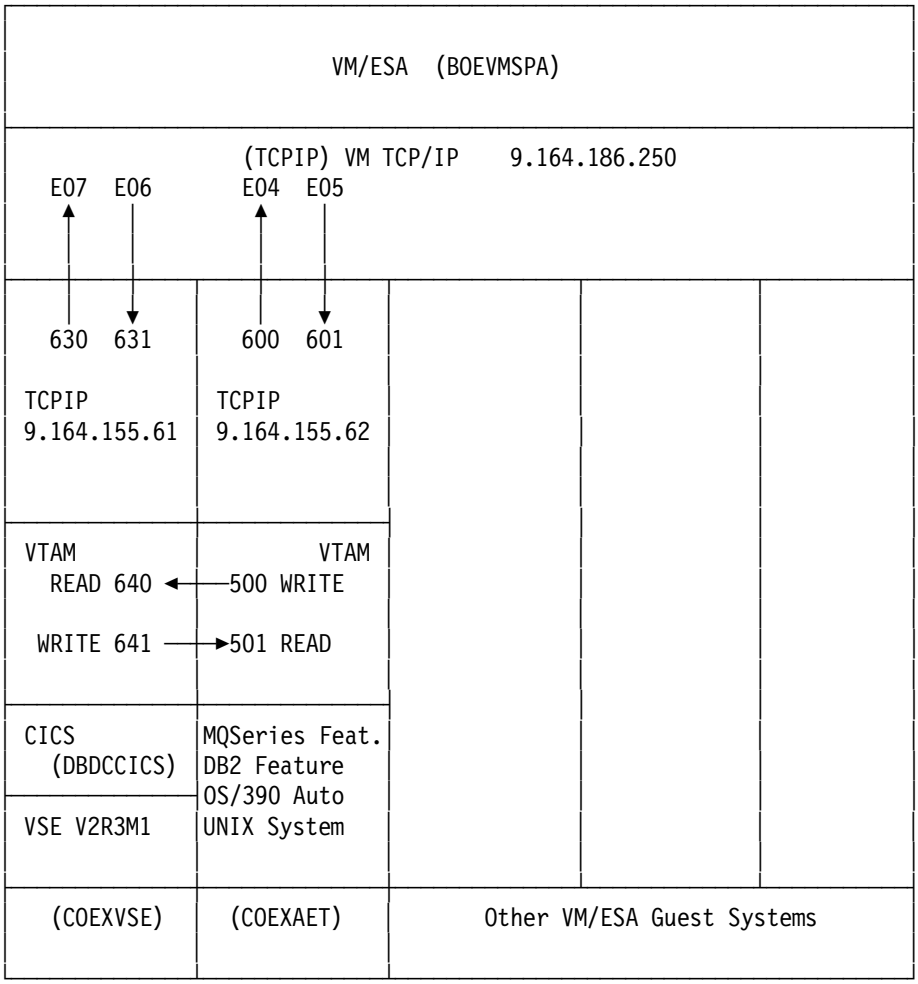


Figure 16. Software System Configuration

The channel-to-channel (CTC) communications path included in Auto UNIX System provides a high speed, proprietary channel path between two S/390 systems. In Figure 16 we have illustrated the CTC channels set up for Auto UNIX System. The CTC connections defined for our Auto UNIX System were a TCP/IP connection to the VM/ESA TCP/IP guest machine (TCPIP), and a VTAM

connection to our VSE/ESA guest machine (COEXVSE). We also defined a CTC connection between our VSE/ESA guest and the VM/ESA TCP/IP system to be used by our VSE/ESA TCP/IP partition.

We have included three figures in Appendix B, “Figures and Samples for VSE/ESA Setup” on page 111. These figures are an attempt to diagram and cross reference the parameters required in the definition and configuration of our test system.

Figure 51 on page 111 describes the virtual channel-to-channel definitions and VM COUPLE commands to connect the VM/ESA virtual guest machines.

Figure 52 on page 112 shows the relationships between the:

- VSE/ESA IPL procedure
- TCP/IP initialization member (IPINIT01.L)
- VTAM application IDs to be used for virtual terminal LU names
- TCP/IP definitions on the VM/ESA system.

Figure 53 on page 113 identifies the connections between the:

- VSE/ESA IPL procedure
- VTAM major nodes used for APPN
- CICS resource definition

to connect VSE/ESA to Auto UNIX System. On the Auto UNIX System side, the channel-to-channel addresses are defined during the initial installation. The Auto UNIX System APPN partner LU name is predefined as EZM01LUA and can not be changed.

The configuration allows for communication directly between VSE/ESA VTAM and Auto UNIX System. It also allows for communication between the VM/ESA TCP/IP guest system and both VSE/ESA and Auto UNIX System.

You need VSE/ESA 2.3 or higher to communicate with Auto UNIX System.

To connect the VSE/ESA system to Auto UNIX System, you need to:

1. Install TCP/IP for VSE/ESA.
2. Set up the virtual channel-to-channel connections.
3. Configure the VSE/ESA system.

3.2 Update the VSE/ESA IPL Procedure

Note: Before you start configuration, make sure that all your involved software is at the latest service level.

In our VSE/ESA configuration we defined four channel-to-channel address. Each connection requires a pair of channel addresses. We defined a TCP/IP connection to VM/ESA TCP/IP with address 630 and 631. We defined a VTAM connection to Auto UNIX System with address 640 and 641.

```
ADD 630:631,CTCA,EML      TCPIP connection
ADD 640:641,CTCA,EML      VTAM  connection
```

Note: The EML option is required to indicate that the device type as specified by the user should be used to assign the VSE device type code. The EML operand causes IPL to ignore device type sensing, and add the device as the type specified in the ADD command.

For our test configuration we defined a new VSE/ESA sublibrary (PRD2.UNIX). The purpose of this sublibrary was to contain any members changed or created for this test. The sample job to create this library is listed in Figure 54 on page 114. A new VTAM startup job was created with this library in the LIBDEF chain. A new CICS startup job was created which also included this new library in the LIBDEF chain. The new VTAM and CICS startup jobs are listed in Figure 55 on page 114 and Figure 56 on page 115.

3.3 Communicating Using VSE/ESA TCP/IP

To communicate with the VM/ESA TCP/IP system we needed to adapt the VSE/ESA TCP/IP configuration library member (IPINIT00.L). We changed the name of the standard member to IPINIT01.L and cataloged it into the PRD2.UNIX library.

The changed member is listed in Figure 62 on page 125. The changes we made are highlighted.

1 SET IPADDR contains the IP address of our VSE/ESA system.

2 SET MASK defined the sub-system mask for the TCP/IP configuration.

3 SET GATEWAY = OFF was set since the VSE system was not being used as a gateway.

4 We defined the standard translation table.

5 We defined a name for Auto UNIX System to make it easier to test.

6 We defined the LINK and ROUTE for our connection to VM/ESA TCP/IP. The VM/ESA TCP/IP system was defined as a gateway. All IP address resolution was handled by VM/ESA TCP/IP.

7 We defined the NFSD to activate NFS. This definition should follow all DEFINE FILE and DEFINE FILESYS definitions.

The sample TCPSTART job stream from the PRD1.BASE library was modified to include the changed configuration member IPINIT01. The sample TCPSTART is listed in Figure 63 on page 128.

3.4 Verify the TCP/IP Communication

To verify the communication, you must:

⇒ Modify the VSE IPL procedure

⇒ Define and couple the VM CTCA definitions

⇒ Re-IPL your VSE/ESA system

The CTCA definitions will now be active and available. Submit and release the modified TCPSTART job. The job is set up to run in VSE/ESA partition F7. Check the console log for the following messages:

IPN109I Initialization has completed.

IPL381I Start Complete CTC Adapter, Cuu:0630

These messages will indicate TCP/IP is available and communicating. To test the connection issue the PING command from the VSE/ESA console. For example:

At the VSE console enter **MSG F7**

The TCP/IP system will respond:

F7-00nn IPN300I Enter TCP/IP Command

Respond to this message with:

nn PING 9.164.186.250 (where the IP address is defined as a valid address on your network)

The output should look like this:

```
msg f7
AR 0015 1I40I  READY
F7-0092 IPN300I Enter TCP/IP Command
92 ping 9.164.186.250
F7 0090 TCP910I  Client manager connection Established.
F7 0090 TCP910I  PING
F7 0090 TCP910I  PING Ready:
F7 0090 TCP910I  SET HOST= 9.164.186.250
F7-0092 IPN300I Enter TCP/IP Command
F7 0090 TCP910I  PING Ready:
F7 0090 TCP910I  PING
F7 0090 TCP910I  PING 1 was successful, milliseconds:0000
F7 0090 TCP910I  PING 2 was successful, milliseconds:0000
F7 0090 TCP910I  PING 3 was successful, milliseconds:0000
F7 0090 TCP910I  PING 4 was successful, milliseconds:0000
F7 0090 TCP910I  PING 5 was successful, milliseconds:0000
F7 0090 TCP910I  PING Ready:
F7 0090 TCP910I  QUIT
```

Figure 17. Screen after Successful PING

Note: This must be successful before you proceed. If the PING command does not work, then none of the other functions will work either.

3.5 Configuring VSE/ESA VTAM

3.5.1 Prerequisites

The net ID of your VSE/ESA system and Auto UNIX System must be different. You can use a net ID of your choice other than EZMNET, which is reserved by Auto UNIX System.

3.5.2 APPN

APPN Host-to-Host Connection (AHHC). Auto UNIX System is pre-configured as an APPN network node. If your VSE/ESA node has no APPN capabilities, then you must change the VSE/ESA VTAM configuration to support APPN.

If you wish to permit access to your VSE/ESA applications via TELNET-TN3270, you must define one or more VTAM application IDs to be used as virtual terminal LU names.

The sample job included with VSE/ESA in ICCF library 59 (SKAPPN) listed in Figure 64 on page 129 was submitted. This job defines the VTAM clusters used for APPN. The cluster names are:

```
VTAM.DSDBCTL       NAME:   DSDBCTL
VTAM.DSDB1         NAME:   DSDB1
VTAM.DSDB2         NAME:   DSDB2
VTAM.TRSDB         NAME:   TRSDB
```

The required Class of Service (COS) table is copied to PRD2.CONFIG.

VSE/ESA provides a sample ″TCPAPPL.B″ member in library PRD1.BASE. This member was changed to include two additional definitions to match the sample TCP/IP definitions. The new member is listed in Figure 65 on page 130.

We need to update the VTAM startup list (ATCSTR00) with the APPN definitions. Refer to Figure 66 on page 131 for a list of the changes we made. We added the NODETYPE=NN keyword. The SSCPID, and SSCPNAME values are your choice. The NETID value should not be EZMNET, anything else is OK. APPNCOS, CONNTYPE, INITDB, SORDER, are the IBM defaults, but were added for documentation purposes.

Auto UNIX System supplies two samples, EZMVSV03 and EZMVSV04. These samples are listed in Figure 67 on page 132 and Figure 68 on page 132. These samples are required to define a VTAM connection to Auto UNIX System. The EZMVSV03 sample defines the AHHC major node. The EZMVSV04 sample defines the Transport Resource List (TRL) major node. Make sure the channel addresses ″640″ and ″641″ match the channel addresses you added to your IPL procedure. Make sure that the even channel (*write* channel) on the Auto UNIX System side is connected to the *read* channel on the VSE side, and vice versa.

Finally the ATCCON00 member listed in Figure 69 on page 133 was updated to include the TCPAPPL, VTMTRL, and VTMAHHC startup definitions. It is important that VTMTRL is activated before VTMSNA.

Make the VTAM configuration changes and submit the VSE/ESA VTAMUNIX job, see Figure 55 on page 114 for an example of a VTAM startup job. The VSE/ESA console messages at startup time should appear so:

```
F3 0003 // JOB VTAMUNIX START UP VTAM WITH AUTO UNIX CONNECTION
        DATE 11/18/1998, CLOCK 15/15/18
F3 0003 IST1497I VTAM FUNCTIONAL SUPPORT LEVEL IS MULTIDOMAIN
F3 0003 IST093I ISTCDRDY ACTIVE
F3 0003 IST315I VTAM INTERNAL TRACE ACTIVE - MODE = INT, SIZE = 050
F3 0003 IST199I OPTIONS = NONE
F3 0003 IST314I END
F3 0003 IST093I COSAPPN ACTIVE
F3 0003 IST093I ISTDSWMN ACTIVE
F3 0003 IST984I USER EXIT ISTEXCSD IS ACTIVE
F3 0003 IST093I TCPAPPL ACTIVE
F3 0003 IST093I VTMAPPL ACTIVE
F3 0003 IST093I ISTTRL ACTIVE
F3 0003 IST093I VTMAHHC ACTIVE
F3 0003 IST093I VTMSNA ACTIVE
F3 0003 IST1086I APPN CONNECTION FOR EZMNET.EZSSCP01 IS ACTIVE - TGN = 21
F3 0003 IST093I CAHHC1 ACTIVE
F3 0003 IST1096I CP-CP SESSIONS WITH EZMNET.EZSSCP01 ACTIVATED
F3 0003 IST093I VTMNSNA ACTIVE
F3 0003 IST093I VTMCTCA ACTIVE
F3 0003 IST093I VTMPATH ACTIVE
F3 0003 IST093I VTMPATH ACTIVE
F3 0003 IST093I VTMCA1 ACTIVE
F3 0003 IST093I VTMCA2 ACTIVE
F3 0003 IST093I VTMCA3 ACTIVE
F3 0003 IST1115I CDRM NAME ACFVTAM IS DIFFERENT THAN SSCPNAME START OPTIO
F3 0003 IST1116I SSCP NAME SSCP01 IS USED
F3 0003 IST093I VTMCDRM ACTIVE
F3 0003 IST093I VTMCDRS ACTIVE
F3 0003 IST093I VTMSW1 ACTIVE
F3 0003 IST020I VTAM INITIALIZATION COMPLETE FOR V4R2
F3 0003 IST1349I COMPONENT ID IS 5686-065O1-FE6
F3 0003 IST1348I VTAM STARTED AS INTERCHANGE NODE
```

Figure 18. Screen after VSE/ESA VTAM Startup

The two most important messages are:

IST1086I APPN CONNECTION FOR EZMNET.EZSSCP01 IS ACTIVE - TGN = 21

IST1096I CP-CP SESSIONS WITH EZMNET.EZSSCP01 ACTIVATED

These two messages show the definitions and connections between VSE/ESA and Auto UNIX System are working. This must be successful before you proceed. If the CP-CP sessions are not activated, then none of the other functions will work either.

3.6 Configuring CICS

After you have both the TCP/IP and VTAM subsystems defined and the connections are active, it is time to configure CICS.

Auto UNIX System provides sample CICS definitions and programs to verify the CICS configuration.

We added ISC=YES to the CICSUNIX job stream defined in Figure 56 on page 115.

EZMVSC07 supplied with Auto UNIX System, described in Figure 57 on page 117, defines connections and sessions necessary for CICS to communicate

with Auto UNIX System using the VTAM APPN connections. Also included with Auto UNIX System, are a sample CICS program and the required DFHCSD definitions to enable the program. The program and definition members are described in Figure 58 on page 118 and Figure 59 on page 119. Procedures to run these samples on your VSE/ESA system are described in the manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993.

We have included two additional samples, AUXCSD and AUXCOB. AUXCSD described in Figure 60 on page 122 defines an additional sample program and transaction. It also adds the TCPIP group to the standard CICS startup group list. The sample CICS program supplied with Auto UNIX System is written in C/VSE.

We converted this C/VSE sample into a COBOL/VSE sample. The COBOL sample is listed in Figure 61 on page 123. The sample is very simple. It receives a message from Auto UNIX System, writes that message to the VSE console and then sends a reply back to Auto UNIX System. The program could be changed to perform other CICS functions, such as reading a file or database. It could also call or transfer control to any other CICS application. It is a good sample to get you started with APPC communications between CICS and Auto UNIX System.

3.7 Verify the CICS Setup

Before starting the sample, you can verify the connection between VSE/ESA and Auto UNIX System.

After CICS is started, from the VSE/ESA console, issue the commands as shown in Figure 19.

```
msg f2
AR 0015 1I40I  READY
F2-0002
2 cemt i conn
F2 0002
F2 0002    Conn(CPIC) Net(EZM01LUA)     Ins Acq
F2 0002  RESPONSE: NORMAL TIME:  11.29.30  DATE: 11.19.98
F2 0002  APPLID=DBDCCICS
```

Figure 19. Screen after CEMT Command

Note: If the keywords **Ins** (for in service) and **Acq** (for acquired) are missing, it means the connection between VSE CICS and Auto UNIX System is not in service or acquired.

The instructions to set up and run the sample program are described in the manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993. The new VSE/ESA CICS COBOL program requires no changes to the Auto UNIX System sample program.

3.7.1 Running the APPC Sample

Follow the instructions to set up and run the Auto UNIX System APPC sample program as described in the Auto UNIX System documentation, *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993. After this is successful, you may run the VSE/COBOL sample described in Figure 61 on page 123. This is a CICS COBOL program which must be compiled and cataloged into the PRD2.UNIX library. The required transaction and program

definitions for CICS are shown in Figure 60 on page 122. After the sample program has been compiled and the CICS resources are defined, you may run this COBOL sample.

On the UNIX shell, in your private directory where you compiled the APPC sample (for example **/u/myuser/appc**), enter:

hello VTAM1 DBDCCICS AUXX

where:

VTAM1 is the NETID of the VTAM of your VSE/ESA operating system.

DBDCCICS is the partner LU name (APPLID) of your VSE/ESA CICS.

AUXX is the name of the COBOL sample transaction id.

If everything has been set up correctly the Auto UNIX System part of the sample writes on the UNIX shell:

```
STAN:/u/stan/appc:>hello VTAM1 DBDCCICS AUXX

Hello world ! was sent to the partner.

Received from Partner: HELLO AUTO UNIX !
```

Figure 20. Output from CICS COBOL Sample

The VSE/ESA counterpart writes on the VSE console:

```
 F2 0002 Hello world !

 F2 0002 HELLO AUTO UNIX !        HAS BEEN SENT BACK TO PARTNER.
```

Figure 21. Output from CICS COBOL on VSE Console

Chapter 4. Auto UNIX System and VSE/ESA FTP

4.1 Using FTP Between VSE/ESA, OS/2 and Auto UNIX System

In this chapter we will show how to transfer files with FTP between VSE/ESA, OS/2 and Auto UNIX System.

4.1.1 Transferring a Job from OS/2 FTP

Today, nearly any system is capable of running FTP, so what we did on OS/2 will probably work with other operating systems that have FTP capabilities.

First, we will show that it is possible to have a job sent via FTP to the POWER RDR Queue. We will send a simple DITTO DVT job just to prove that it works. This sample has nothing really to do with Auto UNIX System, but is included to show that it works independent of the FTP platform.

This is our job:

```
* $$ JOB JNM=DVTDOSRS,CLASS=O,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB DVTDOSRS
// OPTION LOG
// UPSI 1
// ASSGN SYS001,DISK,VOL=DOSRES,SHR
// EXEC DITTO
$$DITTO DVT INPUT=SYS001,SORTBY=EXTENT
$$DITTO EOJ
/*
/&
* $$ EOJ
```

Note: We have only one TCP/IP partition running on our VSE. It was set up with ID=00 in the TCPIP startup job, with the
// EXEC IPNET,...PARM='...,ID=00,...' statement.
Since ID=00 is the default ID, it has not been described in the commands, but it is worth mentioning, because if it is set differently in the startup it will always be necessary to specify the correct ID in the commands. This is because we can have several TCP/IP jobs running concurrently in one VSE.

Before transferring the job to the reader queue, the VSE console looked like Figure 22 on page 36.

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                      USER:  SYS
                                                                TIME:  13:46:27
 F2-0002
 d rdr,0
 AR 0015 1C39I COMMAND PASSED TO VSE/POWER
 F1 0001 1R46I  READER QUEUE   P D C S  CARDS
 F1 0001 1R46I  PRTDUMPA 00036 3 L 0        6 FROM=(SYSA)
 F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)









  ==>

 1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD         12=RTRV

 ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 22. VSE Console before Transferring Job to Reader Queue

First start your FTP session on an OS/2 window:

```
[C:\]ftp 9.164.155.61 ◄──────────────────────────── the COEXVSE node FTP Daemon
IBM TCP/IP for OS/2 - FTP Client ver 08:36:08 on Jul 22 1996
Connected to 9.164.155.61.
220-TCP/IP for VSE -- Version 01.03.00 -- FTP Daemon
    Copyright (c) 1995,1997 Connectivity Systems Incorporated
220 Service ready for new user.
Name (9.164.155.61): sysa
331 User name okay, need password.
Password: nnnnnn
230 User logged in, proceed.
ftp> cd power.rdr ◄──────────────────────────────── change to the directory to
                                                   where you want your file sent
                                              (POWER.RDR is treated as a directory)
250 Requested file action okay, completed.
ftp> put c:\docs\dvtdosrs.job anywilldo ◄──────────── observe the remote file name,
200 Command okay.                              anywilldo: you can specify anything,
150-File: POWER.RDR.0.ANYWILLDO                since it makes no sense to POWER.
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:      80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
Sent 244 bytes
226-Bytes sent:             244
    Records sent:            12
    Transfer Seconds:         2.36 (       K/Sec)
    File I/O Seconds:          .56 (      0K/Sec)
226 Closing data connection.
local: c:\docs\dvtdosrs.job remote: anywilldo
244 bytes sent in 0.01 seconds (23 Kbytes/s)
ftp>
```

After submitting the job, the console looked like Figure 23 on page 37.

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                        USER:  SYS

                                                                 TIME:  14:09:05
  F2-0002
  d rdr,0
  AR 0015 1C39I COMMAND PASSED TO VSE/POWER
  F1 0001 1R46I  READER QUEUE  P D C S  CARDS
  F1 0001 1R46I  PRTDUMPA 00036 3 L 0        6 FROM=(SYSA)
  F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)
1 F7 0090 001A: FTP934I FTP Session Established with:SYSA             from
2 F7 0090 Ipaddr:9.164.179.32 Id:FTP11 Port:21
3 F7 0090 001A: FTP936I FTP Daemon Storing File, Count: 12 Userid: SYSA
4 F7 0090 File: POWER.RDR.0.ANYWILLDO
  d rdr,0
  AR 0015 1C39I COMMAND PASSED TO VSE/POWER
  F1 0001 1R46I  READER QUEUE   P D C S  CARDS
  F1 0001 1R46I  PRTDUMPA 00036 3 L 0        6 FROM=(SYSA)
  F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)
5 F1 0001 1R46I  DVTDOSRS 00937 3 L 0       10 FROM=(SYSTCPIP)



  ==>

  1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD         12=RTRV

  ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 23. Console after Submitting the Job

Note: 1 and 2 indicate the start of the FTP session (9.164.179.32 was the host that started the session).
3 and 4 are the actual transfer.
5 here you can see the job as it arrived in the RDR queue.

4.1.2 FTP from Auto UNIX System

Basically we did the same job submission as before, but this time from Auto UNIX System. Before starting the transfer the VSE console looked like Figure 24.

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                       USER:  SYS
                                                                TIME:  15:19:32
F2-0002
d rdr,0
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R46I  READER QUEUE   P D C S  CARDS
F1 0001 1R46I  PRTDUMPA 00036 3 L 0        6 FROM=(SYSA)
F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)




==>

1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD          12=RTRV

ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 24. VSE Console before Transfer

This is the job that we sent to the POWER RDR queue from Auto UNIX System:

```
* $$ JOB JNM=DVTDOSRS,CLASS=O,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB DVTDOSRS
// OPTION LOG
// UPSI 1
// ASSGN SYSOO1,DISK,VOL=DOSRES,SHR
// EXEC DITTO
$$DITTO DVT INPUT=SYSOO1,SORTBY=EXTENT
$$DITTO EOJ
/*
/&
* $$ EOJ
```

using the following command:

[C:]**telnet 9.164.155.62**

After signing on to the Auto UNIX System system, 9.164.155.62, with our user ID, the screen looked like Figure 25.

```
IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 1998
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*             OS/390 Application Enabling Technology             *
*                                                                *
*                      Powered by S/390(R)                       *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*



EZMSUPR /etc/profile active
EZMSUPR:/:>
```

Figure 25. Screen after Sign on to Auto UNIX System

Now you must enter the FTP connection command:

```
EZMSUPR:/:>ftp 9.164.155.61
IBM FTP CS V2R6  1998 113 03:1
Connecting to:  9.164.155.61 p
220-TCP/IP for VSE -- Version
    Copyright (c) 1995,1997 Co
220 Service ready for new user
NAME (9.164.155.61:EZMSUPR):
sysa <------------------------------------ enter your user ID on the VSE system
>>> USER sysa
331 User name okay, need password.
PASSWORD:
nnnnnn <----------------------------------- enter your password

>>> PASS
230 User logged in, proceed.
Command:
cd power.rdr
>>> CWD power.rdr
250 Requested file action okay, completed.
Command:
put /u/jobs/dvtdosrs.job anywilldo <-------- the destination filename
>>> PORT 9,164,155,62,4,18                   makes no sense to POWER
200 Command okay.
>>> STOR anywilldo
150-File: POWER.RDR.ANYWILLDO <------------- name does not show up in POWER
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:      80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
226-Bytes sent:             244
    Records sent:            12
    Transfer Seconds:          .59 (       K/Sec)
    File I/O Seconds:          .01 (      0K/Sec)
226 Closing data connection.
244 bytes transferred in 0.005 seconds.  Transfer rate 48.80 Kbytes/sec.
Command:
```

On the VSE side, the console looked like Figure 26 on page 40.

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                        USER:  SYS
                                                                 TIME:  15:45:21
    d rdr,0
    AR 0015 1C39I COMMAND PASSED TO VSE/POWER
    F1 0001 1R46I  READER QUEUE   P D C S  CARDS
    F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)
 1  F7 0090 001A: FTP900I Daemon Startup FTP Id:FTP11 Port:21
 2  F7 0090 001A: FTP934I FTP Session Established with:SYSA             from
 3  F7 0090 Ipaddr:9.164.155.62 Id:FTP11 Port:21
 4  F7 0090 001A: FTP936I FTP Daemon Storing File, Count: 9 Userid: SYSA
 5  F7 0090 File: POWER.RDR.ANYWILLDO
    d rdr,0
    AR 0015 1C39I COMMAND PASSED TO VSE/POWER
    F1 0001 1R46I  READER QUEUE   P D C S  CARDS
    F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)
    F1 0001 1R46I  PAUSEBG  00957 3 L 0        0 FROM=(SYSTCPIP)
 6  F1 0001 1R46I  DVTDOSRS 00994 3 L 0       10 FROM=(SYSTCPIP)
   ==>

1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD         12=RTRV

   ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 26. VSE Console

Note: **1** and **2** are the start of the FTP session coming in from Auto UNIX System.
3 9.164.155.62 is the Auto UNIX System IP address.
4 and **5** are the actual file transfer.
6 here you can see the job as it arrived in the RDR queue.

The execution of the job is shown in Figure 27.

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                        USER:  SYS
                                                                 TIME:  15:45:21
 BG 0001 1Q47I   BG DVTDOSRS 00994 FROM (SYSTCPIP) , TIME=15:45:22
BG 0000 // JOB DVTDOSRS
        DATE 11/17/1998, CLOCK 15/45/22
BG 0000 EOJ DVTDOSRS  MAX.RETURN CODE=0000
        DATE 11/17/1998, CLOCK 15/45/22, DURATION   00/00/00
BG 0001 1Q34I   BG WAITING FOR WORK




==>

1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD         12=RTRV

ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 27. VSE Console

That is it. Seems to run OK. But we can check if it did what we expected. It is also possible to get reports out of the POWER LST Queue. So let us fetch our report back to Auto UNIX System and examine its output.

In order to do that, we go back to the FTP environment in Auto UNIX System, and do the following:

```
Command: ◄──────────────────────────────────── FTP prompt
cd / ◄──────────────────────────────────── go to the root, if not there
>>> CWD /
200 Command okay. Operating in Unix simulation mode.
Command:
cd power.lst.a ◄──────────────────────────── change directory to POWER
>>> CWD power.lst.a
250 Requested file action okay, completed.
Command:
dir ◄──────────────────────────────────── check if it is really there
>>> PORT 9,164,155,62,4,31
200 Command okay.
>>> LIST
150 File status okay; about to open data connection
-rw-rw-rw-  1 vse  direct       66 NOV 18 09:49 AUXDFILE.00970.00
-rw-rw-rw-  1 vse  direct       65 NOV 18 09:49 DVTDOSRS.00994.00 ◄── here is the one
226 Closing data connection.
Command:
get dvtdosrs.00994.00 /u/jobs/results.txt ◄──── get it back to AutoUnix, into
>>> PORT 9,164,155,62,4,32                          results.txt file
200 Command okay.
>>> RETR dvtdosrs.00994.00
150-File: POWER/LST/A/DVTDOSRS.00994.00
    Type: Ascii  Recfm: V  Lrecl:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
226-Bytes sent:           4,841
    Records sent:            60
    Transfer Seconds:          .60 (      OK/Sec)
    File I/O Seconds:          .02 (      OK/Sec)
226 Closing data connection.
4841 bytes transferred in 1.540 seconds.  Transfer rate 3.14 Kbytes/sec.
Command:


quit ◄──────────────────────────────────────── if you feel like leaving
                                                        the FTP environment
>>> QUIT
221 Service closing control connection.
EZMSUPR:/:>cd /
EZMSUPR:/:>cd /u/jobs
EZMSUPR:/u/jobs:>ls ◄──────────────────────── list files, to check it is there
pausebg.job           dvtdosrs.job
results.txt ◄──────────────────────────────── looks correct


EZMSUPR:/u/jobs:>vi results.txt ◄──────────── let  us check it
```

The vi editor screen looked like Figure 28 on page 42.

```
1DITTO/ESA for VSE
                 11/18/1998 (1998-322) 09:50  Page 2
O$$DITTO DVT INPUT=SYS001,SORTBY=EXTENT
-* * * * Device 0150, VOLSER=DOSRES, 3380 with 885 cyls, 15 trks/cyl, 47968 byte
s/trk                                          * * * *
---- Data Set Name --- sorted by EXTENT ----- Ext     Begin-end    Reltrk,
 File  Volume  Created   Expires
 1...5...10...15...20...25...30...35...40.... seq  Cyl-hd   Cyl-hd       numtrks
 type  serial  YYYY.DDD  YYYY.DDD

 VSE.SYSRES.LIBRARY                             0     0  1    63 14      1,959
 SAM   DOSRES  1998.292  9999.365
 DOS.LABEL.FILE.FF0645119672.AREA1              0    64  0    66 14    960,45
 UND   DOSRES  1998.314  9999.366
 VSE.POWER.QUEUE.FILE                           0    67  0    67  4   1005,5
 DAM   DOSRES  1998.292  9999.366
   *** FREE EXTENT ***                          0    67  5    67 14   1010,10
 Z9999996.VSAMDSPC.TB139BD8.T1A4301F            0    68  0    75 14   1020,120
 VSAM  DOSRES  1998.292  9999.366
 Z9999992.VSAMDSPC.TB139BD8.T5E915CA            0    76  0   209 14   1140,2010
 VSAM  DOSRES  1998.292  9999.366
   *** FREE EXTENT ***                          0   210  0   210 10   3150,11
  *** VTOC EXTENT ***                           0   210 11   210 14   3161,4
```

Figure 28. vi Editor Screen

Notice that the printer Control Characters are returned with the file. Most important is that instead of CTLCHAR we get ASA Control Characters, thus the file is ready for printing.

What we did in this chapter was a fully interactive job submission and a report transferring back to Auto UNIX System.

But what most people probably will need is to have this type of routine done more frequently, and certainly running in batch. Well, let us talk about that in our next chapter.

Chapter 5. Auto UNIX System and VSE/ESA FTP - Batch

Besides the interactive way of transferring files by means of FTP, another facility is set up in TCP/IP for VSE/ESA, which allows for more automated file transfers. Many installations find it important to have, for example, some files created at the central site, usually running a VSE/ESA system, and then send them to remote sites, such as remote offices, plants, retail stores or customers. This is now available for VSE/ESA users. With the same tool we can also fetch files from remote machines once they are set as FTP Daemons.

In this chapter we will show how to transfer files with FTP among VSE/ESA, OS/2, VM/ESA and Auto UNIX System in batch mode.

5.1 Introduction

We will be exercising two scenarios here. The first one will deal with sending a file from VSE to other sites, and the second scenario will do the opposite of that: fetch files from different machines.

But, before we start, any files to be dealt with may be defined in the VSE TCP/IP initialization file, as in our IPINIT01, listed below.

We defined two files: one KSDS and one ESDS for our sample procedures (AUXFILK and AUXFILE).

```
// EXEC LIBR
ACC S=PRD2.UNIX
CATALOG IPINIT01.L                          REPLACE=YES
*-----------------------------------------*
*                                         *
*    The IPADDR and the subsystem MASK    *
*    are dependent on your local          *
*    TCP/IP network configuration.        *
*                                         *
*-----------------------------------------*
SET IPADDR   = 9.164.155.61
SET MASK     = 255.255.255.000
*
 .. etc..

*-----------------------------------------*
*        Setup the File System            *
*-----------------------------------------*
DEFINE FILESYS,LOCATION=SYSTEM,TYPE=PERM
DEFINE FILE,PUBLIC='VSESPUC',DLBL=VSESPUC,TYPE=VSAMCAT
DEFINE FILE,PUBLIC='AUXFILK',DLBL=AUXFILK,TYPE=KSDS
DEFINE FILE,PUBLIC='AUXFILE',DLBL=AUXFILE,TYPE=ESDS
*
 .. etc..
```

There is another way to specify files to be transferred, which does not use the predefined files in IPINITxx, thus enabling us to dynamically choose the file ids we need to send or receive. It is achieved by using the % sign in front of the file name, indicating the name is of a DLBL statement instead of the DEFINE FILE's name. We will talk about this in 5.3, "Autonomous FTP" on page 49.

5.2 Scenario One - Pushing a File from VSE to Auto UNIX System

Here we will be sending a file from Auto UNIX System to VSE, actually a job that defines and initializes a VSAM file and at the end sends it back to Auto UNIX System. The job is created at Auto UNIX System, in /u/jobs directory, with the name of get.vse.vsam.file, as shown below. It starts by defining a simple KSDS file with a two byte key field at offset 0. We used the well known 'vi' editor in UNIX, as shown in Figure 29.

```
All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*             OS/390 Application Enabling Technology                *
*                                                                   *
*                      Powered by S/390(R)                          *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

EZMSUPR /etc/profile active
EZMSUPR:/:>cd /u/jobs
EZMSUPR:/u/jobs:>ls
get.vse.vsam.file     results.txt
dvtdosrs.job          pausebg.job           vsamfile
EZMSUPR:/u/jobs:>vi get.vse.vsam.file
```

Figure 29. Using the vi Editor to Look at the Job we will Run in VSE

Here is the VSE job:

```
* $$ JOB JNM=VSETOAUS,CLASS=O,DISP=D
* $$ LST CLASS=A,DISP=D
* this job is sent from Auto UNIX to VSE, creates a file
* and sends the file back to Auto UNIX
// JOB VSETOAUS
// EXEC IDCAMS,SIZE=AUTO
 /*                                                        */
 /*  DELETE VSAM FILE                                      */
 /*                                                        */
     DELETE (AUTO.UNIX.KSDS) CL NOERASE PURGE  -
     CATALOG(VSESP.USER.CATALOG)
 /*                                                        */
 /*  DEFINE VSAM FILE                                      */
 /*                                                        */
 DEF CLUSTER(NAME(AUTO.UNIX.KSDS)              -
     VOL(SYSWK1 DOSRES)                        -
     RECORDS (100 100)                         -
     RECORDSIZE (0080 0080)                    -
     INDEXED                                   -
     KEYS(2 0)                                 -
     SHR(2))                                   -
     DATA (NAME (AUTO.UNIX.KSDS.D) CISZ(4096)) -
     INDEX (NAME (AUTO.UNIX.KSDS.I) CISZ(512)) -
     CATALOG(VSESP.USER.CATALOG)
/*
```

The job will also insert some data into the file, simulating a real file creation process:

```
*
*    INITIALIZE AUXFILK
*
// DLBL AUXFILK,'AUTO.UNIX.KSDS',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE   -
        (SYSIPT                 -
         ENVIRONMENT            -
          (RECORDFORMAT (FIXUNB) -
           BLOCKSIZE(80)         -
           RECORDSIZE (80)))     -
        OUTFILE (AUXFILK)
01 RECORD ONE
02 RECORD TWO
03 RECORD THREE
04 RECORD FOUR
05 RECORD FIVE
06 RECORD SIX
07 RECORD SEVEN
08 RECORD EIGHT
09 RECORD NINE
10 RECORD TEN
/*
```

And then... we want the job to send the file to Auto UNIX System, after processing at VSE. We will use this step to send the processed VSAM file:

```
*
*    SEND FILE TO AUTOUNIX SYSTEM
*
// EXEC FTP,PARM='IP=9.164.155.62,ID=00,RETRY=4,RETRYTIME=3000'
EZMSUPR ◄──────────────────────── user id at Auto UNIX System
AUS                               and password
WERN ◄──────────────────────────── user id at local VSE/ESA
WERPAS                            and password
CD /u/filetransf ◄──────────────── lower case
PUT AUXFILK AUXFILK.VSE ◄──────── AUXFILK is defined in IPINI01.L
QUIT                      AUXFILK.VSE is the destination filename
/*
/&
$$ EOJ
```

Note: Auto UNIX System is case sensitive, as all UNIX systems are, so your commands will fail if the case does not match!

Parameters RETRY and RETRYTIME are to be set at choice.

After editing the above job we then transferred it to VSE.

```
EZMSUPR:/u/jobs:>ftp 9.164.155.61
IBM FTP CS V2R6  1998 113 03:11 UTC
Connecting to:  9.164.155.61 port: 21.
220-TCP/IP for VSE -- Version 01.03.00 -- FTP Daemon
    Copyright (c) 1995,1998 Connectivity Systems Incorporated
220 Service ready for new user.
NAME (9.164.155.61:EZMSUPR):
wern
>>> USER wern
331 User name okay, need password.
PASSWORD: nnnnnn

>>> PASS
230 User logged in, proceed.
Command:
cd power.rdr
>>> CWD power.rdr
250 Requested file action okay, completed.
Command:
put /u/jobs/get.vse.vsam.file anywilldo
>>> PORT 9,164,155,62,4,14
200 Command okay.
>>> STOR anywilldo
150-File: POWER.RDR.ANYWILLDO
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:      80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:           3,494
    Records sent:            60
    Transfer Seconds:         1.70 (      3K/Sec)
    File I/O Seconds:          .01 (      0K/Sec)
226 Closing data connection.
```

```
3494 bytes transferred in 0.010 seconds.
Transfer rate 349.40 Kb/sec.
Command:
```

The VSE console is shown in Figure 30 and Figure 31 on page 48.

```
SYSTEM:  VSE/ESA             VSE/ESA 2.3                USER:  WERN
                                                      TIME:  16:04:14
F7 0085 0018: FTP900I Daemon Startup FTP Id:FTP02 Port:21
F7 0085 001B: FTP900I Daemon Startup FTP Id:FTP12 Port:21
F7 0085 001B: FTP934I FTP Session Established with:WERN    from
F7 0085 Ipaddr:9.164.155.62 Id:FTP12 Port:21
F7 0085 001B: FTP936I FTP Daemon Storing File, Count: 60 Userid: WERN
F7 0085 File: POWER.RDR.ANYWILLDO
BG 0001 1Q47I   BG VSETOAUS 01159 FROM (SYSTCPIP) , TIME=16:02:08
BG 0000 * this job is sent from Auto UNIX to VSE,
BG 0000 * it creates a file and sends it back to Auto UNIX
BG 0000 // JOB VSETOAUS
        DATE 11/20/1998, CLOCK 16/02/08
BG 0000 *
BG 0000 *    INITIALIZE AUXFILK
BG 0000 *
BG 0000 *
BG 0000 *    SEND FILE TO AUTOUNIX SYSTEM
BG 0000 *
BG 0092 FTP100I TCP/IP -- FTP Client -- Version 01.03.00(I), 10/19/98
BG 0092 FTP101I Copyright 1995,1998 (C) Connectivity Systems Inc.
BG 0092 FTP105I Processing the execution parm override
BG 0092 FTP109I IP Address is set to 9.164.155.62
BG 0092 FTP111I System ID is set to 00
BG 0092 FTP131I Retry Counter is set to 4
BG 0092 FTP132I Retry Timer is set to 3000
BG 0092 FTP106I Processing of parm override complete
==>

1=HLP 2=CPY 3=END      6=CNCL 7=BWD 8=FWD 9=EXPL 10=INP       12=INFO
```

Figure 30. First VSE Console Display

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                 USER:  WERN
                                                       TIME:  16:04:27
BG 0092 FTP106I Processing of parm override complete
BG 0092 FTP115I Connecting with TCP/IP for VSE partition
BG 0092 FTP117I Now connected with partition
F7 0085 0018: FTP934I FTP Session Established with:WERN    from
F7 0085 Ipaddr:9.164.155.61 Id:FTP02 Port:21
F7 0085 IPF300I I/O Handler Startup VSAM KSDS
F7 0085 0018: FTP936I FTP Daemon Retrieving File, Count: 10 Userid: WERN
F7 0085 File: AUXFILK
F7 0085 0018: FTP900I Daemon Startup FTP Id:FTP02 Port:21
BG 0092 FTP102I TCP/IP -- FTP Client -- Complete
BG 0092 FTP118I Terminating connection
BG 0092 FTP119I Connection is already closed
BG 0000 EOJ VSETOAUS  MAX.RETURN CODE=0000
        DATE 11/20/1998, CLOCK 16/02/19, DURATION   00/00/10
BG 0001 1Q34I   BG WAITING FOR WORK
==>

1=HLP 2=CPY 3=END      6=CNCL 7=BWD 8=FWD 9=EXPL 10=INP       12=INFO
```

Figure 31. Second VSE Console Display

Now we can check at the Auto UNIX System side if we got the file we wanted, just by using the 'vi' Auto UNIX System editor:

```
EZMSUPR:/:>cd /u/filetransf ◄──────────── change to the target directory
EZMSUPR:/u/filetransf:>ls ◄──────────────── list files
AUXFILK.VSE ◄──────────────────────────── here is our file
EZMSUPR:/u/filetransf:>
EZMSUPR:/u/filetransf:>vi auxfilk.vse ◄── note that this does not work
```

The reason why it did not work, is that we typed the name in lowercase and then we changed to UPPERCASE like this:

```
EZMSUPR:/u/filetransf:>vi AUXFILK.VSE
```

This is what the vi editor shows:

```
01 RECORD ONE
02 RECORD TWO
03 RECORD THREE
04 RECORD FOUR
05 RECORD FIVE
06 RECORD SIX
07 RECORD SEVEN
08 RECORD EIGHT
09 RECORD NINE
10 RECORD TEN
"AUXFILK.VSE" 10 lines, 810 characters
```

This confirms that the file was sent correctly from VSE to Auto UNIX System.

5.3 Autonomous FTP

Normally, the VSE FTP Daemon performs all file transfers, and therefore, all files must be defined to the TCP/IP partition.

Under some circumstances, this can be inconvenient. For example, when a batch process creates a new file which is to be sent to a remote workstation, several interactions with TCP/IP for VSE/ESA are required to define the new file.

In the beginning of this chapter we said that it is possible to dynamically choose the file to be sent or received. Using this method, the TCP/IP partition does not need to have advanced knowledge of the data sets to be transferred.

Rather than force operator intervention, an extension to the FTP commands is provided that will permit specification of a locally-defined DLBL, this facility is called Autonomous FTP.

To transfer a file in this mode, use a command of the following format:

```
PUT %dlbl,type[,recfm,lrecl,blksize]  filespec
GET filespec  %dlbl,type,[recfm,lrecl,blksize]
```

The percent sign (%) indicates that a DLBL has been supplied rather than a file name. The other parameters are as follows:

filespec The file name on the remote system, client.

type The file type. Supported values are SAM, ESDS, and KSDS.

recfm The file record format. Allowable values are F, FB, V, and VB.

lrecl The file logical record length. Not applicable to VSAM KSDS and ESDS.

blksize The file block size. N/A to VSAM KSDS and ESDS.

The file is transferred by the FTP Daemon executing in the TCP/IP for VSE/ESA partition. The file must be defined and accessible to the TCP/IP for VSE/ESA partition and not the batch client's partition **unless** the autonomous feature is used.

When a file is specified by local DLBL name, the TCP/IP for VSE/ESA partition dynamically allocates the file and creates a temporary file system entry. After the transfer operation, the temporary entry is deleted from the file system.

Remote user IDs and passwords are limited to a maximum of 32 characters each.

If we were to use this feature, called Autonomous FTP, we would not have to alter the initialization file, (refer to Figure 62 on page 125 for an example of an initialization file) and the FTP step would appear as shown below:

```
// DLBL DYNAME,'AUTO.UNIX.KSDS.DYN',0,VSAM,CAT=VSESPUC
// EXEC FTP,PARM='IP=9.164.155.62,ID=00,RETRY=4,RETRYTIME=3000'
                  ↑
                  └──── IP address of partner FTP at Auto UNIX
EZMSUPR ◄──────────── user id at partner node Auto UNIX
AUS ◄──────────────── password
WERN ◄─────────────── VSE user id
WERPAS ◄───────────── password
CD /u/filetransf ◄─── change directory to where to send the file
PUT %DYNAME,KSDS DYNAME.FILE.VSE
QUIT
/*
```

We believe that this is an interesting way to transfer files using FTP, since it does not require any reconfiguration to change file ids, and that might be the choice for many installations to deal with their daily operations. Of course then, those alterations in the initialization file IPINIT01 would not be necessary.

Note: If a file is listed in the initialization file, then it will not be eligible to be used by the Autonomous FTP Feature, since the file is already allocated and all we are likely to get is an OPEN ERROR. This is why in our example we used file AUTO.UNIX.KSDS.DYN instead of the predefined one AUTO.UNIX.KSDS.

Please consider that all we said about PUTting a file via Autonomous FTP works as well for GETting files. More information can be obtained in *TCP/IP for VSE/ESA User*, SC33-6601.

5.4 Scenario Two - Pulling Files from Other Machines

Here we will have our VSE system collecting files from different sources. We will pull files from two OS/2 workstations, from Auto UNIX System and from a VM/ESA user minidisk. All four files will be grouped together in one master file in VSE for further processing in the night shift, for example. The OS/2 workstations were also configured as FTP Daemons. The files were created using the text editor of each system and we identified in the records where they were created, so that once consolidated in VSE they could easily be identified as to their origin.

The master file in VSE is called DIR, we chose it to be an ESDS, since this type of file is best suited to allow multiple appending transfers. The remote office files can be accessed through FTP users specially defined for providing access from VSE to them and these users are named after our branch names: CARM, JLC, JDS and RTNH. CARM and JLC are OS/2 FTP Daemon users, JDS is the user in Auto UNIX System with rights over the files to be collected by VSE and RTNH is a VM user who creates the records in VM due to be transferred and processed in VSE.

Here is what they look like:

```
+++ RECORDS FROM JLC +++
JLC  01
JLC  02
JLC  03
JLC  04
JLC  05
JLC  06
JLC  07
JLC  08
JLC  09
JLC  10
+++ END JLC RECORDS +++
```

```
+++ RECORDS FROM CARM +++
CARM 01
CARM 02
CARM 03
CARM 04
CARM 05
CARM 06
CARM 07
CARM 08
CARM 09
CARM 10
+++ END CARM RECORDS +++
```

```
+++ RECORDS FROM JDS +++
JDS  01
JDS  02
JDS  03
JDS  04
JDS  05
JDS  06
JDS  07
JDS  08
JDS  09
JDS  10
+++ END JDS RECORDS +++
"jds_today.data" 12 lines, 129 characters
```

```
 RTNH     DATA     A1  F 80  Trunc=80 Size=12 Line=0 Col=1 Alt=16

      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..>
===== * * * Top of File * * *
===== +++ RECORDS FROM RTNH +++
===== RTNH 01
===== RTNH 02
===== RTNH 03
===== RTNH 04
===== RTNH 05
===== RTNH 06
===== RTNH 07
===== RTNH 08
===== RTNH 09
===== RTNH 10
===== +++ END RTNH RECORDS +++
===== * * * End of File * * *
```

The job we are going to use in order to get all four files we need from the remote offices is this:

```
* $$ JOB JNM=COLLECT,CLASS=O,DISP=D,DUETIME=hhmm,DUEDAY=DAILY
* $$ LST CLASS=A,DISP=D
// JOB COLLECT
* This job collects data from the branch offices into
* one ESDS file for further processing the consolidated data
// DLBL DIR0000,'DIR.INCOMING.FILES',0,VSAM,CAT=VSESPUC
// EXEC FTP,PARM='IP=9.164.155.62,ID=00,RETRY=4,RETRYTIME=3000'
                  ▲
                  └── IP address of partner FTP at Auto UNIX
jds ◄──────────── the user responsible for JDS's files
jdspas ◄────────── his password
WERN ◄──────────── VSE user id
WERPAS ◄────────── password
GET jds_today.data %DIR0000,ESDS
CLOSE ◄─────────── use this command to close the current connection
OPEN 9.164.179.32 ◄── use OPEN to set up another connection
carm ◄──────────── user and password on the remote FTP site
carmpas
GET carm.data %DIR0000,ESDS ◄──────────── fetch the file
CLOSE
OPEN 9.164.165.221 ◄──────────────────────── JLC site
jlc
jlcpas
GET jlc.data %DIR0000,ESDS ◄──────────── fetch the file
CLOSE
OPEN 9.164.186.250 ◄──────────────────────── RTNH site
rtnh
rtnhpas
GET rtnh.data %DIR0000,ESDS ◄──────────── fetch the file
QUIT
/*
/&
* $$ EOJ
```

Note: The first node we need to contact must be specified in the PARM='IP=.... of the // EXEC FTP statement and then the other FTP Daemons can be accessed through CLOSEing the current session and OPENing the next. The IP parameter is mandatory.

We wanted to check that the files were correctly transferred and did this with IDCAMS PRINT:

```
// JOB PRINT
* This job prints the contents of the collected
* data file from our branch offices
// DLBL INFILE,'DIR.INCOMING.FILES',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
IDCAMS  SYSTEM SERVICES

 PRINT INFILE ( INFILE ) -
          CHARACTER
IDCAMS  SYSTEM SERVICES
LISTING OF DATA SET -DIR.INCOMING.FILES
RBA OF RECORD - 0
+++ RECORDS FROM JDS +++
RBA OF RECORD - 80
JDS  01
  ...
  ...
RBA OF RECORD - 800
JDS  10
RBA OF RECORD - 880
+++ END RECORDS FROM JDS +++
RBA OF RECORD - 960
+++ RECORDS FROM CARM +++***
RBA OF RECORD - 1040
CARM 01
  ...
  ...
RBA OF RECORD - 1760
CARM 10
RBA OF RECORD - 1840
+++ END CARM RECORDS +++
RBA OF RECORD - 1920
+++ RECORDS FROM JLC +++
RBA OF RECORD - 2000
JLC  01
  ...
  ...
RBA OF RECORD - 2720
JLC  10
RBA OF RECORD - 2800
+++ END JLC RECORDS +++
RBA OF RECORD - 2880
+++ RECORDS FROM RTNH +++
RBA OF RECORD - 2960
RTNH 01
  ...
  ...
RBA OF RECORD - 3680
RTNH 10
RBA OF RECORD - 3760
+++ END RTNH RECORDS +++

IDC0005I NUMBER OF RECORDS PROCESSED WAS 48
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDCAMS  SYSTEM SERVICES

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
1S55I  LAST RETURN CODE WAS 0000
EOJ PRINT     MAX.RETURN CODE=0000
```

It is interesting to note that one single collecting step, as we did in our example, has performance advantage over several // EXEC FTP,PARM... runs. But be careful: If one of the FTP Daemons we try to reach is not active, then TCP/IP for VSE/ESA will flush the rest of the command stream, leaving us with an incomplete file transfer, resulting from that one non-OK node, and stopping the whole transfer. We can say that it stops at any error. We simulated this scenario: nodes Auto UNIX System, JLC, and RTNH are up and running, but CARM is shutdown. Let us see what happened when we tried to run our collecting job:

```
// JOB COLLECT                                          DATE 11/25/1998
* This job collects data from the branch offices into
* one ESDS file for further processing the consolidated data
// DLBL DIR0000,'DIR.INCOMING.FILES',0,VSAM,CAT=VSESPUC
// EXEC FTP,PARM='IP=9.164.155.62,ID=00,RETRY=4,RETRYTIME=3000'
11251998 131256.95 FTP100I TCP/IP FTP Client Version 01.03.00(I)
11251998 131256.95 FTP105I Processing the execution parm override
11251998 131256.95 FTP109I IP Address is set to 9.164.155.62
11251998 131256.95 FTP111I System ID is set to 00
11251998 131256.95 FTP131I Retry Counter is set to 4
11251998 131256.95 FTP132I Retry Timer is set to 3000
11251998 131256.95 FTP106I Processing of parm override complete
11251998 131256.97 FTP115I Connecting with TCP/IP for VSE partition
11251998 131257.07 FTP117I Now connected with partition
F: 220-USERA001 IBM FTP CS V2R6 at autounix1, 13:12:57 on 1998-11-25.
220 Connection will not timeout.
Enter Foreign User ID or "LOGOFF":
jds
F: USER jds
F: 331 Send password please.
Enter Foreign Password or "LOGOFF"

F: PASS XXXXXX
F: 230 JDS is logged on.  Working directory is "/u/jds".
Foreign host connection established.
L: 220-TCP/IP for VSE -- Version 01.03.00 -- FTP Daemon
    Copyright (c) 1995,1998 Connectivity Systems Incorporated
220 Service ready for new user.
Enter Local User ID, null, or "LOGOFF":
WERN
L: USER WERN
L: 331 User name okay, need password.
Enter Local Password, null, or "LOGOFF"

L: PASS XXXXXXX
L: 230 User logged in, proceed.
Local host connection established.
Ready:
BATCH
Ready:
GET jds_today.data %BGESDS,ESDS <------- pull file from JDS

L: PASV
L: 227 Entering Passive Mode (009,164,155,061,016,007).
F: PORT 009,164,155,061,016,007
F: 200 Port request OK.
L: STOR %BGESDS,ESDS
L: 150-File: Local.File.Definition
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:      80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
```

```
    Translate with US_ENG_01
150 File status okay; about to open data connection
F: RETR jds_today.data
F: 125 Sending data set /u/jds/jds_today.data
L: 226-Bytes sent:             146
    Records sent:             12
    Transfer Seconds:          1.47 (       K/Sec)
    File I/O Seconds:           .47 (      OK/Sec)
226 Closing data connection.
F: 250 Transfer completed successfully.
Ready:
CLOSE
Foreign host connection closed.

Ready:
OPEN 9.164.179.32 <---------------------- try to connect to CARM
Error: Could not connect with foreign host
11251998 131301.97 FTP124W Flushing the rest of system input
11251998 131301.98 FTP102I TCP/IP -- FTP Client -- Complete
11251998 131301.98 FTP118I Terminating connection
11251998 131301.99 FTP120I Connection has closed, successfully
1S55I  LAST RETURN CODE WAS 0004
EOJ COLLECT   MAX.RETURN CODE=0004
```

The collecting job stops running, not even trying the next command. If we need to be sure that we get the files from all sites that are up and running without the risk of breaking the normal file transfer flow, then we must run one // EXEC FTP step for each remote FTP Daemon. Using this technique, the failing nodes or hosts will be the only ones missing at the end of the job. We suggest to check for return codes in your jobs to be sure all files came in OK, and if not, send a message to someone who will take the actions best suited for your installation.

5.5 Final Considerations on VSE FTP

There are a few issues we would like to talk about before leaving FTP. From the point of view of VSE it is interesting to note the way things happen when receiving files sent from a client FTP.

Two kinds of VSAM files can be sent to VSE via FTP:

- KSDS files
- ESDS files

5.5.1 FTP to VSE KSDS Files

When sending records to a KSDS file, FTP will insert records. It never updates any record, so, if records come in whose keys already exist in the target file, FTP will immediately end the transfer at that point. On the other hand, as the file is KSDS then the records will be inserted in their true record key position in the file. We did three file transfers using FTP to VSE/ESA, into a KSDS file.

First file:

```
01 RECORD ONE
02 RECORD TWO
03 RECORD THREE
04 RECORD FOUR
05 RECORD FIVE
07 RECORD SEVEN
08 RECORD EIGHT
09 RECORD NINE
10 RECORD TEN
```

Note: We do not have record six.

After sending the file to VSE, we try to send another file to the same target:

Second file:

```
06 RECORD SIX,      INSERTED IN SECOND RUN
14 RECORD FOURTEEN, INSERTED IN SECOND RUN
13 RECORD THIRTEEN, INSERTED IN SECOND RUN
```

The resulting file, is shown by IDCAMS PRINT:

```
// EXEC IDCAMS,SIZE=AUTO
IDCAMS  SYSTEM SERVICES

 PRINT INFILE ( INFILE ) -
          CHARACTER
IDCAMS  SYSTEM SERVICES
LISTING OF DATA SET -AUTO.UNIX.KSDS
KEY OF RECORD - 01
01 RECORD ONE
KEY OF RECORD - 02
02 RECORD TWO
KEY OF RECORD - 03
03 RECORD THREE
KEY OF RECORD - 04
04 RECORD FOUR
KEY OF RECORD - 05
05 RECORD FIVE
KEY OF RECORD - 06
06 RECORD SIX,      INSERTED IN SECOND RUN
KEY OF RECORD - 07
07 RECORD SEVEN
KEY OF RECORD - 08
08 RECORD EIGHT
KEY OF RECORD - 09
09 RECORD NINE
KEY OF RECORD - 10
10 RECORD TEN
KEY OF RECORD - 13
13 RECORD THIRTEEN, INSERTED IN SECOND RUN
KEY OF RECORD - 14
14 RECORD FOURTEEN, INSERTED IN SECOND RUN
IDC0005I NUMBER OF RECORDS PROCESSED WAS 12
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDCAMS  SYSTEM SERVICES
```

We see that record six was inserted in place, as well as records 13 and 14. Then we transferred the third file into the same target, the following:

```
15 RECORD FIFTEEN, INSERTED IN STEP 3
16 RECORD SIXTEEN, INSERTED IN STEP 3
10 TRY TO UPDATE, SHOULD NOT WORK IF TARGET IS KSDS!
17 RECORD SEVENTEEN  WILL IT GO INTO THE FILE?
```

At the VSE console we saw that something went wrong:

```
F7 0085 001B: FTP936I FTP Daemon Storing File, Count: 4 Userid: WERN
F7 0085 File: AUXFILK
F7 0085 IPF307I VSAM KSDS File switching from Skip Sequential to Direct.
F7 0085 DLBL:AUXFILK
F7 0085 IPF306W VSAM KSDS File duplicate record, update ignored.
F7 0085 DLBL:AUXFILK
```

We printed the file to find out what happened, and saw that nothing was processed beyond the point of failure: Record key 10 already existed. Records 15 and 16 were inserted, but 17 was not.

```
// JOB PRINT
* This job prints the contents of the collected
* data file from our branch offices
// DLBL INFILE,'AUTO.UNIX.KSDS',,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
IDCAMS  SYSTEM SERVICES

 PRINT INFILE ( INFILE ) -
         CHARACTER
IDCAMS  SYSTEM SERVICES
LISTING OF DATA SET -AUTO.UNIX.KSDS
KEY OF RECORD - 01
01 RECORD ONE
KEY OF RECORD - 02
02 RECORD TWO
KEY OF RECORD - 03
03 RECORD THREE
KEY OF RECORD - 04
04 RECORD FOUR
KEY OF RECORD - 05
05 RECORD FIVE
KEY OF RECORD - 06
06 RECORD SIX,      INSERTED IN SECOND RUN
KEY OF RECORD - 07
07 RECORD SEVEN
KEY OF RECORD - 08
08 RECORD EIGHT
KEY OF RECORD - 09
09 RECORD NINE
KEY OF RECORD - 10
10 RECORD TEN
KEY OF RECORD - 13
13 RECORD THIRTEEN, INSERTED IN SECOND RUN
KEY OF RECORD - 14
14 RECORD FOURTEEN, INSERTED IN SECOND RUN
KEY OF RECORD - 15
15 RECORD FIFTEEN, INSERTED IN STEP 3
KEY OF RECORD - 16
16 RECORD SIXTEEN, INSERTED IN STEP 3
IDC0005I NUMBER OF RECORDS PROCESSED WAS 14
```

```
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDCAMS  SYSTEM SERVICES
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
1S55I  LAST RETURN CODE WAS 0000
EOJ PRINT     MAX.RETURN CODE=0000
```

5.5.2 FTP to VSE/ESA ESDS Files

When sending records to an ESDS file, FTP will always write them at the end of the file, since no insertions can be expected. This kind of file will expose you to a lower risk level, since your transfer will not be cancelled by duplicated records. On the other hand, using ESDS obviously does not prevent you at all from sending the same file twice.

5.5.3 Binary and Text Files

In our examples we have been doing file transfers of text files only, mainly because they are extremely easy to deal with, such as creating simple data with an editor or printing the files after they have been handled. It is worth noting that files transferred among different platforms such as VSE, Auto UNIX System and OS/2 must be translated from EBCDIC to ASCII in accordance with the transfer direction. This is done using translate tables available at TCP/IP for VSE/ESA. But, there are some files, on the other hand, that are not text files: they contain binary data which can not be translated whatsoever.

For those, FTP provides the means to transfer without translating. The option may be set at the file definition statement DEFINE FILE using the TRANSLATE=IMAGE option in the TCP/IP for VSE/ESA configuration file. If the translation is not defined there, probably because of Autonomous FTP, there is also the possibility of using the TYPE command, before starting a certain transfer. The TYPE will set the conversion mode from that point on. Its format is:

```
TYPE ASCII | EBCDIC [ NONPRINT | TELNET | CARRIAGE ]
TYPE BINARY
```

The BINARY option avoids any kind of changes to the files. But note, please, that files with mixed types of data are not suited for FTP, since either the records will be entirely translated or not at all. For this kind of file, site provided tools, such as file conversion utilities, will be the only solution to enable FTP use.

5.5.4 Hint

Make sure you are in the right directory before sending a file to VSE. Just to give you an example: if you had sent a file to the POWER RDR, and the next thing you want to do is to append some records to a VSAM file, if you do not **CD /**, before sending your records to the VSAM file, then your records will end up in the reader queue and may even try to run, instead of going into the intended VSAM target.

Chapter 6. Auto UNIX System and VM/ESA FTP

This chapter describes how to transfer files with the FTP (File Transfer Program) between VM/ESA and Auto UNIX System.

6.1 Prerequisites for Using FTP between Auto UNIX System and VM/ESA

To transfer files between VM/ESA and Auto UNIX System the following are needed:

1 On the VM/ESA side:

- Access to the TCP/IP File Transfer Program (FTP)

2 On the Auto UNIX System side:

- A user ID.
- A Hierarchical File System (HFS) for a user home directory.

Note: You must be a superuser to add a new user to an existing group or in addition create a new group for this user.

If you are not a superuser, ask your administrator to do the following steps:

1 Create a group (optional).

2 Create an HFS data set for the home directory you will specify during add user with the home=/x/yyyy parameter.

3 Add a user to the new group or an existing group.

For more detailed information refer to the manual: *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363.

6.2 Creating a New User Group and Adding a New User

Note: Because you can only create users for groups that already exist, first use the **mkgroup** command to create groups, then use the **mkuser** command to create users to connect to this group.

For example, assume that you want do define a user in a new, not yet defined group with the name ″JDSSTAFF″ and assign a GID to the group:

1 Login as superuser

2 Create group with:
mkgroup gid=0000111111 JDSSTAFF

Note: Do not use reserved GID numbers (0-100)

3 Add a user with:
mkuser gecos=′VM FTP′ home=/u/walter/ id=0000034103 pgrp=JDSSTAFF su=yes walter

Note: The user′s password is set by the system, equal to the user ID WALTER. The first time the user logs in, the system will request that the user supply a new password.

If the command was successful, do a: **lsuser walter**.

The output should look like Figure 32 on page 60.

```
USERID=WALTER
PGRP=JDSSTAFF
GECOS=VM FTP
SU=YES
GROUPS=JDSSTAFF
ID=0000034103
HOME=/u/walter
```

Figure 32. Display with lsuser Command

4 Create the user's home directory.

Note: This can be a directory in an existing HFS or a new HFS that you create for the user. The recommended method is to create an HFS for each user and mount them on the **/u** directory.

At the time you added the user, you decided that your home directory is named **/u/walter** therefore you have to create the directory and mount it. Assume you want to create an initial space of 10 000 256-byte blocks. Use the following commands to do this:

- **mkdir /u/walter**
- **mkds user.walter 10000**
- **ezmount user.walter /u/walter**

6.3 Transferring Data with FTP between Auto UNIX System and VM/ESA

The following description assumes you have VM/ESA 2.3 and TCP/IP V3.1.

To ensure that your TCP/IP connection from VM/ESA to Auto UNIX System is up and running:

- Login to Auto UNIX System.
- Use the **oping** command from the Auto UNIX System Workstation session to verify the connection, see Figure 33 on page 61.

```
OS/2      Ctrl+Esc = Window List       Type HELP = help

[C:\]telnet 9.164.155.62

EZYTE27I login: walter
EZYTE28I walter Password: ......
IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 1998
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*             OS/390 Application Enabling Technology            *
*                                                               *
*                       Powered by S/390(R)                     *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*



 WALTER /etc/profile active
WALTER:/u/walter:>oping -c 5 9.164.186.250
CS V2R6: Pinging host 9.164.186.250
Ping #1 response took 0.002 seconds.
Ping #2 response took 0.002 seconds.
Ping #3 response took 0.001 seconds.
Ping #4 response took 0.002 seconds.
Ping #5 response took 0.001 seconds.
WALTER:/u/walter:>
```

Figure 33. Display of the oping Command from an Auto UNIX System Workstation

1. At the WS login to FTP with the IP address of the router (VM/ESA TCP/IP IP address 9.164.186.250).
2. You will be prompted for user ID and password, see Figure 34 on page 62.

```
WALTER:/u/walter:>ftp 9.164.186.250
IBM FTP CS V2R6  1998 113 03:11 UTC
Connecting to:  9.164.186.250 port: 21.
220-FTPSERVE IBM VM Level at BOEVMSPA.boeblingen.de.ibm.co, 09:22:10 CET
12/11/98
220 Connection will close if idle for more than 5 minutes.
NAME (9.164.186.250:WALTER):
coex01
>>> USER coex01
331 Send password please.
PASSWORD:xxxxx

>>> PASS
230 COEX01 logged in; working directory = COEX01 191
Command:
get old.txt old.txt
>>> PORT 9,164,155,62,4,32
200 Port request OK.
>>> RETR old.txt
150 Sending file 'old.txt' FIXrecfm           80
250 Transfer completed successfully.
492 bytes transferred in 0.005 seconds.  Transfer rate 98.40 Kbytes/sec.
Command:
quit
>>> QUIT
221 Quit command received. Goodbye.
WALTER:/u/walter:>ls
```

Figure 34. Display of Login Procedure and FTP Dialog

Check on the Auto UNIX System side if the data transfer was successful.

```
WALTER:/u/walter:>ls o*.*

old.txt

WALTER:/u/walter:>
```

Figure 35. Display of Directory with ls Command

You may now edit and modify the file.

```
WALTER:/u/walter:>vi old.txt

This is a testfile to show the data transfer
requested by Auto UNIX System
  ...
  ...
"old.txt" 6 lines, 486 characters

End the editor, save the file and put the file back to VM/ESA

 WALTER:/u/walter:> put testfile.ftp testfile.ftptest.a
```

Figure 36. Display of File old.txt with vi Editor

Chapter 7. Network File System (NFS) Connection Auto UNIX System to VM/ESA

This chapter describes how to configure and work with the Network File System.

7.1 Setup NFS between Auto UNIX System and VM/ESA

NFS must be a part of TCP/IP on Auto UNIX System and VM/ESA. Depending on the TCP/IP version it can be part:

- of TCP/IP and must only be enabled.
- an optional feature to be installed.

Note: TCP/IP on VM/ESA can only act as NFS-Server. TCP/IP on Auto UNIX System can act as NFS-Server and NFS-Client. Therefore only Auto UNIX System offers a command for the mounting of a foreign file system **(ezmount)**.

Consider the following when you use NFS:

- You need to supply the name of your local domain, the IP address of your nameserver, and your hostname, in the **/etc/resolv.conf** file. Edit this file and modify these lines to your requirements, for example:

  ```
  domain      boeblingen.de.ibm.com
  nameserver  9.164.178.1
  hostname    autounix1
  ```

- Edit the **/etc/exports** file, if necessary, to ensure that it contains correct directory descriptions.

  The initial entry of the exports file is:

  ```
  /hfs/usr
  /hfs/etc
  ```

  Example of the exports file we refer to in this book:

  ```
  /hfs/usr
  /hfs/etc
  /hfs/u/vm
  /hfs/u/walter
  /hfs/u/stan     -access=wernos2,rw=stanos2
  /hfs/u/walter   -rw=waltos2
  /hfs/u/ingrid   -access=wernos2,ro
  ```

  The syntax for a directory entry is:

  ```
  directory    -ro
  directory    -rw=clients
  directory    -access=clients
  directory
  ```

  The keywords ″ro″ and ″rw″ are mutually exclusive.

- To check the status of the NFS server, use the **nfsstat** command.
- When using your system as a client and mounting remote NFS file systems, to perform mount or unmount operations you must have UID=0.

7.2 Mounting and Using NFS from Auto UNIX System

The following example shows how to mount an NFS file system:

```
Assume :
Remote host name is                 BOEVMSPA
The remote VM-guest is              COEX01
The Minidisk you want to access is  191    (A-Disk)
Mount this file system as           /u/vm
Name the mounted HFS                VM.data
```

For a complete description of the mount command refer to *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993, and *VM/ESA V2R3.0 TCP/IP FL310 User*, SC24-5848.

Execute the following command on your Auto UNIX System WS:

```
EZMSUPR:/:>ezmount -t nfs -p boevmspa:coex01.191 vm.data /u/vm
EZM051I Mount will proceed asynchronously.
```

Note: -t nfs ===> type NFS file system
-p ===> path information

To check if the NFS file system is mounted correctly, change to the directory that you have specified in the mount command (/u/vm) and do a list command. The result should be the contents from the A-Disk 191 of user coex01 on the host boevmspa. For example:

```
EZMSUPR:/:>cd u/vm
EZMSUPR:/u/vm:>ls

3390ad00.epsbin  coex01.netlog    instau05.epsbin  l.script
390con00.epsbin  coexaet.direct   instau06.epsbin  profile.exe
5377c020.script  dasd00.epsbin    instau07.epsbin  profile.exec
5377c060.script  dasdad00.epsbin  instau08.epsbin  profile.xedit
5377c070.script  diagda00.epsbin  instau09.epsbin  start00.epsbin
EZMSUPR:/:>
```

You are now able to use this as an additional file space.

Note: If you are using RACF or another External Security Manager (ESM) you have to supply a password with the **MOUNTPW** command to the VM system *before* you issue the **ezmount** command.

A sample program (MOUNTPW C) is supplied with 5654-030 for use on NFS client systems. See Appendix C, “Sample MOUNTPW C Program” on page 135 for the listing of this C program. The MOUNTPW C file must be copied to the client system and compiled into an executable program before you can execute the MOUNTPW command. Compile the MOUNTPW program using a C compiler resident on the client system.

Chapter 8. Network File System (NFS) for VSE/ESA

8.1 Using the Network File System (NFS) for VSE/ESA

As shown in previous chapters, there are several methods to couple a workstation or an Auto UNIX System and a VSE/ESA host system based on TCP/IP protocols and applications. Assuming you intend to access a data set stored on a VSE/ESA host system, you could, for example, use the File Transfer Protocol (FTP) to transfer it onto a system of your choice, process the data there and transmit it back to the VSE/ESA system. However, this process would require three steps and you could not be sure that anyone else also worked on the same data file as FTP does not provide any locking file features. But using an NFS client you could take your favorite client tool and directly process the data file on the VSE/ESA host, without having to transmit it entirely over a network.

The Network File System (NFS) is a client/server application that lets you access files on a remote system and operate them as if they were on your local computer. This means the client could access VSE/ESA VSAM ESDS files, VSE library members or VSE/POWER queue entries as if they were part of the client′s local file system. For this purpose the remote host needs to have an NFS client application running, and the VSE/ESA host needs an NFS server providing the host file access. Both client and server application are coupled using TCP/IP as a method of network based data exchange.

The NFS protocol allows to glue part of the server′s file system into the client file system. The client applications can transparently access the server′s data files and need not take care of network semantics. The method of attaching the server file system or parts of it to the client file system is called to ″mount″ the server′s file system.

Be aware that the TCP/IP for VSE/ESA NFS server is provided as a separately priced feature. This feature does not include NFS client support, neither for the VSE/ESA host itself, nor downloadable to a client. Instead, it depends on the client′s operating system environment whether an NFS client may already be preinstalled (for example, most UNIX alike systems).

If you are running OS/2 or Windows on the workstation instead of a UNIX alike operating system, you could mount the host system on an empty logical drive letter, for example ″X:″. To ″mount″ the VSE resources with an OS/2 NFS client issue:

mount x: vsea:/

However, while the workstation can now operate on the VSE/ESA virtual file system though it was local to the workstation, the file naming convention of the VSE/ESA data access methods still applies. Therefore the VSE/ESA dependent part of the workstation′s file system view is case insensitive, also when mounted on a UNIX alike workstation. While workstation file systems may allow a file system object name built on special characters such as ″-″ (dash), ″_″ (underscore) and others, valid characters allowed on VSE/ESA names are 0-9, A-Z, #, §, $ and a dot (″.″) only. Using a client editor or storing a local file on the

remote host may otherwise cause the file I/O operation to fail. Check the VSE/ESA access method (for example, VSAM or the VSE/Librarian) for naming convention details.

The ″Network File System″ (NFS) for TCP/IP for VSE/ESA is a TCP/IP based network application built on Sun Microsystem′s NFS Version 2 protocol exploiting the ″Remote Procedure Call″ (RPC) protocol. RPC was also developed by Sun Microsystems as an Application Programming Interface (API) to write distributed, TCP/IP based network applications. To achieve system independence, data is converted into a system independent standard representation before being transmitted over the network (in either direction: ″client to server″ or ″server to client″). It is possible to transmit such highly system dependent data types as floating point numbers. This conversion method is known as ″eXternal Data Representation″ (XDR).

Be aware there already exist more recent NFS specifications. NFS clients built on NFS protocol versions other than Version 2 are not supported by NFS for TCP/IP for VSE/ESA.

As NFS was originally developed for UNIX alike operating systems, its semantics are also tightly coupled to UNIX alike file system structures. Therefore NFS does not support record oriented file data organizations, but assumes contiguous, byte stream oriented files only. NFS supports applications processing just parts of a data file. If the data file residing on the VSE/ESA host is organized in records (fixed or variable), NFS hides this semantic from the client application and transmits the file′s contents as though the data organization was byte stream oriented. This may require overlapped record handling on the VSE/ESA host.

To ensure clients can store arbitrary files on the VSE/ESA host and retrieve them with identical input and output formats, the VSE/ESA NFS server stores everything in binary format. So applications running on the VSE/ESA host may not be able to process the data stored by a client if the client is using ASCII encoded data while the VSE/ESA host applications use EBCDIC encoded access semantics.

It is essential to properly define the files (VSAM) or member extensions (VSE/Librarian) for data sets VSE/ESA applications also intend to process.

This causes NFS to perform proper code page translations as well as to determine which files need to be stored in record format and which as contiguous byte streams. Those specifications are provided by NFSTYPES.L. The sample provided with VSE/ESA is shown in Figure 70 on page 139. This configuration file defines the way a file stored on a VSE/ESA system is visible to remote clients, as well as how files are to be stored on a VSE/ESA host, if stored into the TCP/IP for VSE/ESA defined virtual file system and not existing yet.

8.2 Defining Mount Points

NFS servers in operating system environments other than VSE/ESA may require the administrator to explicitly define mount points per user. The NFS server may export different mount points for different users, that is, different users may have different views to the file system tree exported by the NFS server.

NFS for TCP/IP for VSE/ESA however uses any file system entry point as defined by any of your DEFINE FILE statements in your IPINITxx.L configuration file as

mount points for remote systems. It uses the same directory structure as FTP and the WebServer. Further, the use of DEFINE FILESYS will allow to mount any logical file system level generated by TCP/IP based on the user′s standard labels. With this information in mind it is important to specify DEFINE NFSD in the IPINITxx.L configuration member following all DEFINE FILE and DEFINE FILESYS definitions.

So you must make sure your mount points are defined to TCP/IP. SAM files cannot be accessed as they do not have a directory structure. VSAM ESDS files however comprise a logical directory structure and can therefore be accessed by NFS. Further VSE/Librarian libraries and sublibraries may be used as mount points and accessed from remote NFS clients as well as POWER queues.

An example of defining a VSAM catalog to NFS and FTP would be:

DEFINE FILE,PUBLIC=′UCAT′,DLBL=IJSYSUC,TYPE=VSAMCAT

If you define a mount point after NFS is active, you will need to terminate NFS and restart it once more.

8.3 Activating NFS for VSE/ESA

Enter the NFS startup command. You can activate the NFS Daemon with the following command:

DEFINE NFSD,ID=identifier,TRANSLATE=name,CONFIG=name

Remember, ″identifier″ is any name you choose. For starters, you might use NFSD as an identifier. Also, the DEFINE command can either be entered from the console, or it can be included in the IPINITnn initialization deck for TCP/IP for VSE/ESA. If you do not enter a translation table name, then the normal translation table ″DEFAULT″ will be used. If you enter an invalid translation table name, then NFS will use the table that is in the supervisor. The CONFIG name will default to NFSCFG. You can override this to use a different configuration file name. This is the name of the file that will contain an assortment of special parameters. A default file, NFSCFG.L is provided in PRD1.BASE. This file is shown in Figure 71 on page 141. You will want to copy this file into a sublibrary that is searched before PRD1.BASE so future updates will not overlay your modified file.

8.4 Operating NFS with VSE/ESA

From time to time, you may want to monitor or modify the NFS system. You can do this by issuing commands at the partition REPLID prompt in the TCP/IP for VSE/ESA partition.

For example, if you are running TCP/IP for VSE/ESA in F7, but there is no outstanding reply identifier, you would enter the VSE command:

MSG F7

A partition reply identifier will be displayed on the VSE console.

To free the NFS cache area you can issue the command:

087 NFS FLUSH subdirectoryname

This command may be necessary because the information which is derived from NFS resides in NFS cache. When NFS updates the directory, the cache is also updated. But if the directory is updated externally to NFS, the NFS cache will not be updated.

For example, if you do a DIRLIST of PRD2.GEN1, and then you add a macro to the library, the NFS client will not see the new entry. The command FLUSH PRD2.GEN1 will tell NFS to free the cache area. The next DIRLIST request for PRD2.GEN1 will cause new cache information to be loaded.

8.5 NFS Command Processing from Batch

Besides having an interactive NFS command interface (see above), you can also enter NFS commands via the NFSUTIL batch utility.

Since NFS does not know when an external source updates the directory, you will need to provide an indicator for NFS.

For example, if you run a batch job that creates a new file, you will need to tell NFS to update its directory cache. You can either do this through an operator console command, or set up a timer value in the DEFINE FILE statement (this option is available with TCP/IP for VSE, 1.3.0 - APAR PQ14724 or later). For example, if you want a VSE library to be flushed from cache every 10 minutes, you would define the library as follows:

DEFINE FILE,PUBLIC='PRD2',DLBL=PRD2,TYPE=LIBRARY,NFSTIMER=600

This also means that, for example PRD2.GEN1 and PRD2.CONFIG will have the same NFS timer values. All subdirectories of a parent directory will have the same inherited value.

In the following example, we are updating the VSE library, and then flushing the directory buffers NFS may hold for the sublibrary. As mentioned above, if an external batch program updates a sublibrary that is also managed by NFS, NFS will not know about this change as it is performed outside of its control. Therefore you need to clear the NFS cache, forcing real I/O to take place with the next directory access and NFS to update the directory cache with the new data contents.

```
// JOB LIBR UPDATE
// EXEC LIBR
 A S=PRD2.CONFIG
 DELETE OLD.TXT
/*
// LIBDEF *,SEARCH=PRD2.CONFIG
// EXEC NFSUTIL,PARM='FLUSH PRD2.CONFIG'
/&
```

Figure 37. NFS Batch Utility

8.6 Terminating NFS for VSE/ESA

On occasion, you might have the need to terminate NFS. If you do a normal TCP/IP for VSE/ESA shutdown, NFS will notice the request and terminate normally. So you do not need to terminate NFS before you shut down TCP/IP for VSE/ESA.

If you want to terminate just an NFS Daemon without performing a TCP/IP, type the following at the VSE console:

DELETE NFSD,ID=idname

You should see a series of messages indicating NFS is terminating, and the NFS Daemon will indicate when shutdown is complete. After the Daemon has been terminated, NFS will take a few extra seconds to clean up some control blocks and perform some other tasks.

Then you will see the message:

IP0602I NFSD Shutdown complete for idname

where idname is the ID used to initialize NFS in the first place.

You will not be able to reactivate NFS with the same idname until NFS has completely terminated.

8.7 NFS Considerations ...

NFS uses the common file I/O system used by the other file access daemons, such as FTP and HTTP. However, NFS has certain special requirements. For example, you cannot tell VSE/ESA how many records are in the file you are writing to the mainframe DASD, so determining the actual resource requirements can become quite complicated. NFS will make an assortment of assumptions based on settings maintained in the configuration file. It will attempt some best-guess definitions where settings are not made or are unavailable. However, because of the diversity of file systems that are available, some activities may not be possible, or may require some user training, such as not including imbedded spaces in a file name.

8.7.1 For VSE/POWER

The VSE/POWER queues can hold data that can be used just as any other file system, but there are some limitations of which you need to be aware.

You cannot read a file that is active. This means if you attempt to read some JCL from the reader queue that is currently being executed, you will get a read error. You may see an invalid name in the reader queue. When you drag and drop a name, such as ″SAMPLE.JCL″ into the reader queue, some systems, such as Windows ′95, will not perform a DirList() function to verify the contents of the directory, but will instead update the listing of the local folder contents. In this case, you will actually see a SAMPLE.JCL in the reader. But what has really happened is an XPCC PUT into the reader queue has occurred, and the file will be accessed in the following manner:

membername.number.segment.queue

If POWER has assigned a reader queue number of 00123 to the new entry, the result is the real name:

SAMPLE.00123.00.RDR

In order to see the real contents, you must perform a ″VIEW″ and a ″REFRESH″ selection while in Windows to see the accurate directory display.

VSE/POWER is the one file system where the member sent is actually named differently than what the client indicated. Because of this, a ″RENAME″ command is not allowed to be processed within POWER. If you want to rename the membername part of the name, just copy it with the new name and delete the original.

A VSE/POWER member name sent to VSE/ESA cannot be more than eight bytes long. You can send ″TEST.JCL″, but you cannot send ″TESTINGJCL″. The file type that follows can also only by eight bytes long, but is not used by VSE/POWER.

There are two special classes in the NFS POWER queue display. They are ″BIN″ and ″ALL″. ″ALL″ will display all members within the queue. ″BIN″ will force any member read from or sent to this class to be in binary mode (no EBCDIC/ASCII translation occurs). If you need to submit binary data into the reader queue, this is where it should go.

Although print queue records are stored in variable length format, the NFS server will send them in fixed length format. The reason for this is speed. When the client wants a print queue member, it asks for the exact size. FTP does not have this requirement. NFS will calculate the true size based on the largest print line in the file times the number of total lines (with some CR/LF info thrown in as well). To send variable length files would require a double-read of the file, which would time-out at the client-side for most larger print outs. Therefore, fixed records are sent, which is why FTP will transmit faster, since it will send the variable length records without the need to pad the right side with blanks.

8.7.2 For the Librarian

The Librarian can store any type of data you require. However, remember if you store record types that are not defined in the NFSTYPES.L file, the data will not be easily accessed by a VSE/ESA program. The data will not be translated, will be unblocked, and the LIBR PUNCH request will fail to output this copy elsewhere. Once in a special ″string″ format, any client will be able to use this data, but the host will most likely not.

Therefore, it is important that any text file to be used by VSE/ESA has a file type defined as one that will be translated. All text has a limitation of 80-byte records. If your text has more than 80 bytes in a record, then it will be split into multiple records. If your text has less than 80 bytes in a record, then it will be right-filled with blanks.

8.7.3 For VSE/VSAM

NFS can make use of VSAM ESDS files. In order to read a catalog, the DLBL for the VSAM catalog must contain a pointer back to itself as follows:

```
// DLBL IJSYSUC,'VSAM.USER.CATALOG',,VSAM,CAT=IJSYSUC
```

You can put this in your STDLABEL area for each entry, or in the startup JCL for TCP/IP for VSE/ESA.

To define the VSAM area, you would use a definition such as:

```
DEFINE FILE,PUBLIC='IJSYSUC',DLBL=IJSYSUC,TYPE=VSAMCAT
```

The way that different clients handle the creation of files will cause changes in how a VSAM file is created. For example, under OS/2, if a file already exists, then a WRITE request is immediately started. Under another vendor's product in Windows '95, the creation of a file will first see if the file exists, and if it does, then it is deleted, a null file is created in its place, and the file is then written. If you have read a VSAM file for input, made changes, and now you want to save it, it is important that a DELETE/DEFINE is performed by IDCAMS. For some clients, you may have to issue a DEL command to cause a delete to occur before a define can take place.

All VSAM files have an attribute entry in NFS cache. When the file is deleted, its attributes are "remembered" and are used during the DEFINE process. The IDCAMS utility is used to perform the delete and define process. This utility is CDLOADed into partition GETVIS once. Its starting address is maintained in one of the TCP/IP for VSE/ESA system control blocks for other processes to be able to invoke this program.

By default, output from IDCAMS is routed to SYSLST. This may or may not be active in your environment. If it is, you can look at the output by issuing a SEGMENT command to TCP/IP for VSE/ESA in order to debug any problems you may be having.

8.8 Usage Hints for VSE/ESA NFS Server

If you choose to process a host file using a client editor and change a single byte, the editor will rewrite the complete file. This is always necessary, regardless of the data organization on the host. Depending on the file size this may take some time.

When processing huge files the processing may become very slow if the "read cache" defined to the NFS server is too small. You may want to accommodate the value of "READCACHESIZE" to your most usual requests to experience better "read" performance. A read cache unable to hold average file system read requests may also lead to high CPU utilization. This happens because the cache may be reused immediately after being filled when processing huge files, thus introducing unnecessary processing overhead. As the NFS read cache is allocated in partition GETVIS 31 you should define as much as you can afford for best data throughput.

Chapter 9. NFS - A Practical Approach

In this chapter we will show you a few aspects of how to use NFS in the real world.

9.1 Introduction

NFS for VSE/ESA runs as a separate daemon inside the TCP/IP partition. It is optionally started and stopped, not demanding the entire TCP/IP job to recycle whenever we wish to have NFS running or not. For example, you may have to change the configuration of NFS, or you may want to prevent users from accessing the files while updating them.

9.2 Putting NFS to Work

We can have NFS started using the DEFINE NFSD. This command can be used either in the TCP/IP configuration file, after the definitions for the File System are done, or at any time after TCP/IP is up and running, simply by entering at the console: **MSG fx** (where fx is the TCP/IP partition) and then, after it opens a reply id:

xx DEFINE NFSD,ID=nfsdid[,CONFIG=configfile][,TRANSLATE=tablename]

The ID= is any ID you choose. CONFIG can specify a file name containing your specifications for this run of the daemon. We will talk a little about translation at the end of this chapter. Below is an example of what could be a configuration file containing all options and defaults:

```
CATALOG NFSCFG.L                          REPLACE=YES
*
* The following are suitable for site tuning
*
* Please note that the extra lines (such as these) with our
* comments should NOT be present in your file, as they will
* cause ERRORs during initialization
DIRCACHESIZE=1M          /* 80K (IBM DEFAULT) space for directory            */
                         /* entries: 40 to 84 bytes each (RECFM=V)           */
DirGroupSize=10          /* The number of entries to send during I/O         */
DirlistExit=No           /* DIRLIST exit name, or YES for "NFSEX01"          */
                         /* Here you can specify if you want or need to use  */
                         /* an EXIT to better control access to the files.   */
                         /* We strongly recommend you to have an exit, as    */
                         /* you will probably see that the standard NFS      */
                         /* security will not suffice.                       */
                         /* In TCP/IP for VSE/ESA there is a good            */
                         /* description on how to write a user exit.  You    */
                         /* will also find a sample in PRD1.BASE by the      */
                         /* name of NFSEX01.A. Remember: this exit will      */
                         /* allow directory entries to be shown or not.      */
                         /* True security though, must be dealt with in the  */
                         /* exit specified in the TCP/IP configuration file, */
                         /* refer to Figure 62 on page 125 for more details. */
                         /* Use DEFINE SECURITY,DRIVER=your_exit and         */
                         /* SET SECURITY=ON                                  */
```

```
DirlistNullFiles=Yes    /* Yes=Include zero-length files in DirList          */
                        /* if you do not want to show user files that        */
                        /* are EMPTY, choose NO.                             */

LimitVSAMToESDS=Yes     /* Yes=Display only ESDS data set via VSAM           */
                        /* Since NFS can only deal with VSAM ESDS files      */
                        /* this is a good choice. We believe that            */
                        /* showing anything to the user which he/she will    */
                        /* not be able to use can be confusing.              */

LoadPowerClasses=No     /* Yes=Define all classes at startup                 */

LoadPowerQueues=No      /* Yes=Define all queue names at startup             */

MaxAllocErrs=3          /* Maximum consecutive ALLOC FRBLOK errors           */

MaxRequests=1000        /* Maximum requests that can be serviced             */
                        /* each entry occupies 160 bytes of 31-bit GETVIS    */
                        /* area. The minimum 100 works, but may be small     */
                        /* if the number of users increase. 1000 is large.   */
                        /* We suggest to start your configuration at about   */
                        /* 400 and try. It is always better to give more     */
                        /* later, than to specify large values               */
                        /* which may do the job, but at the expense of       */
                        /* valuable resources, and never be sure of what     */
                        /* you really need.                                  */

OS2Support=Yes          /* Perform additional routines for OS/2              */
                        /* Do not use if you do not have OS/2 workstations   */
                        /* in your network. Otherwise YES is recommended     */

PrintIDCAMS=Yes         /* Yes=Output IDCAMS results to SYSLST               */
                        /* We believe that once things are fully tested      */
                        /* and running for a while at your site, this        */
                        /* parameter should be set to NO, since probably     */
                        /* no one is going to check the automatic IDCAMS     */
                        /* runs implied by NFS every time it has to          */
                        /* define a new file or delete it. With Yes,         */
                        /* you can always have them at the POWER LST         */
                        /* queue for auditing purposes.                      */

READCACHESIZE=1M        /* 512K (IBM default) The bigger the better          */
                        /* within reasonable values. But not always. If      */
                        /* you have a few files to be shared with the end    */
                        /* users then maybe this rule applies. But           */
                        /* remember that this is an individual file cache    */
                        /* allocation figure. Imagine that if you have       */
                        /* many files accessed at the same time and if       */
                        /* their sizes are large, how much memory do you     */
                        /* really need? It also depends on how large the     */
                        /* files are and on the frequency with which they    */
                        /* are accessed. There is a trade-off with           */
                        /* available storage and paging here, with which     */
                        /* you will have to deal.                            */
                        /* Sometimes you will gain no performance just by    */
                        /* specifying a higher value.                        */
                        /* NFS is smart enough to find out if the file it    */
                        /* is about to process is smaller than the value     */
                        /* for READCACHESIZE, if it is, then it allocates    */
                        /* the cache from GETVIS just about the file size.   */

ReadCacheTime=30        /* Max seconds a recently-read entry remains         */

Security=No             /* Yes=Use the same user exit that FTP does          */
```

```
SHOWACTIVE=YES          /* YES=SHOW PWR Q ENTRIES DISP=*                  */
                        /* NFS will show in a DIR display POWER queue     */
                        /* entries which are in DISP=* (active job        */
                        /* executing in the RDR, report being printed).   */
                        /* There is no way NFS will enable any other      */
                        /* access to those files, not even allowing the   */
                        /* user to read them.                             */
ShowNonExported=No      /* Yes=List truncated UDP EXPORT lists            */
ShowPhases=No           /* No=Suppress .PHASE info in DIRLIST             */
                        /* The recommended value is NO, IBM default.      */
                        /* Users usually take no interest in PHASEs,      */
                        /* selecting NO will reduce both CACHE usage and  */
                        /* paging, and prevent useless information from   */
                        /* being sent over TP facilities to the user.     */
ShowRPCError=No         /* Yes=Show info when GETPORT fails via RPC       */
ShowStaleInfo=No        /* Yes=Show info when request goes stale          */
TimeOffset=+0           /* Number of minutes to use for accurate TOD      */
Truncate=Yes            /* No=Don't truncate WRITE records to VSE         */
UnixTextMode=Off        /* ON=Omit CR character from end-of-record        */
VSAMTableSize=64k       /* Maximum size for the VSAM attribute table      */
WakeUpTime=5            /* # of seconds to check cache usage              */
WriteCacheTime=30       /* Max seconds a written entry remains            */
XDRExit=No              /* Indicate XDR support and possible name         */
*
* The following are options for internal use and should not be
* used unless requested to do so by Technical Support.
*
DatagramTrace=NO        /* Yes=produce SYSLST output of I/O traffic       */
Debug=No                /* Yes=Output a lot of debugging info             */
ListExports=No          /* Yes=List all mount points that we know of      */
MaxExportSize=4K        /* Maximum outgoing EXPORT datagram (MOUNTD)      */
MAXPACKETSIZE=8K        /* DEFINES THE LARGEST SUPPORTED DATAGRAM         */
VERSION3=NO             /* TO ALLOW VERSION 3 PROTOCOL                    */
WatchDirCache=No        /* Yes=Show DIR cache real time statistics        */
WatchReadCache=No       /* Yes=Show READ cache real time statistics       */
WatchReads=No           /* Yes=Show READ real time activity               */
WatchWrites=No          /* Yes=Show WRITE real time activity              */
WatchWriteCache=No      /* Yes=Show WRITE cache real time statistics      */
/+
```

Once the NFS Daemon is running, VSE becomes an NFS server. VSE NFS can not be an NFS client. But, what does that mean? It means that clients can MOUNT network drives which are actual VSE entities, VSE will not mount network drives existing on other systems.

Mounting a drive actually starts a drive simulation in the system. The drive is actually a directory or a set of directories in an NFS server. The directories which are allowed to be mounted by the clients must be specifically authorized

in a list, a table or file. In VSE, the files or directories to be mounted are specified in the configuration file IPINITxx via DEFINE FILE statements, see Appendix B, “Figures and Samples for VSE/ESA Setup” on page 111, for more information. These statements define the File System and are the same ones used to define files for FTP operations.

9.3 Auto UNIX System Mounting Points

In other systems, such as Auto UNIX System, the directories to be accessed are defined in a file called EXPORTS. Any directory or subdirectory included in that file is called a MOUNTING POINT in that system. It means that they are available to be mounted by clients of the NFS Daemon. Mounting a VOLUME, simulates a drive at a workstation (or Auto UNIX System) whose root is a mounting point at the NFS server. The EXPORTS file at Auto UNIX System is listed below as an example:

```
#
# Licensed Material - Property of IBM
#
# 5647-A01
#
# (C) Copyright IBM Corp. 1997, 1998
#
# Status = HEZ6606
#

/hfs/usr
/hfs/etc
/hfs/u/stan     -access=wernos2,rw=stanos2
/hfs/u/walter   -rw=waltos2
/hfs/u/ingrid   -access=wernos2,ro
```

EXPORTS file at Auto UNIX System

There are a few rules to be aware of in order to fill in a correct exports file. The example above shows that directory /hfs/u/stan can be accessed by workstation wernos2 and workstation stanos2. The difference is that wernos2 can only read from the directory but workstation stanos2 can also write to it. It also makes the directory not mountable by anybody else. For directory /hfs/u/walter, workstation waltos2 will have rw authority while everybody else will have read only access to it. Finally, /hfs/u/ingrid can only be accessed in read only mode from workstation wernos2.

The manual *DFFSMS/MVS Version 1 Release 2 Network File System Customization and Operation*, SC26-7029 has a good description on how to specify the parameters. We copied some text from that manual here, to make it easier to understand the subject. For full information please always refer to that manual.

```
>>- directory--* -ro---------------------------------+----><
               ]     <-:---+                         ]
               + -rw=client]-------------------------+
               ]         <-:---+                     ]
               + -access=client]--+----------------+--+
                                  ]     <-:---+    ]
                                  ] ,rw=client]----+
                                  + ,ro------------+
```

directory MVS high-level qualifier, data set name, or alias to a user catalog; the name must conform to MVS data set naming conventions unless OpenEdition MVS is used (for an HFS directory entry, you need to use the hfs prefix that is used at your site).

-ro Export the directory as read only. If not specified, the directory is exported as read/write.

-rw=client[:client...]
The directory is exported as read/write to specified clients, and read-only to everyone else. Separate client names by colons.

-access=client[:client...][[,rw=client[:client...]] | [,ro]]

Gives access only to clients listed.

If neither rw nor ro is specified for the -access parameter, then the clients listed have read/write access and the rest of the clients has no access.

If the rw parameter is specified for the -access parameter, the associated clients have read/write access to the directory, and the clients specified in the access list but not in the rw list have read-only access.

If the ro parameter is specified for the -access parameter, the clients in the access list have read-only access to the directory, and the rest of the clients has no access.

Separate client names by colons.

If no options are specified, the default value allows any client to mount the given directory with read/write access.

Following are examples of entries in an exports data set:

```
mvsnfs          -ro                  # give read-only
                                     # access to all
                                     # clients

daniel.text                          # give read/write
                                     # access to all
                                     # clients
```

```
andreas.mixds   -rw=fsrs001:fslab004:fslab007
                                          # give read/write
                                          # access to clients
                                          # fsrs001, fslab004 &
                                          # fslab007, and give
                                          # read-only access to
                                          # all other clients


/hfs/newproductdirectory  -rw=johnson     # give r/w access
                                          # to this directory
                                          # to the client
                                          # johnson;
                                          # give r/o access
                                          # to all others

rita.pds       -access=fsrs001:fslab007
                                          # give r/w access
                                          # only to clients
                                          # fsrs001and fslab007
```

Notes:

1. The keywords *ro* and *rw* are mutually exclusive.
2. The ability to write (that is, *rw* specified or *access* specified without other parameters) implies read access also.
3. If *access* and *rw* are specified together, the client names in the rw list are logically or'ed with the access list to determine the total list of clients with read access.
4. Multiple lines can be used in the exports data set for a given directory to merge the access list and the *rw* list. However, similar clauses (for example, an *access* followed by an *access*) completely replace any previous specification. If *ro* is specified for a data set on one line and a further line specifies *rw* for that data set, the *rw* undoes the *ro* specified earlier. Similarly, a line with null options completely undoes all previous specifications for that directory, giving read/write access to all clients.

9.4 Some Practical Examples

From now on we will exercise some features available in VSE and Auto UNIX System NFS.

9.4.1 VSE/ESA

The prerequisites to run the samples are very few. It is necessary to have the ″hosts″ file updated with the IP addresses of all hosts involved and the NFS client must be configured and running in the workstations, that is all. Having that set up we can start referring to the NFS servers at COEXVSE and COEXAET by the names chosen in the ″hosts″ file. We picked the short names ″vse″ to designate COEXVSE and ″aus″ for the Auto UNIX System COEXAET machine. To show that the same system can be called by any name, the very same COEXVSE was nicknamed ″coexvse″ in the Auto UNIX System ″hosts″ file. To show that VSE/ESA NFS is not only connectable to Auto UNIX System, but to any NFS platform, imagine an OS/2 window, where we could type, in order to create a virtual drive, the i:

```
[C:docs]mount i: vse:\
mount: vse:\
user name: wern
password:******

NFS Drive 'i:' was attached successfully.

[C:docs]i:

[I:\]dir | more

The volume label in drive I is NFS.
The Volume Serial Number is 0100:068B.
Directory of I:\

11-27-98 12:05p         <DIR>      0 ----  AUTO
11-27-98 12:05p         <DIR>      0 ----  AUXFILK
11-27-98 12:05p         <DIR>      0 ----  CICS
11-27-98 12:05p         <DIR>      0 ----  CU37XX
11-27-98 12:05p         <DIR>      0 ----  IJSYSRS
11-27-98 12:05p         <DIR>      0 ----  INFO
11-27-98 12:05p         <DIR>      0 ----  POWER
11-27-98 12:05p         <DIR>      0 ----  PRD1
11-27-98 12:05p         <DIR>      0 ----  PRD2
11-27-98 12:05p         <DIR>      0 ----  SYS
11-27-98 12:05p         <DIR>      0 ----  TCPCAT
11-27-98 12:05p         <DIR>      0 ----  VSAM
11-27-98 12:05p         <DIR>      0 ----  VSE
11-27-98 12:05p         <DIR>      0 ----  VSESP
11-27-98 12:05p         <DIR>      0 ----  VSESPUC
-- More --

       15 file(s)         9,216 bytes used
                    536,870,912 bytes free
```

9.4.2 VSE/POWER

This section shows how to handle VSE/POWER queues and files.

```
[I:]
[I:\]cd power

[I:\power]dir

The volume label in drive I is NFS.
The Volume Serial Number is 0100:068B.
Directory of I:\power

11-30-98  6:54a         <DIR>      0 ---r  .
11-30-98  6:54a         <DIR>      0 ---r  ..
11-30-98  6:45a         <DIR>      0 ----  LST
11-30-98  6:45a         <DIR>      0 ----  PUN
11-30-98  6:45a         <DIR>      0 ----  RDR
        5 file(s)          2,560 bytes used
                    536,870,912 bytes free

[I:\power]
```

The POWER queues appear as DIRectories.

Let's look at one of them, the RDR queue in more detail.

```
[I:power]cd rdr

[I:\power\rdr]dir | more


The volume label in drive I is NFS.
The Volume Serial Number is 0100:068B.
Directory of I:\power\rdr

11-30-98  6:59a         <DIR>      0 ---r  .
11-30-98  6:59a         <DIR>      0 ---r  ..
11-30-98  6:45a         <DIR>      0 ----  0
11-30-98  6:45a         <DIR>      0 ----  1
11-30-98  6:45a         <DIR>      0 ----  2
11-30-98  6:45a         <DIR>      0 ----  3
11-30-98  6:45a         <DIR>      0 ----  4
11-30-98  6:45a         <DIR>      0 ----  5
11-30-98  6:45a         <DIR>      0 ----  6
11-30-98  6:45a         <DIR>      0 ----  7
11-30-98  6:45a         <DIR>      0 ----  8
11-30-98  6:45a         <DIR>      0 ----  9
11-30-98  6:45a         <DIR>      0 ----  A
11-30-98  6:45a         <DIR>      0 ----  ALL
11-30-98  6:45a         <DIR>      0 ----  B
11-30-98  6:45a         <DIR>      0 ----  BIN
11-30-98  6:45a         <DIR>      0 ----  C
11-30-98  6:45a         <DIR>      0 ----  D
-- More --

11-30-98  6:45a         <DIR>      0 ----  E
11-30-98  6:45a         <DIR>      0 ----  F
11-30-98  6:45a         <DIR>      0 ----  G
11-30-98  6:45a         <DIR>      0 ----  H
11-30-98  6:45a         <DIR>      0 ----  I
11-30-98  6:45a         <DIR>      0 ----  J
11-30-98  6:45a         <DIR>      0 ----  K
11-30-98  6:45a         <DIR>      0 ----  L
11-30-98  6:45a         <DIR>      0 ----  M
11-30-98  6:45a         <DIR>      0 ----  N
11-30-98  6:45a         <DIR>      0 ----  O
11-30-98  6:45a         <DIR>      0 ----  P
11-30-98  6:45a         <DIR>      0 ----  Q
11-30-98  6:45a         <DIR>      0 ----  R
11-30-98  6:45a         <DIR>      0 ----  S
11-30-98  6:45a         <DIR>      0 ----  T
11-30-98  6:45a         <DIR>      0 ----  U
11-30-98  6:45a         <DIR>      0 ----  V
11-30-98  6:45a         <DIR>      0 ----  W
11-30-98  6:45a         <DIR>      0 ----  X
11-30-98  6:45a         <DIR>      0 ----  Y
11-30-98  6:45a         <DIR>      0 ----  Z
       40 file(s)        20,480 bytes used
-- More --

                    536,870,912 bytes free
```

As you can see, the classes are viewed as directories as well. The VSE/POWER queues can hold data that can be used like any other file system, but there are some limitations of which you need to be aware.

Observe the two special subdirectories: BIN and ALL. ″BIN″ will force any member read from or sent to this class to be in binary mode (no EBCDIC/ASCII translation occurs). If you need to submit binary data into the reader or punch queue, this is where it should go. VSE/ESA POWER can be used as a central hub for forwarding files, for instance. ″ALL″ will display all members that are within the queue, that is: if you CD to ALL, the files will be shown as if they were in one single directory.

You cannot read a file that is active, that means: with DISP=*, which would be the case for a running job or a report being printed. Depending on the options selected in the NFSCFG file, the active files may not even be shown. This means that if you attempt to read some JCL from the reader queue that is currently being executed, you will get a read error.

You may see a different name in the reader queue when you drag and drop a file, such as ″SAMPLE.JCL″, into the reader queue. Some systems, such as Windows ′95, will not do a DirList() function to verify the contents of that directory, but will only update the listing of the local folder contents. In this case, you will actually see a SAMPLE.JCL in the reader queue which does not have exactly this name. What has really happened is that an XPCC PUT into the reader queue has occurred, and the file will have the following name, by which it must be accessed: membername.number.segment.queue

If POWER has assigned a reader queue number of 00123 to the new entry, then the real name is: SAMPLE.00123.00.RDR. In order to see the real contents, you must perform a ″VIEW″ and a ″REFRESH″ selection while in Windows to see the accurate directory display.

VSE/POWER is the one file system where the member sent is actually named differently from what the client indicated. Because of this, a ″RENAME″ command is not allowed to be processed within POWER. If you need to rename the membername part of the name, the only way is by copying it to the new name and deleting the original member.

A VSE/POWER member name sent to VSE/ESA cannot be more than eight bytes long. You can send ″TEST.JCL″, but you cannot send ″TESTINGJCL″. The file type that follows can also only by eight bytes long, but is not used by VSE/POWER.

Although print queue records are stored in variable length format, the NFS server will send them in fixed length format. The reason for this is speed. When the client wants a print queue member, it asks for the exact size. FTP does not have this requirement. NFS will calculate the size based on the largest print line in the file times the number of total lines (with some CR/LF info thrown in as well). To send variable length files would require a double-read of the file, which would time-out at the client-side for most larger print outs. Therefore, fixed records are sent, which is why FTP will transmit faster, since it will send the variable length records without the need to pad the right side with blanks.

Let us go deeper into the directory tree and find out what more there is to see. We chose at random class 7:

```
[I:powerrdr]cd 7
[I:\power\rdr\7]dir
```

```
The volume label in drive I is NFS.
The Volume Serial Number is 0100:068B.
Directory of I:\power\rdr\7

11-30-98  7:02a         <DIR>      0 ---r  .
11-30-98  7:02a         <DIR>      0 ---r  ..
11-30-98  4:59a           328      0 ----  PAUSEF7.00033.00.RDR
11-30-98  4:59a           738      0 ----  TCPSTART.01789.00.RDR
        4 file(s)         2,090 bytes used
                    536,870,912 bytes free
```

We could try to EDIT one of the jobs:

```
[I:powerrdr7]e
pausef7.00033.00.rdr
```

and this is what the text editor shows us: nothing else but the PAUSE for F7:

```
* $$ LST CLASS=A,DISP=D
// JOB PAUSEF7
// PAUSE
/&
```

But, wait, since we are editing, if we SAVE the file, will it alter a job in the RDR queue? The answer is no, but you can still alter the job. It sounds strange, until you find out that what actually happens is that NFS issues an XPCC FUNC=SENDR... macro to store the file being edited in the RDR queue, so, you end up submitting a job at each save command in the editor. Unnecessary to say that this can be harmful to your operations. Either, never edit jobs with D or K dispositions, or do not save them until they are absolutely in accordance with what you want them to be. If the jobs have a disposition of L or H you can always delete them from the RDR after editing, if saving more than once.

But, then, submitting a job to the RDR from any directory, either in a PC or in the HFS of Auto UNIX System, is as simple as copying the file to the directory POWER.RDR, in the mounted drive. For example, we will mount an 'r' drive directly into the POWER RDR, on an OS/2 workstation and then similarly on Auto UNIX System:

```
[C:docs]mount r: vse:\power.rdr
NFS Drive 'r:' was attached successfully.
mount: vse:\power.rdr
user name: wern
password:******

   NFS Drive 'r:' was attached successfully.

[C:\docs]
[C:\docs]copy nfsos2vse r:
   1 file(s) copied. ←————————job submitted
```

The VSE console shows that the job was submitted:

```
SYSTEM:  VSE/ESA              VSE/ESA 2.3                    USER:  WERN
                                                             TIME:  09:09:00
 F7-0082 IPN300I Enter TCP/IP Command
 d rdr,0
 AR 0015 1C39I COMMAND PASSED TO VSE/POWER
 F1 0001 1R46I  READER QUEUE   P D C S  CARDS
 F1 0001 1R46I  OS2TOVSE 03029 3 L 0        6 FROM=(SYSTCPIP)
 r rdr,os2tovse
 AR 0015 1C39I COMMAND PASSED TO VSE/POWER
 F1 0001 1R88I  OK
 BG 0001 1Q47I   BG OS2TOVSE 0270Z FROM (SYSTCPIP) , TIME= 9:09:00
 BG 0000 // JOB OS2TOVSE
         DATE 12/11/1998, CLOCK 09/09/00
 BG 0000 * this job is sent from OS/2 to VSE
 BG-0000 // PAUSE just to show NFS job submission
 0
 BG 0000 EOJ OS2TOVSE
         DATE 12/11/1998, CLOCK 09/10/32, DURATION   00/01/32
 BG 0001 1Q34I   BG WAITING FOR WORK

 ==>

 1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD     12=RTRV

 ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Doing the same on Auto UNIX System is just as easy.

```
EZMSUPR:/:>
EZMSUPR:/:>ezmount -t nfs -p coexvse:/ vse.data /u/vse
  EZM051I Mount will proceed asynchronously.

EZMSUPR:/:>cd /u/vse
EZMSUPR:/u/vse:>ls
  AUTO     CICS     CU37XX   INFO    POWER   PRD2    SYS     VSE     VSESPUC
  AUXFILK  COMMON   IJSYSRS  NFSCAT  PRD1    STAN    VSAM    VSESP   VTAM
EZMSUPR:/u/vse:>cp /u/jobs/austovse.job /u/vse/power/rdr/0/austovse
                                  ^
                                  |
                                  +---------- job submitted
EZMSUPR:/u/vse:>
```

Later, in chapter 9.4.7, “NFS Translation Tables” on page 94 we will discuss file extensions, but for now just keep in mind that we have an ASCII file on Auto UNIX System (which is a job) and are submitting (copying) it to the VSE/ESA POWER RDR queue, which in turn deals only with EBCDIC files, as a rule. Keep in mind its extension: .JOB.

After copying it, the console shows:

```
 SYSTEM:  VSE/ESA              VSE/ESA 2.3                      USER:  WERN
                                                              TIME:  13:54:54
F7-0082 IPN300I Enter TCP/IP Command
d rdr,0
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R46I  READER QUEUE   P D C S  CARDS
F1 0001 1R46I  PRTDUMPA 00036 3 L 0        6 FROM=(SYSA)
F1 0001 1R46I  PRTDUMPB 00037 3 L 0        6 FROM=(SYSA)
F1 0001 1R46I  EZMVSC08 02136 3 L 0       39 FROM=(STAN)
F1 0001 1R46I  EZMVSV06 02147 3 L 0       21 FROM=(SYSTCPIP)
F1 0001 1R46I  PAUSEBG  03023 3 L 0        4 FROM=(STAN)
F1 0001 1R46I  AUSTOVSE 03066 3 L 0        6 FROM=(SYSTCPIP)
r rdr,austovse
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R88I  OK
BG 0001 1Q47I   BG AUSTOVSE 03066 FROM (SYSTCPIP) , TIME=13:54:50
BG 0000 // JOB AUSTOVSE
        DATE 12/11/1998, CLOCK 13/54/50
BG 0000 * this job is sent from Auto Unix System to VSE
BG-0000 // PAUSE to show NFS job submission to POWER RDR
0
BG 0000 EOJ AUSTOVSE
        DATE 12/11/1998, CLOCK 13/54/54, DURATION   00/00/03
BG 0001 1Q34I   BG WAITING FOR WORK
```

9.4.3 VSE/VSAM Files

In using NFS, there is more than simple job submission from Auto UNIX System to VSE/ESA. We can also work with VSAM files. Here we will just use the vi editor to open a VSAM file that we know to be recorded in text format, to show that it is possible to work with foreign files as if they were native file system ones.

```
EZMSUPR:/:>
EZMSUPR:/:>cd /u/vse
EZMSUPR:/u/vse:>ls
  AUTO     CICS     CU37XX   INFO     POWER   PRD2     SYS     VSE     VSESPUC
  AUXFILK  COMMON   IJSYSRS  NFSCAT   PRD1    STAN     VSAM    VSESP   VTAM
EZMSUPR:/u/vse:>cd vsespuc
EZMSUPR::>ls
  AUTO.UNIX.ESDS                  DOS.WORKFILE.SYS002.RECOVER.F2
  CICS.AUTO.STATS.A               DOS.WORKFILE.SYSLNK.F8
  CICS.AUTO.STATS.B               MQSERIES.MQFSSET
  CICS.DUMPA                      CICS.DUMPB
  CICS.TD.INTRA                   VSE.PRIMARY.LIBRARY
  TEXTFILE.TXT                    VTAM.DSDB1
  DEFAULT.MODEL.ESDS.SAM          VTAM.DSDB2
  DFHTEMP                         VTAM.DSDBCTL
  DIR.INCOMING.FILES              VTAM.TRSDB
  DOS.WORKFILE.SYS001.RECOVER.F2
EZMSUPR::>vi textfile.txt
```

This is what the vi screen shows:

```
01/31/98 130    1200.00                          ¬M
02/28/98 150    1400.00                          -M
03/31/98 160    1500.00                          -M
04/30/98 140    1300.00                          -M
05/31/98 150    1400.00                          -M
06/30/98 160    1500.00                          -M
07/31/98 180    1700.00                          -M
08/31/98 170    1600.00                          -M
09/30/98 170    1600.00                          -M
10/31/98 175    1650.00                          -M
11/30/98 180    1700.00                          -M
12/31/98 182    1720.00                          -M
```

Which is the file contents. In this book we are using the most simple tools we have, but it is fairly obvious that you will not be editing VSAM files. You will have Auto UNIX System programs accessing VSE/VSAM files. If we copy a file from any Auto UNIX System directory into a VSAM catalog, just by typing for example:

```
cp/u/jobs/tovse.txt  /u/vse/vsespuc/FROMAUST.TXT
```

NFS will internally run IDCAMS and define a cluster before copying its data. IDCAMS is also activated if we delete a file. Refer to 9.5, “IDCAMS Invocation by NFS” on page 95 for more information.

9.4.4 VSE/ESA Library Members

Besides VSAM files, it is also possible to work with libraries and library members.

```
EZMSUPR:/:>cd u/vse
EZMSUPR:/u/vse:>cd prd2
EZMSUPR::>ls
  AFP       COMM2     GEN1      NFSO11OH  SCEEBASD  SCEECICS  TCPIP
  CICSR     CONFIG    GEN1D     PROD      SCEEBASE  TCP3      UNIX
  COMM      DBASE     JOBS      SAVE      SCEECICD  TCP3NFS
EZMSUPR::>cd /u/jobs ◄——— Auto Unix directory where VSE jobs are kept
EZMSUPR:/u/jobs:>ls
  austovse.job          dvtdosrs.job          results.txt
  get.vse.vsam.file     vsamfile              create.common.table
  pausebg.job
EZMSUPR:/u/jobs:>
EZMSUPR:/u/jobs:>cp * /u/vse/prd2/jobs ◄——— we will try to copy them
                                          all to a sublibrary of PRD2
cp: !/u/vse/prd2/jobs/vsamfile: close failed: EDC5111I Permission denied.
                                    ▲
                                    └———— oops!, an error occurred.
EZMSUPR:/u/jobs:>cd /u/vse/prd2/jobs
EZMSUPR::>ls
  AUSTOVSE.JOB   DVTDOSRS.JOB   PAUSEBG.JOB
  CREATE.COMMON  GET.VSE        RESULTS.TXT
EZMSUPR::>
```

We call your attention to the naming conventions used in VSE libraries, please note the Auto UNIX System files with three or more segment names. They were stripped from the third ones on. A similar problem occurred with files that have only one segment, as the one called ″vsamfile″, only that in these cases the copy fails. VSE librarian routines need the file to be qualified (in the VSE/ESA library they are known as members, not as files). So, be careful when differentiating files beyond the second segment in Auto UNIX System, because the differences

will not be transferred to VSE libraries as it supports only two segment names. We can look at the files (members) using, for example the online DITTO utility. Let us take GET.VSE for example (this is one of those files stripped of further segments):

```
 Process   View   Options   Help
--------------------------------------------------------------------------------
DITTO/ESA for VSE        LDL - Library Directory List              Line 1 of 6

Library PRD2

--Library members sorted by NAME   ---Bytes---- Library ----Last Update----
-Member- --Type-- -LIBID- -Sublib- --Records--- -blocks ---Date--- --Time--
AUSTOVSE JOB      PRD2    JOBS              8 R       1 1998-12-14 12:59.33*
CREATE   COMMON   PRD2    JOBS           3846 B       4 1998-12-14 12:59.38*
DVTDOSRS JOB      PRD2    JOBS             12 R       1 1998-12-14 12:59.38*
GET      VSE      PRD2    JOBS             60 R       2 1998-12-14 12:59.38*
PAUSEBG  JOB      PRD2    JOBS              9 R       1 1998-12-14 12:59.38*
RESULTS  TXT      PRD2    JOBS             60 R       3 1998-12-14 12:59.38*
****  End of data  ****
```

We selected GET.VSE and pressed F2, browse, and we got this:

```
 Process   View   Options   Help
--------------------------------------------------------------------------------
DITTO/ESA for VSE         LB - Library Member Browse

Member GET.VSE           Library PRD2.JOBS          Col 1          Format CHAR
                                                                SYSIPT data NO
      1...5...10....5...20....5...30....5...40....5...50....5...60....5...70..
00000 ****  Top of data  ****
00001 * $$ JOB JNM=VSETOAUS,CLASS=0,DISP=D
00002 * $$ LST CLASS=A,DISP=D
00003 * this job is sent from Auto Unix to VSE,
00004 * it creates a file and sends it back to Auto Unix
00005 // JOB VSETOAUS
00006 // EXEC IDCAMS,SIZE=AUTO
00007  /*                                                              */
00008  /*  DELETE VSAM FILE                                            */
00009  /*                                                              */
00010      DELETE (AUTO.UNIX.KSDS) CL NOERASE PURGE  -
00011      CATALOG(VSESP.USER.CATALOG)
00012  /*                                                              */
00013  /*  DEFINE VSAM FILE                                            */
00014  /*                                                              */
00015  DEF CLUSTER(NAME(AUTO.UNIX.KSDS)             -

     etc....
```

Note: CREATE.COMMON has different characteristics, for instance: it does not have records, it has bytes (3846 B). Also, when we ask to browse it, this is what is shown:

```
Process   View   Options   Help
--------------------------------------------------------------------------------
DITTO/ESA for VSE         LB - Library Member Browse

Position 00000000        Size 3846
Member CREATE.COMMON     Library PRD2.JOBS          Col 1          Format CHAR

Position Byte      1...5...10....5...20....5...30....5...40....5...50....5...60
00000000        1  .....Ä|..Ä+(..|(...<..<.........&...+..............
0000003C       61  ............Ä|...|(...<.....+....<..................
00000078      121  ..............................(....!.....|........
000000B4      181  ............................................................
000000F0      241  ........................................<.......(...<....
0000012C      301  ............................................................
00000168      361  ............................................................
000001A4      421  .<......|((|+.......<.+|......&.................
000001E0      481  ..........<|......&..........<|.....................
```

The file actually is a text file, but is shown as if it were a binary one. The reason for this you will find in 9.4.7, “NFS Translation Tables” on page 94.

9.4.5 Change Directory

We also found that sometimes the cd .. command not always returned to the previous directory level.

```
EZMSUPR:/u/vse:>ls
  AUTO    CICS    CU37XX  INFO    POWER   PRD2    SYS     VSE     VSESPUC
  AUXFILK COMMON  IJSYSRS NFSCAT  PRD1    STAN    VSAM
  VSESP   VTAM EZMSUPR:/u/vse:>cd prd2
EZMSUPR::>ls
  AFP      COMM2    GEN1     PROD     SCEEBASE TCP3     UNIX
  CICSR    CONFIG   GEN1D    SAVE     SCEECICD TCP3NFS
  COMM     DBASE    NFSO11OH SCEEBASD SCEECICS TCPIP
EZMSUPR::>cd unix
EZMSUPR::>ls
  ATCCONOO.B    INDEX.HTML   NFSCFG.L     VTMTRL.B
  ATCSTROO.B    IPINITO1.L   TCPAPPL.B    XX.XXX
  EZMVSCO8.JCL  NEWMEM.B     VTMAHHC.B
EZMSUPR::>cd ..
EZMSUPR::>ls
  AFP      COMM2    GEN1     PROD     SCEEBASE TCP3     UNIX
  CICSR    CONFIG   GEN1D    SAVE     SCEECICD TCP3NFS
  COMM     DBASE    NFSO11OH SCEEBASD SCEECICS TCPIP
EZMSUPR::>cd ..
EZMSUPR::>ls
  AFP      COMM2    GEN1     PROD     SCEEBASE TCP3     UNIX
  CICSR    CONFIG   GEN1D    SAVE     SCEECICD TCP3NFS
  COMM     DBASE    NFSO11OH SCEEBASD SCEECICS TCPIP
EZMSUPR::>
```

Also, the only way to return to the VSE ″root″ (mounting point) is by:

```
EZMSUPR::>cd /u/vse
EZMSUPR:/u/vse:>ls
  AUTO    CICS    CU37XX  INFO    POWER   PRD2    SYS     VSE     VSESPUC
  AUXFILK COMMON  IJSYSRS NFSCAT  PRD1    STAN    VSAM    VSESP   VTAM
```

9.4.6 A Real Life Possible Scenario

We will now exercise a possible scenario in which a file will be created in VSE from a job submitted from Auto UNIX System. This file can be accessed in read mode in Auto UNIX System and it will be loaded into a Lotus123 spreadsheet as well.

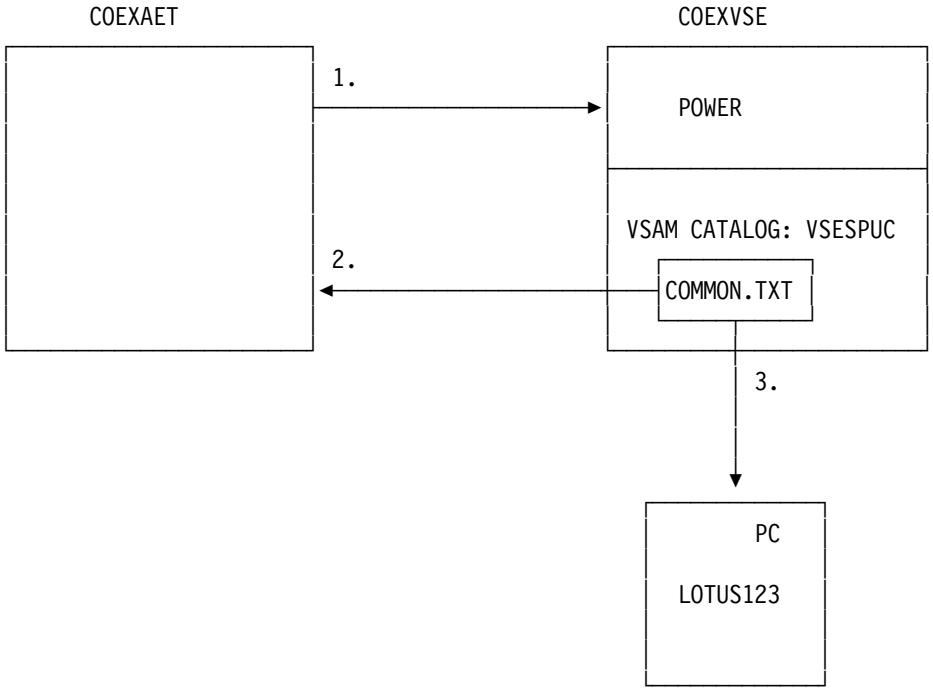


Figure 38. NFS Scenario 1

For the three mounts in Figure 38 the following commands were issued:

1. **ezmount -t nfs -p coexvse:/power.rdr.all vse.power /u/power**
2. **ezmount -t nfs -p coexvse:/vsespuc vse.vsam /u/vsam**
3. **mount i: vse:vsespuc**

We will submit the job to create a common file from Auto UNIX System, called COMMON.TXT, which, once created on VSE will be available to every system with NFS capabilities. As you can see, it is not necessary to change environments to transfer the file, as with FTP. We are already logged in, the current directory is /u/jobs.

The job is shown below, copied from the vi editor screens, in Figure 39 on page 89, Figure 40 on page 89, and Figure 41 on page 90.

```
* $$ JOB JNM=COMTABL,CLASS=0,DISP=D,NTFY=YES
// JOB COMTABL DEFINE FILE
// EXEC IDCAMS,SIZE=AUTO
 /*                                                              */
 /*  DELETE VSAM FILE                                            */
 /*                                                              */
     DELETE (COMMON.TXT) CL NOERASE PURGE  -
     CATALOG(VSESP.USER.CATALOG)
 DEFINE CLUSTER ( -
        NAME (COMMON.TXT           ) -
        CYLINDERS( 2  2 )              -
        SHAREOPTIONS ( 2 )             -
        RECORDSIZE (80 80 )            -
        VOLUMES ( DOSRES )             -
        NOREUSE                        -
        NONINDEXED                     -
        FREESPACE (15 7)               -
        NOCOMPRESSED                   -
        TO (99366  ))                  -
        DATA (NAME (COMMON.TXT.DATA ) -
        CONTROLINTERVALSIZE ( 4096 ))   -
        CATALOG ( VSESP.USER.CATALOG )
  IF LASTCC NE 0 THEN CANCEL JOB
/*
"create.common.table" 61 lines, 3837 characters
```

Figure 39. First vi Editor Screen

```
*
*    INITIALIZE COMMON TABLE
*
// DLBL COMTABL,'COMMON.TXT',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE   -
        (SYSIPT                 -
         ENVIRONMENT            -
          (RECORDFORMAT (FIXUNB) -
           BLOCKSIZE(80)         -
           RECORDSIZE (80)))     -
        OUTFILE (COMTABL)
01/31/98 130    1200.00
02/28/98 150    1400.00
03/31/98 160    1500.00
04/30/98 140    1300.00
05/31/98 150    1400.00
06/30/98 160    1500.00
07/31/98 180    1700.00
08/31/98 170    1600.00
09/30/98 170    1600.00
10/31/98 175    1650.00
11/30/98 180    1700.00
12/31/98 182    1720.00
```

Figure 40. Second vi Editor Screen

```
/*
// OPTION STDLABEL=DELETE
COMTABL
/*
// OPTION STDLABEL=ADD
// DLBL COMTABL,'COMMON.TXT',,VSAM,CAT=VSESPUC
/*
// EXEC IESVCLUP,SIZE=AUTO
D                                   COMTABL
A COMMON.TXT                        COMTABL VSESPUC
/*
* TO INVALIDATE BUFFERS AND FORCE NEW IO
// EXEC NFSUTIL,PARM='FLUSH VSESPUC    '
/&
* $$ EOJ
```

Figure 41. Third vi Editor Screen

To submit the job, we need to be connected to VSE/ESA NFS Daemon. To mount a hierarchical file structure (HFS) the following is necessary, (notice that we are creating an HFS at POWER mounting point):

```
EZMSUPR:/u/jobs:>ezmount -t nfs -p coexvse:/power.rdr.all vse.power /u/power
  EZM051I Mount will proceed asynchronously.

EZMSUPR:/u/jobs:>
```

After mounting the VSE/POWER file system (note that this can only be done by the superuser EZMSUPR for Auto UNIX System), submitting a job, is as simple as copying it to the target directory:

```
EZMSUPR:/u/jobs:>ls
  get.vse.vsam.file     vsamfile
  create.common.table   pausebg.job
  dvtdosrs.job          results.txt
EZMSUPR:/u/jobs:>cp create.common.table /u/power
EZMSUPR:/u/jobs:>
```

We are not going to discuss the ezmount command syntax, but it might be worth saying that **vse.power** is any name you choose, but must be unique in your system, and **/u/power** must specify an empty directory to where the mount point will be glued. The **vse.power** will be used later on by the superuser if it decides to unmount the HFS just mounted, the superuser will then issue the command: **ezunmount vse.power**:

```
EZMSUPR:/u/vse:>cd ..
EZMSUPR:/u:>ezunmount vse.power
  EZM100I Command processing was successful.
```

The superuser must make sure its current directory is not the one it is trying to unmount, otherwise the command will fail. After the job runs, this VSAM file will be usable by Auto UNIX System as a local file in directory /u/vse, just by entering the following at any Auto UNIX System terminal, logged in as the superuser (ezmsupr):

```
EZMSUPR:/u:>ezmount -t nfs -p coexvse:/vsespuc vse.vsam  /u/vsam
  EZM051I Mount will proceed asynchronously.

EZMSUPR:/u:>cd vse
EZMSUPR:/u/vsam:>ls
  AUTO.UNIX.ESDS                  DOS.WORKFILE.SYS002.RECOVER.F2
  CICS.AUTO.STATS.A               DOS.WORKFILE.SYSLNK.F8
  CICS.AUTO.STATS.B               MQSERIES.MQFSSET
  CICS.DUMPA                      CICS.DUMPB
  CICS.TD.INTRA                   VSE.PRIMARY.LIBRARY
  COMMON.TXT                      VTAM.DSDB1
  DEFAULT.MODEL.ESDS.SAM          VTAM.DSDB2
  DFHTEMP                         VTAM.DSDBCTL
  DIR.INCOMING.FILES              VTAM.TRSDB
  DOS.WORKFILE.SYS001.RECOVER.F2
EZMSUPR:/u/vsam:>
```

That is one of the benefits of the NFS protocol: to permit a portion of the server′s file system to be glued into the client′s file system. The client applications can transparently access the server′s data files and need not be aware of network semantics. The method of attaching the server file system or parts of it to the client file system is called ″mounting″ the server′s file system as mentioned before. Please refer to *TCP/IP for VSE/ESA User*, SC33-6601 for details. This, in other words means that you can use the file management commands of the client system to handle files: to copy, rename, erase and so on, as if they were really local and native files.

The file is VSAM ESDS, thus be aware that it will only support record appending. In our scenario we visualize applications in Auto UNIX System reading the file for local processing, as queries for instance, whereas in the PC workstation the file can be used, for example, as input to a spreadsheet program.

Important to note is that the file itself is VSAM ESDS, it will not support updating of any record. Do not save the spreadsheet back to the file, it will duplicate the records. If you edit the file using any text editor and save it, the file will have all records appended as well, duplicating them all at each save. As a suggestion to avoid record duplication you might consider defining the ESDS files as read only to NFS clients. Just as a reminder: it is done in the DEFINE FILE.

Now, please note that it is also possible to give access to a whole bunch of files if TYPE=VSAMCAT is selected. Be aware that the use of the VSAMCAT type of file has the ability to leave an open door to all files in the catalog. It may be a good idea to have a special catalog just for NFS access, defined with the READONLY=YES option for it, instead of having all files defined and specified as READONLY.

We understand that the first approach when using NFS for VSE and Auto UNIX System should be using VSE/ESA as a server, concerning VSAM files. Remember that the **only** type of VSAM files supported with NFS is ESDS.

We used LOTUS-123 to access the same file for the example. Let us take a look at the workstation and see if the file is there and try to access it. We can mount drive i: to the VSAM catalog:

```
[C:docs]
[C:\docs]mount i: vse:\vsespuc
mount: vse:\vsespuc
user name: wern
password:******

NFS Drive 'i:' was attached successfully.

[I:\]dir com*

The volume label in drive I is NFS.
The Volume Serial Number is 0100:068B.
Directory of I:\

12-07-98  9:22a           960      0 ----  COMMON.TXT
        1 file(s)            960 bytes used
                     536,870,912 bytes free

[I:\]
```

Yes, the file is there. We can now go to the desktop and click on the LOTUS123 icon which opens a .txt file from drive i:, as shown in Figure 42 on page 93.

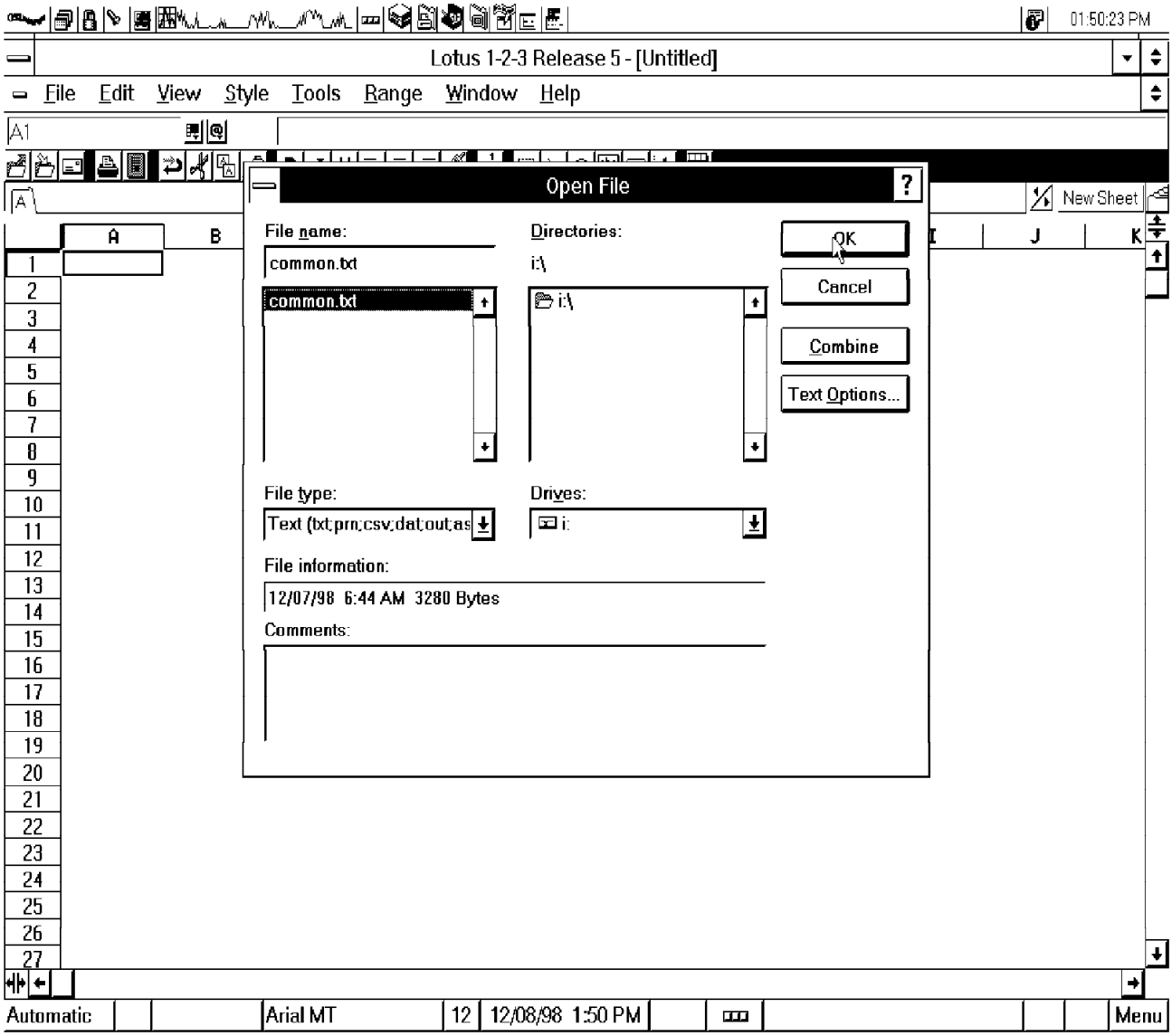


Figure 42. Opening the File as a Spreadsheet

Figure 43 on page 94 shows the file imported to the spreadsheet.

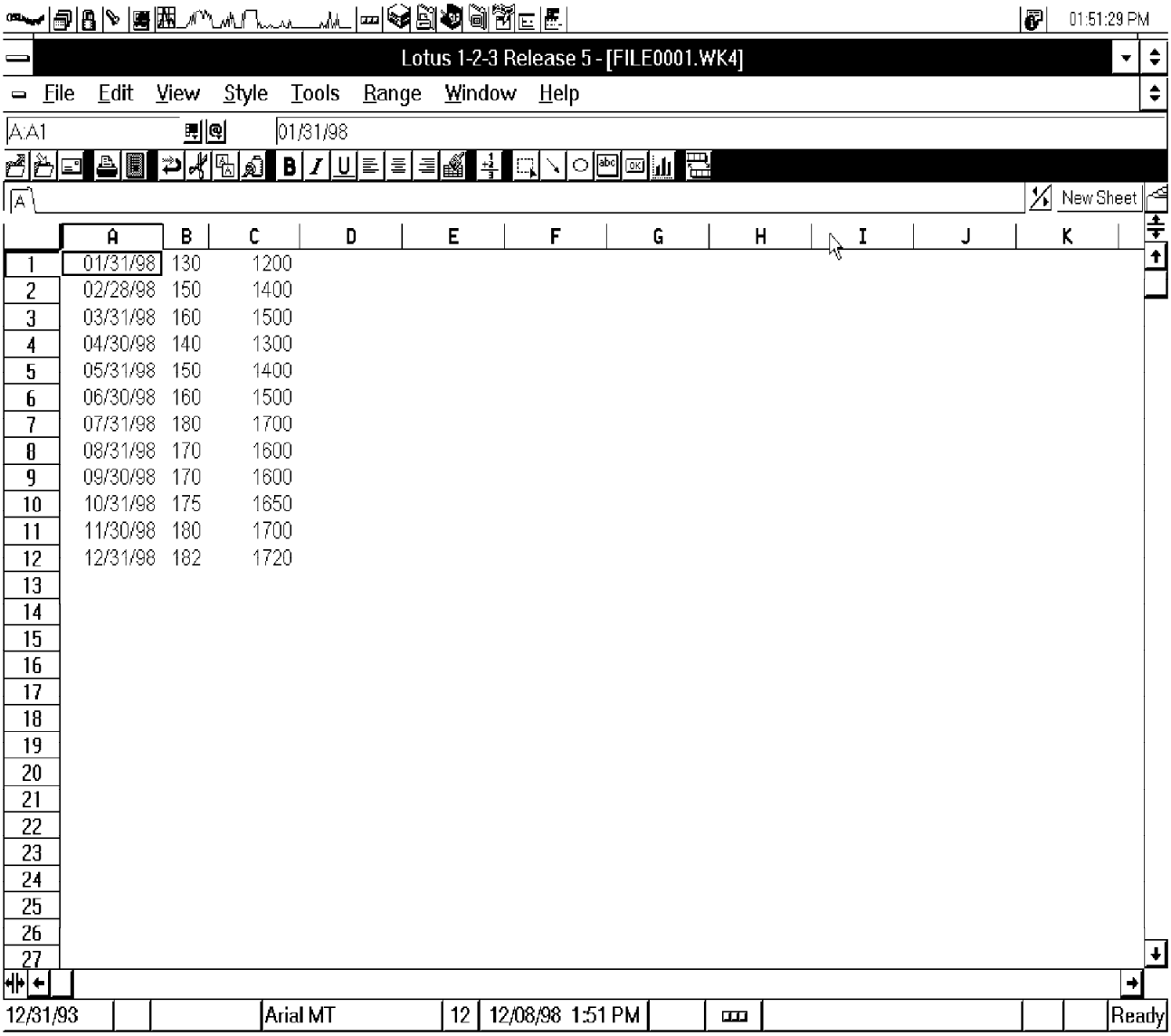


Figure 43. The VSAM File Imported to Lotus Spreadsheet

9.4.7 NFS Translation Tables

We have to comment on the miracle that is happening here. Maybe you have already figured it out, maybe not. The file is a VSAM file, created in VSE/ESA and its contents are in EBCDIC. But curiously enough when accessed by NFS protocol its data is translated to ASCII. There are many translation tables that come with the VSE/ESA NFS. The ones you wish to use have to be specified in the DEFINE NFSD. If you do not specify anything, the one called DEFAULT will be used.

When is translation used? That is easy. All file types that we wish to have translated must be set up in a table called NFSTYPES.L in source code format; it does not need compiling. If the files in our examples did not have their extensions such as .TXT or .JOB, but something else not in the table, when used in Auto UNIX System or in the workstation their contents would not have been translated to ASCII, rendering the files useless to remote users.

The opposite is true, also: saving files from clients (Auto UNIX System) with file extensions not of NFSTYPES will inhibit their use to VSE/ESA programs, since they will be in ASCII.

9.5 IDCAMS Invocation by NFS

You can customize IDCAMS cluster definition parameters. There is a member in the NFS sublibrary, called NFSMODEL.L where the user can set the site standards best suited for a particular installation. There you can set values for CATALOG, VOLUMES and so on, to be used whenever NFS needs to define a cluster automatically.

9.6 Automounting

In a production environment it may not be reasonable to have the superuser mount every mounting point at every system startup. Auto UNIX System provides a facility called AUTOMOUNTING, whereby the mounting points and authorities are specified in two files:

```
/etc/auto.master
```

and

```
/etc/MapName
```

The automount monitor is to be started at IPL time, and only after the mounting parameters are set up in those files mentioned above. Whenever any of the directories specified in /etc/auto.master, or in /etc/MapName are accessed, it will automatically be mounted for the user, and if so specified, also unmounted, based on some timeout criteria.

Chapter 10. MQSeries with Auto UNIX System and VSE/ESA

This topic introduces IBM MQSeries and describes its relationship with other products.

For more detailed explanations of these topics, refer to the MQSeries Reference manuals.

10.1 MQSeries and Message Queuing

MQSeries products enable applications to use message queuing to participate in message-driven processing. With message-driven processing, applications can communicate with each other on the same or different platforms, by using the appropriate message queuing software products. For example, Auto UNIX System and VSE/ESA applications can communicate through MQSeries for Auto UNIX System and MQSeries for VSE/ESA respectively. With MQSeries products, all applications use the same kind of messages; communications protocols are hidden from the applications.

10.2 MQI - a Common Application Programming Interface

MQSeries products implement a common application programming interface, the message queue interface (MQI), that is used on whatever platform the applications run on. The calls made by the applications and the messages they exchange are common. This makes it much easier to write and maintain applications than using traditional methods. It also facilitates the migration of message queuing applications from one platform to another.

10.3 Time-independent Applications

With message queuing, the exchange of messages between the sending and receiving programs is time independent. This means that the sending and receiving applications are decoupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message. The receiving application may be busy when the message is sent. Indeed, the receiving application doesn't even need to be running. MQSeries holds the message in the queue until it can be processed.

10.4 Message-driven Processing

Message-driven processing is a style of application design.

With this style, the application is divided into a number of separate, discrete, functional blocks, with each block having well-defined input and output parameters. Each functional block is coded as an application program, with its input and output parameters being interchanged between other application programs by placing their values in messages, which are then put on queues.

By use of the appropriate MQSeries programming mechanisms, an application program can start executing as a result of one or more messages arriving on a queue. If required, the program can terminate when all the messages in a queue have been processed.

This style of application design allows new applications to be built, or existing applications to be modified, more quickly than with some other application design styles.

10.5 Data Integrity and Resource Protection

MQSeries applications can transfer data with an extremely high degree of confidence. Message delivery can be implemented using a syncpoint mechanism and the MQSeries logs or journals for the recovery of important data in the event of system failure.

All resources, such as message queues, can be protected using the security facilities available on the operating platform, for example, Resource Access Control Facility (RACF) on Auto UNIX System, or the security facilities provided by VSE/ESA.

10.6 Messages and Queues

Messages and queues are basic to any queuing system.

10.6.1 What is a Message

A message is a string of bytes that has meaning to the applications that use the message.

In MQSeries, messages have two parts, a message descriptor and application data. The content and structure of the application data are defined by the application programs that use them. The message descriptor identifies the message and contains other control information or attributes, such as the date and time the message was created, the type of message, and the priority assigned to the message by the sending application.

10.6.2 What is a Queue

In physical terms, a queue is a type of list that is used to store messages until they are retrieved by an application.

Queues exist independently of the applications that use them. A queue can exist:

- In main storage if it is temporary.
- On disk or similar auxiliary storage if it must be kept in case of recovery.
- In both places if it is currently being used, and must also be kept for recovery.

Each queue belongs to a queue manager, which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue.

Queues can either be owned by the local queue manager, in which case they are called local queues, or can be owned by another queue manager, in which case they are called remote queues.

In MQSeries, messages can be retrieved from a queue by suitably authorized applications according to these retrieval algorithms:

- First-in-first-out (FIFO).
- Message priority, as defined in the message descriptor. Messages having the same priority are retrieved on a FIFO basis.
- A program request for a specific message.

For more information about queues and their attributes, refer to the *MQSeries Application Programming Guide*, SC33-0807, and *MQSeries for VSE/ESA User's Guide*, SC33-1142.

10.6.3 Message Attributes

For system administrators, messages have two important attributes that are defined in the message descriptor: persistence and priority.

A message is termed persistent if it survives when MQSeries restarts. This implies that the message must be logged, or saved, and can be reinstated as part of the recovery procedure.

Each MQSeries message has a priority assigned to it by the sending application. The priority, which is a number in the range 0 through 9, can affect the order in which a message is retrieved from a queue, and also the way that trigger events are generated.

Triggering is a facility in some MQSeries products. It allows an application to be started automatically when predefined conditions on a queue are met. These conditions include reception of any message or of messages over a particular priority, the number of messages on a queue, and so on. For more information about triggering, see the *MQSeries Application Programming Guide*, SC33-0807.

10.7 Auto UNIX System Provided Examples

Procedures to set up communications using MQSeries and to configure VSE/CICS for MQSeries are described in the manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993.

We followed the instructions and examples in the Auto UNIX System documentation and were able to successfully run the MQSeries sample using the following procedure.

10.8 Running the MQSeries Sample

1. From the VSE II menu, press PF6 to get a native VSE/CICS screen.
2. On the CICS screen, start transaction **SMP1.**
3. To start the sample transaction on Auto UNIX System:

   First sign on to UNIX with a user ID that has been defined in the UNIX group **ezmmqusr**, change to the directory that contains the MQSeries sample program, then at the UNIX prompt, enter:

   vsecics smp2 ′Hello Opa′

   The characters enclosed in quotes may be any input string with a maximum length of 96 characters. This string will be retrieved by CICS transaction SMP2.

If all has been set up correctly you will receive these messages at the UNIX shell:

```
REBECCA /etc/profile active
REBECCA:/u/rebecca:>cd hellomq
REBECCA:/u/rebecca/hellomq:>vsecics smp2 'Hello Opa'
'SMP2 Hello Opa' put to VSE1.REMOTEQ !
Output received from VSE/CICS:
SMP2 answered thanks for sending: Hello Opa

REBECCA:/u/rebecca/hellomq:>
```

Figure 44. UNIX Screen after Running MQSeries Sample

4 At the VSE/ESA console the following messages will be displayed.

```
F2 0002 SMP2 started by SMP1!
F2 0002 SMP2 answered thanks for sending: Hello Opa
```

Figure 45. VSE Console after Running MQSeries Sample

5 To stop the transaction running on VSE/ESA, on the UNIX shell, enter:

vsecics stop

During our set up and testing, we managed to get the two systems out of synchronization. To reset the environment we created, on the UNIX shell, the following MQSeries command file. This will reset the environment between Auto UNIX System MQSeries and the VSE/CICS MQSeries systems.

```
STOP CHANNEL(CSQ1.TO.VSE1)
STOP CHANNEL(VSE1.TO.CSQ1)
RESOLVE CHANNEL(CSQ1.TO.VSE1) ACTION(BACKOUT)
RESET CHANNEL(CSQ1.TO.VSE1) SEQNUM(1)
RESET CHANNEL(VSE1.TO.CSQ1) SEQNUM(1)
START CHANNEL(CSQ1.TO.VSE1)
START CHANNEL(VSE1.TO.CSQ1)
```

Figure 46. Reset VSE/CICS MQSeries Environment from UNIX

We created this command file with the UNIX vi editor, filed with a name of **reset.vsemq**. To use this command file and reset the MQSeries environment, we issued the command from the UNIX shell: **mq -i reset.vsemq**.

The "mq" command is provided with Auto UNIX System and is described in the manual *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993.

The MQSeries sample programs provided with Auto UNIX System are in the **/usr/lpp/ezm/samples/mqseries/vsecics/vse** subdirectory. Two programs are provided and they do the following:

vsecics is the program corresponding to transaction **SMP1**. The program is started by entering the transaction id **SMP1** from the CICS screen. The program is passed data from the Auto UNIX System MQSeries

system when the sample UNIX program is executed with the following command from the UNIX shell:

vsecics SMP2 ′Hello World!′.

SMP2 is the transaction id which will start a second transaction on the VSE/CICS system. The data enclosed in quotes will be passed to the program which is started. This VSE/CICS program then will be executed and puts its output to the queue that is read by the triggering Auto UNIX program.

If transaction id ′STOP′ will be received then it writes an acknowledgement back to Auto UNIX and then stops itself.

vsecics2 is the program corresponding to transaction **SMP2**.

This MQI sample program receives its input string from VSECICS MQ Server sample (which received this input from Auto UNIX via EXEC CICS RETRIEVE INTO call and then it sends an output string containing the received string back to Auto UNIX by directly putting it to the OE MQI queue (a remote queue for VSE).

10.9 MQSeries VSAM File Example

We modified the MQSeries sample provided with Auto UNIX System. On Auto UNIX System, to execute the sample program provided, you enter the command:

vsecics SMP2 ′Hello World!′.

In this command, **SMP2** is a CICS transaction id which will be executed on the VSE/ESA system. The data enclosed in quotes is passed to this transaction as an input stream.

We created a new transaction and program for the VSE/ESA CICS system. The transaction id was ″**INQK**″, and the program was **MQINQ**. The program is shown in Figure 72 on page 143.

This program which is called by transaction id INQK receives the input string that contains the key of a VSAM record. The program retrieves the VSAM record and passes it back to the Auto UNIX System MQSeries program in the MQI queue (a remote queue for VSE). The Auto UNIX System MQSeries program displays the VSAM record on the UNIX terminal.

10.10 Define CICS Resources for Sample Program

To run the new sample program on the VSE/ESA CICS system we needed to define the following:

1. Define and initialize the VSAM file

   We used the job stream described in Figure 73 on page 150 to define the VSAM file and initialize it. This job stream also adds the label for this file to the VSE/ESA standard label area. This is a KSDS VSAM file with 80 byte records and the key is in the first two bytes of the record.

2. Define the VSAM file in the CICS FCT.

   The CICS file control table (FCT) which we used to define the MQSeries files as well as our sample VSAM file is shown in Figure 74 on page 151.

3 Define the FCT in the CICS SIT.

THE CICS system initialization table (SIT) which we used to define our CICS start up is shown in Figure 75 on page 154.

4 Define the transaction and program to CICS.

We created a job to define our new transaction and program to CICS. Also included in this job was the definition of the two CICS PLT programs which autostart and shutdown the MQSeries for VSE/ESA manager. The two PLT entries are named DFHPLTMI and DFHPLTMS. The CSD definition job stream is shown in Figure 76 on page 156.

5 Create a CICS startup job stream.

The CICS startup job called CICSMQ is shown in Figure 77 on page 157.

10.11 CICS MQSeries VSAM Sample Execution

The execution of our new program follows.

1 From the VSE II menu, press PF6 to get a native VSE/CICS screen.

2 On the CICS screen, start transaction **SMP1.**

3 To start the sample transaction on Auto UNIX System:

First sign on to UNIX with a user ID that has been defined in the UNIX group **ezmmqusr**, change to the directory that contains the MQSeries sample program, then at the UNIX prompt, enter:

vsecics inqk ′nn′

The **nn** enclosed in quotes is the key of the VSAM record you would like to retrieve from VSE CICS. This string will be retrieved by CICS transaction INQK.

If all has been set up correctly you will receive these messages at the UNIX shell:

```
REBECCA:/u/rebecca/hellomq:>vsecics inqk ′10′
′INQK 10′ put to VSE1.REMOTEQ !
Output received from VSE/CICS:
INQK10 RECORD TEN
REBECCA:/u/rebecca/hellomq:>
```

Figure 47. UNIX Screen after Running MQSeries VSAM Inquiry Program

Chapter 11. VSE/ESA on the ′Net Using HTTP′

TCP/IP for VSE/ESA includes a Hyper Text Transfer Protocol (HTTP) Daemon to support the use of ″Web Browsers″ to access a variety of data residing on your VSE system. This data may include linked pages of text, graphic images, sound recordings, and video.

In combination with the Auto UNIX System Web facilities, VSE/ESA customers are able to link to Auto UNIX System and have full OS/390 functions available to the VSE/ESA system.

OS/390 Version 2 provides the technologies and services required to conduct true and secured e-Business. OS/390 connects organizations, customers and business partners so they will be able to communicate and carry out e-Business transactions. Version 2 provides this robust capability using Domino Go Webserver for OS/390 which is integrated as a base element in Version 2. It includes the Internet Connection Secure Server (ICSS) and NetQuestion, which is a robust text search engine. OS/390 also delivers an enhanced Communications Server and the integrated support for the IBM Network Station.

OS/390 Version 2 offers maximum security for conducting e-Business over the Web. Version 2 provides access control, including an auditing mechanism and authentication (SSL Version 3), encryption (ICSF - Integrated Cryptographic Service Facility), a Security Server, and integrated Firewall Technologies.

OS/390 Version 2 software technology delivers the most scalable web server in the industry that can easily grow to meet an organization′s expanding e-Business needs. Version 2 contains full web-serving support for Parallel Sysplex and an enhanced Workload Manager (WLM) domain name support (DNS) for improved web-based workload balancing across the parallel sysplex environment.

OS/390 Version 2 offers the most extensive information management tools in the industry, enabling companies to effectively manage their enterprise-wide Internet and intranet information. Version 2 enables customers to easily access, store and search softcopy publications using OS/390′s BookServer and the NetQuestion functions of Domino Go Webserver for OS/390.

11.1 What is HTML?

To interchange Hyper Text Markup Language (HTML) documents, HTTP is used. An HTML document is a file that contains printable text, interspersed with HTML ″tags″ that describe the document to be displayed. Additional elements of HTML allow you to include links to other documents, embedded graphics, and special effects.

There are many HTML texts available on the Internet. Although the HTML language is powerful, the entire HTML language can be summarized on about three pages. You should be able to write a simple HTML document within a few minutes of starting. We have included a sample to get you started.

```
* $$ JOB JNM=INDEX
// JOB INDEX CATALOG INDEX.HTML
// EXEC LIBR
ACC S=PRD2.TCPIP
CATALOG INDEX.HTML                          REPLACE=YES
<html>
<title>
VSE/ESA Just do it!
</title>


<body Bgcolor="#98B8F8">

<img src="ezmhead.gif">

<h1>Welcome to VSE/ESA Web Server         </h1>

Some links to functions on this server:
<ul>
<li><a href="http://9.164.155.62">Auto Unix Web Page</a>
<li><a href="http://stanweeks.com">Help from the expert!</a>
<li><a href="http://www.direcao.com.br">More from Brazil!</a>
</ul>
</body>
</html>

/+
/*
/&
* $$ EOJ
```

Figure 48. Sample INDEX.HTML Job Stream

TCP/IP for VSE/ESA provides special HTML files to allow security for your Web Server.

PASSWORD.HTML

VIOLATED.HTML

BLANKING.HTML

The member HTMLINST.Z in PRD1.BASE contains a jobstream which generates default members of these special HTML files. Refer to *TCP/IP for VSE/ESA User*, SC33-6601.

11.2 Testing Documents

Most web browsers will permit you to open a document on disk. This is much easier than placing each revision on a web site before you test it. You may find it useful to run a text editor in one window and your browser in another. Then, as you revise and save your document, you just click "reload" from the browser to refresh the display.

11.3 The HTTP Daemon

The HTTP Daemon is defined and started with the DEFINE HTTPD command, entered from the VSE console or as part of the TCP/IP for VSE/ESA initialization stream. This Daemon receives all HTTP requests (by virtue of its assigned input port), determines the resource needed to satisfy the request, and transmits any resulting data back to the requester.

When the HTTP Daemon is started, there are parameters which directly affect how it will interact with its clients (browsers). For a full description of the command, see the *TCP/IP for VSE/ESA User*, SC33-6601.

11.3.1 PORT

The port assigned to the HTTP Daemon is, by default, port 80. All web browsers will use this port by default. Some browsers will permit you to select a different port. Changing the port, if practical, will give you a fair amount of security and allow you to operate several web sites concurrently. If you are accessible to external networks, the casual ″net surfer″ will probably not be trying random port numbers to locate your ″site″.

11.3.2 ROOT

The basic HTTP request is for an HTML document. Documents are stored in files. These files are part of the TCP/IP for VSE/ESA file system and are identified by their ″Public Names″. When a ″root″ is specified to the HTTP Daemon, it specifies the library and sublibrary where the HTML documents will reside. The user does not need to have knowledge of your file system′s structure and provide a fully-qualified name with their requests.

For example, if the root is specified as ″PRD2.TESTLIB.HTML″ (and this Public Name indeed corresponds to a library and sublibrary) then all HTTP requests will be taken from this library.

11.3.3 CONFINE

The CONFINE parameter works with the ROOT parameter. In the case where a request is received for an HTTP document and the file cannot be found, the CONFINE value is examined. The default value is CONFINE=YES. **Be very careful with CONFINE=NO. With CONFINE=NO, the requestor may have access to all of the data on your VSE/ESA system.** If CONFINE=NO was specified, the file search is performed once more but without using the ROOT specification. In this way, the requester may specify the fully-qualified Public Name of an HTTP document.

11.3.4 SECURE

The HTTP server executes in ″read-only″ mode so write-access security is not an issue. Since the HTTP protocol makes no provision for user ID or password, the most that can be accomplished is to confine access to a specific sub-directory tree of the file system. This may be implemented by use of the ROOT and CONFINE parameters of the DEFINE HTTPD command.

The basis of TCP/IP for VSE/ESA security technique are three ″special″ HTML documents and a table of currently authorized network addresses. Suppose that HTTPD has been defined with SECURE=YES, then whenever a document request is received from a web browser at an unknown (not in the table) IP

address, the special document ″PASSWORD.HTML″ is sent instead. This document prompts the user for a user ID and password. These values are then verified by the standard routines. If the values are verified, the user′s IP address is placed in the authorized table and the originally-requested document is transmitted. Otherwise, the ″VIOLATED.HTML″ document is sent.

Once authorized, the user is free to request additional documents until his IP address is removed from the table. Removal occurs under two conditions: When an explicit request is made for the document ″BLANKING.HTML″ and upon the expiration of a time-out value with no activity (no documents requested). The time-out value may be specified on the define HTTP Daemon command.

11.4 Web Browsers

All web browsers generally function in a similar fashion. Carefully evaluate their capabilities and compatibilities before either choosing a ″standard″ or deciding which HTML features you will code in your web pages.

11.5 URL Formats

Several URL (Universal Resource Locator) formats can be used to access documents residing on TCP/IP for VSE/ESA′s HTTP Daemon. Examples are:

http://9.164.155.61

In the above, 9.164.155.61 is the IP address assigned to TCP/IP for VSE/ESA on our VSE system. If your network has a domain name server, your may replace this address with its symbolic name. Since no document was specifically requested, the (default) document named INDEX.HTML is returned. Note any root specification made at HTTP Daemon initialization will be prefixed to INDEX.HTML before the search is performed.

http://nn.nn.nn.nn/abc.html

In the above example, everything remains the same except a specific document has been requested. Again, the root specification will be prefixed to the document name.

http://nn.nn.nn.nn/sublib

In the above example, we have specified a sublibrary but no document. Assuming the root value, when prefixed to ″SUBLIB″, forms a fully-qualified Public Name, then member INDEX.HTML is returned.

http://nn.nn.nn.nn/sublib/index.html

This example is the same as the last one, except the document is explicitly requested.

http://nn.nn.nn.nn/prd2/testlib/web/index.html

In the above, the document INDEX.HTML is requested from the library with public name: PRD2.TESTLIB.WEB. This assumes either no root was specified to the Daemon or CONFINE=NO was coded.

11.6 Creating a Web Site

Your web site will be created and maintained in the same manner whether it is accessible via Intranet or Internet. You will need to create a set of HTML documents, possibly with embedded links to other objects (graphics and so on). These will need to be placed into the library defined to the HTTP Daemon. Multiple web sites can be created by the use of sub-directories or by defining multiple HTTP Daemons, each with its own port number. TCP/IP for VSE/ESA does not supply an editor to create HTML documents. The editing of HTML documents is typically done on a PC.

11.7 HTML Documents

HTML documents are created as standard text files with embedded HTML tags. They should have a file extension of HTML. The members′ names are arbitrary, but you should provide a member named INDEX.HTML since this is the default document, fetched when no document name is supplied. Your documents should be stored in EBCDIC, since they will be translated to ASCII during transmission. They will go into the library as fixed-length 80-byte records. Under most circumstances this should pose no problems since web browsers reformat the text anyway. If you must generate lines longer than 80 bytes (for example, you need to display a line from a 132-byte listing) you may end a line with the ″escape″ character (as defined to the HTTP Daemon during initialization) and then continue on the next line. The Daemon will note the escape character, remove it, and concatenate the next line whenever the member is transmitted.

11.8 Graphic Images

Many web pages will include graphic images. This is done by means of an HTML tag that references a member containing a graphic. Common types of graphics include GIF and JPG encoding. GIF files convert quickly to a screen display while JPG files are highly compressed for rapid transmittal. Other types of graphic encoding are also used. The types you choose will depend upon two considerations: the formats you can generate and the formats supported by your web browser.

Regardless of the graphic formats you use, the HTTP Daemon determines the graphic type by way of the file type. The Daemon then generates a file header record that notifies the browser of the graphic type.

In addition to static images, browsers may also support video, audio, plain text, and virtual reality displays.

11.8.1 Translation

Graphic images are stored in binary form and are not translated.

11.8.2 Preparing Graphic Images

You may prepare graphics with any software you desire. Typically, this will be done on a PC. Once an image has been created, you must place it in the appropriate VSE library.

The graphic image is stored in a ″string″ member, the same as a VSE phase (or other binary data) would be stored. The easiest way to load your graphic into a

member is to use FTP. Be sure your FTP client transmits the file with TYPE set to BINARY and record format set to ″S″.

Note: If your PC-based FTP client has no provision for issuing SITE commands, perform the transfer using TCP/IP for VSE/ESA′s CICS or batch FTP client.

Appendix A. Sample VM/ESA Directory and Profile EXEC

```
USER COEXAET JENS 256M 512M G
   CPU 00 CPUID 000001 NODEDICATE
   CPU 01 CPUID 100001 NODEDICATE
   CPU 02 CPUID 200001 NODEDICATE
   IPL CMS PARM AUTOCR
   MACH ESA
   OPTION MAXCONN 15 QUICKDSP MAINTCCW TODENABLE DEVMAINT
   CONSOLE 0009 3215 T
   SPOOL 000C 2540 READER A
   SPOOL 000D 2540 PUNCH A
   SPOOL 000E 3800 A
   LINK MAINT 0190 0190 RR
   LINK MAINT 019E 019E RR
   MDISK 0191 3390 2286 10 VM3V03 MR R W M
* 3390 MOD 1 DASD FOR COEXAET SYSTEMS
   MDISK 0129 3390 2226 1113 VM3V00 MR R W M
   MDISK 012A 3390 2226 1113 VM3V01 MR R W M
   MDISK 012B 3390 2226 1113 VM3V02 MR R W M
* 3390 MOD 2 DASD FOR COEXAET SYSTEMS
   MDISK 0120 3390 1 2225 VM3V00 MR R W M
   MDISK 0121 3390 1 2225 VM3V01 MR R W M
   MDISK 0122 3390 1 2225 VM3V02 MR R W M
   MDISK 0123 3390 1 2225 VM3V03 MR R W M
   MDISK 0128 3390 1 2225 VM3V08 MR R W M
* 3390 MOD 3 DASD FOR COEXAET SYSTEMS
   MDISK 0124 3390 1 3338 VM3V04 MR R W M
   MDISK 0125 3390 1 3338 VM3V05 MR R W M
   MDISK 0126 3390 1 3338 VM3V06 MR R W M
   MDISK 0127 3390 1 3338 VM3V07 MR R W M
*DVHOPT LNK1 LOG0 RCM1 SMS0 NPW1 LNGAMENG PWC19981023 CRC¥
```

Figure 49. Sample VM/ESA Directory for Auto UNIX System

```
/* PROFILE EXEC for Auto UNIX System guest system */
/* COEXAET                                        */
   Address 'COMMAND'
   Trace 'N'
/* Ensure that some defaults settings are effective */
   'CP SET LINEDIT ON'
   'CP SET CONCEAL OFF'
   'CP TERMINAL LINEND ON'
   'CP TERMINAL BRKKEY PA1'
   'SET FULLSCREEN OFF'

/* Other settings */
   'CP SET SVC76 CP'          /* error recording by CP */
   'CP SET MIH OFF'           /* no MIH processing by CP */

/* Set up GRAFs for use as local terminals */
   Call diag 8,'DETACH 700-702'
   'CP DEF GRAF 700'
   'CP DEF GRAF 701'
   'CP DEF GRAF 702'
   'CP DEF GRAF 703'
   'CP DEF GRAF 704'

/* Set up CTCAs */
   Call diag 8,'DETACH 600-601'
   Call diag 8,'DETACH 500-501'
   Call diag 8,'DETACH 660-661'
   'CP DEF CTCA 500'
   'CP DEF CTCA 501'
   'CP COUPLE 500 TO COEXVSE 640'
   'CP COUPLE 501 TO COEXVSE 641'
/*  CTCAs 600 and 601 are for VTAM TCP IP                  */
   'CP DEF CTCA 600'
   'CP DEF CTCA 601'
   'CP COUPLE 600 TO TCPIP  E04'
   'CP COUPLE 601 TO TCPIP  E05'

/* exit to CMS command prompt */
   'VMFCLEAR'
Exit 0
```

Figure 50. Sample Profile EXEC for Auto UNIX System

Appendix B. Figures and Samples for VSE/ESA Setup

VM CTCA Connections

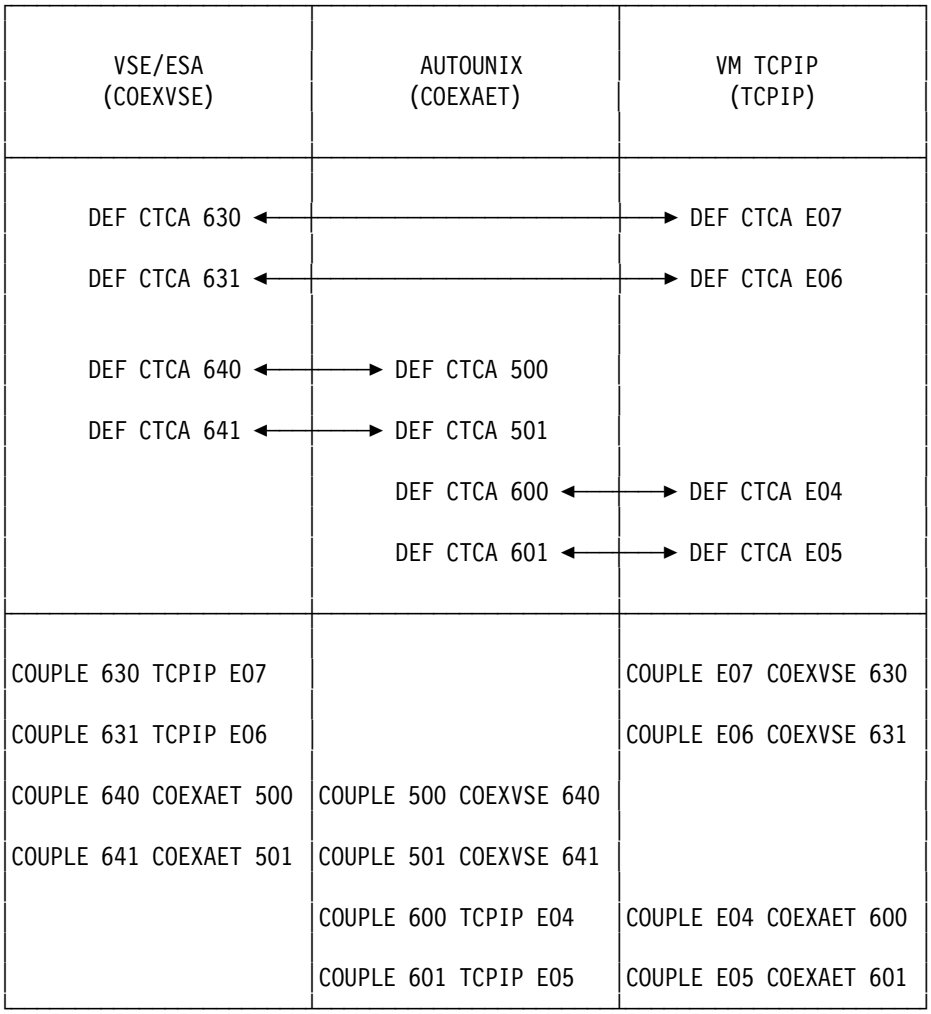


Figure 51. VM CTCA Connections

Note: The upper part of the figure shows the virtual CTC connections. The lower part of the figure shows the corresponding control statement definitions for the same CTC connections (couple).

VSE/ESA VM/ESA TCPIP Connections

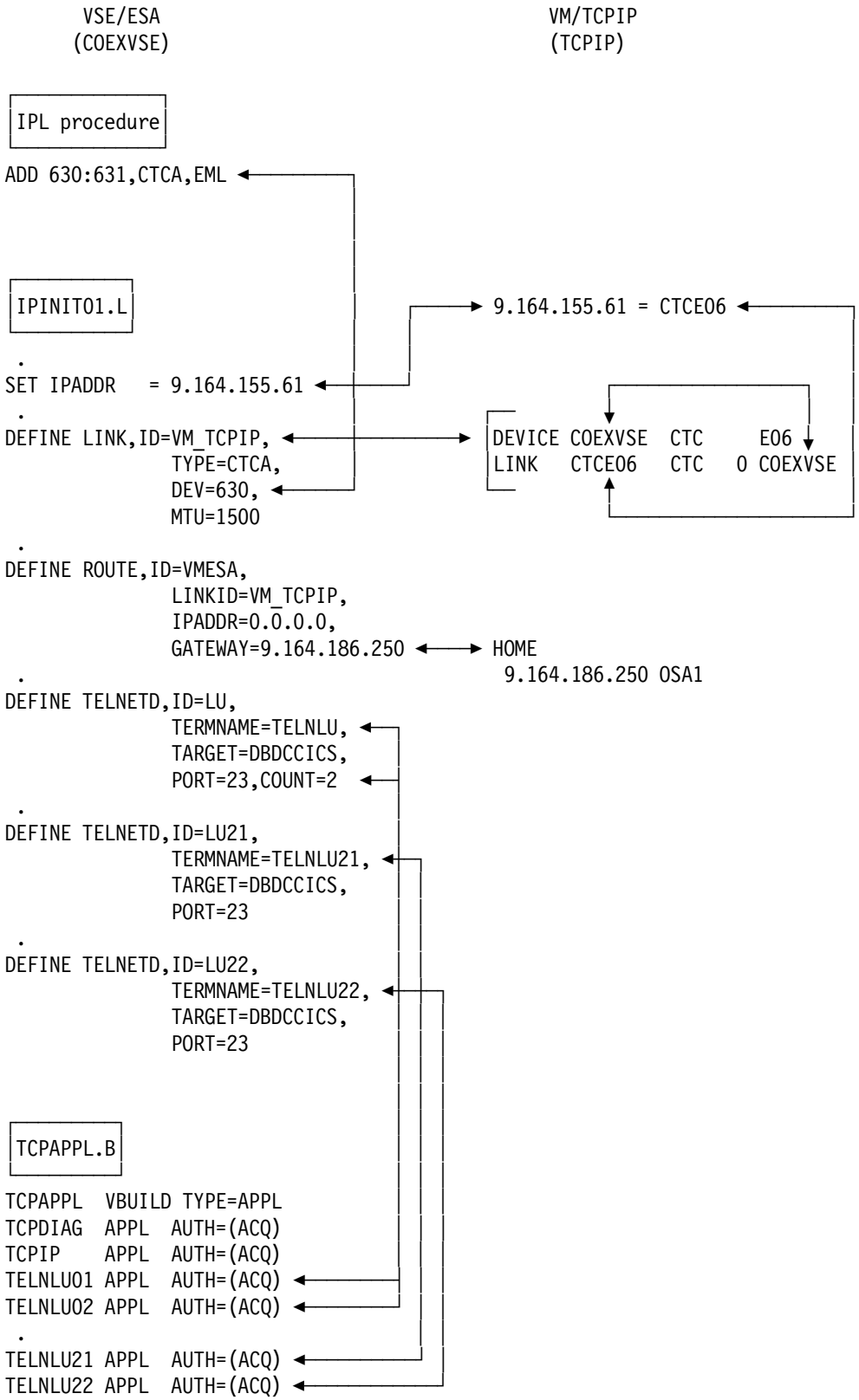


Figure 52. VSE/ESA VM/ESA TCPIP Connections

VSE/ESA VTAM Definitions

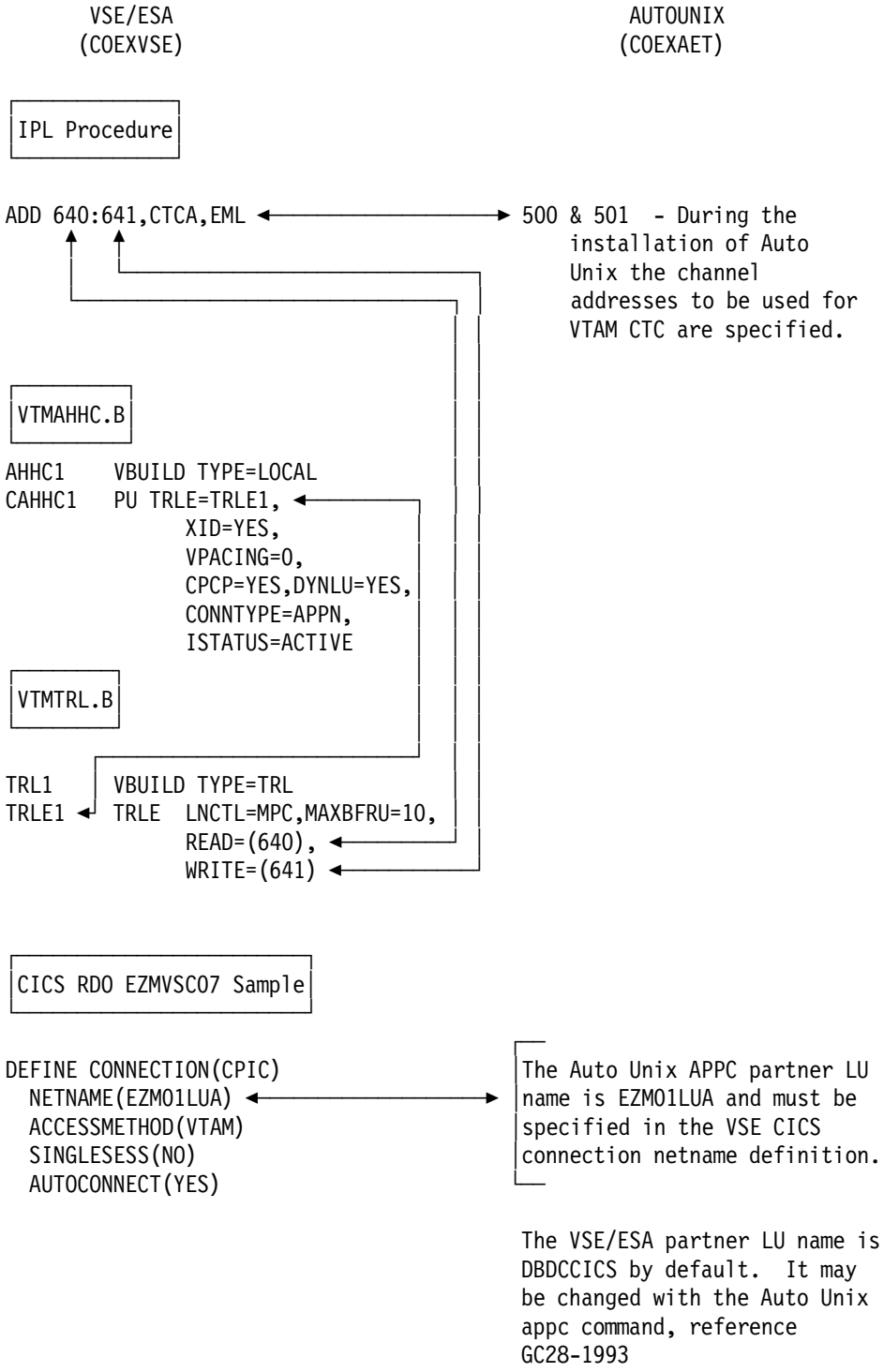


Figure 53. VSE/ESA VTAM Definitions

```
* $$ JOB JNM=AUXDFLIB,CLASS=O
* $$ LST CLASS=A
// JOB CRELIB CREATE SUBLIB FOR UNIX VTAM DEFINITIONS
// EXEC LIBR
DEF S=PRD2.UNIX
/*
/&
* $$ EOJ
```

Figure 54. Sample Job to Create PRD2.UNIX Sublibrary

```
* $$ JOB JNM=VTAMUNIX,CLASS=3,DISP=L
* $$ LST CLASS=A,DISP=D
// JOB VTAMUNIX START UP VTAM WITH AUTO UNIX CONNECTION
// OPTION DUMP,SADUMP=5
// SETPARM XNCPU=''
// EXEC PROC=$COMVAR,XNCPU
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF3=ACTIVE'
/*
// SETPFIX LIMIT=424K
// ASSGN SYS000,UA
// ASSGN SYS001,DISK,VOL=SYSWK1,SHR     TRACE FILE ASSIGNMENT
// ASSGN SYS004,DISK,VOL=SYSWK1,SHR     TRACE FILE ASSIGNMENT
// ASSGN SYS005,DISK,VOL=SYSWK1,SHR     NCP LOAD/DIAG FILE ASSGN
// LIBDEF *,SEARCH=(PRD2.UNIX,PRD2.COMM,PRD2.COMM2,PRD2.CONFIG,
               PRD1.BASED,PRD1.BASE),PERM
// LIBDEF DUMP,CATALOG=SYSDUMP.F3
// EXEC ISTINCVT,SIZE=ISTINCVT,PARM='CUSTNO=C555-555-5555,VTAMPW=5979-4*
               015-4627-6185-9388',DSPACE=2M
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF3=INACTIVE'
/*
/&
* $$ EOJ
```

Figure 55. Sample Job for VTAM Startup

```
* $$ JOB JNM=CICSUNIX,DISP=L,CLASS=2,EOJMSG=YES
* $$ LST CLASS=A,DISP=D,RBS=100
// JOB CICSUNIX STARTUP WITH CONNECTION TO AUTO UNIX
// OPTION SADUMP=5
// UPSI 11100000
// LIBDEF *,SEARCH=PRD2.UNIX
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASED,PRD1.BASE,PRD2.PROD,        X
               PRD2.SCEECICD,PRD2.SCEECICS,PRD2.SCEEBASD,            X
               PRD2.SCEEBASE,PRD2.CICSR,PRD2.DBASE,                  X
               PRD1.MACLIBD,PRD1.MACLIB),PERM
// LIBDEF DUMP,CATALOG=SYSDUMP.F2
// SETPARM TPMODE=''
// SETPARM XNCPU=''
// SETPARM XMODEF2=''
// EXEC PROC=$COMVAR,XNCPU
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF2=ACTIVE'
/*
// EXEC PROC=CPUVAR&XNCPU,TPMODE,XMODEF2
// SETPFIX LIMIT=144K
// EXEC PROC=DTRCICST                  ASSGNS FOR CICS FILES
// EXEC PROC=DTRINFOA                  ASSGNS FOR INFO ANAL FILES
// EXEC PROC=DTRICCF                   ASSGN FOR DTSFILE
// ASSGN SYS005,UA
// ASSGN SYS006,UA
// ASSGN SYS007,UA
// ASSGN SYS008,UA
// ASSGN SYS009,SYSLOG
LOG
// ID USER=FORSEC,PWD=FORSEC
NOLOG
// EXEC DTSANALS                       RECOVER IF DTSFILE DESTROYED
RECOVER OPT
/*
// IF TPMODE=B THEN
// GOTO BTAM
*   WAITING FOR VTAM TO COME UP
// EXEC IESWAITT
// GOTO START
/. BTAM
// ASSGN SYS023,IGN
// ASSGN SYS024,IGN
// ASSGN SYS025,IGN
// ASSGN SYS026,IGN
// ASSGN SYS027,IGN
// EXEC PROC=DTRIBTAM                  ASSGN BTAM TERMINALS
* AT THIS POINT, YOU MAY OPTIONALLY ASSIGN SYS027 TO
* A 3270 PRINTER SUPPORTED AS A 3286. ENTER:
* "// ASSGN SYS027,CUU"  WHERE CUU IS THE PRINTER ADDRESS
// PAUSE
/. START
// IF XMODEF2 = COLD THEN
// GOTO COLD
```

Figure 56 (Part 1 of 2). Sample Job for CICS Startup

```
// EXEC DFHSIP,PARM='SIT=SP,GRPLIST=VSELIST,ISC=YES,$END',DSPACE=2M
/*
// GOTO END
/. COLD
// EXEC DFHSIP,PARM='SIT=SP,GRPLIST=VSELIST,START=COLD,ISC=YES,$END',
              DSPACE=2M
/*
/. END
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF2=INACTIVE'
/*
/&
* $$ EOJ
```

Figure 56 (Part 2 of 2). Sample Job for CICS Startup

```
* $$ JOB JNM=EZMVSC07,CLASS=O,DISP=D
// JOB EZMVSC07 DEFINE RESOURCES TO CICS CSD FOR LU62 CONNECTION (CPIC
* ********************************************************************
*                                                                    *
* UPDATE CICS CSD WITH CONNECTION AND SESSION INFORMATION            *
* TO COMMUNICATE WITH CPIC ON  LU = EZMO1LUA                         *
*                                                                    *
* MAKE YOUR ADAPTATIONS WHERE NECESSARY                              *
*                                                                    *
* ********************************************************************
* ------------------------------------------------------------------*
* UPDATE CICS CSD WITH DEFINITIONS FOR CONNECTIONS AND SESSIONS    *
* ------------------------------------------------------------------*
// EXEC DFHCSDUP
*
DELETE GROUP(GRPCPIC)
* ------------------------------------------------------------------*
*  DEFINITIONS FOR APPC (CPI-C)
* ------------------------------------------------------------------*
*  DEFINITIONS FOR APPC (CPI-C)
* ------------------------------------------------------------------*
DEFINE CONNECTION(CPIC)      GROUP(GRPCPIC)
  NETNAME(EZMO1LUA)
  ACCESSMETHOD(VTAM)     PROTOCOL(APPC)
  SINGLESESS(NO)         DATASTREAM(USER)       RECORDFORMAT(U)
  AUTOCONNECT(YES)       INSERVICE(YES)         ATTACHSEC(LOCAL)
DEFINE SESSIONS(SESCPIC)     GROUP(GRPCPIC)
  CONNECTION(CPIC)
  MODENAME(#INTER)
  PROTOCOL(APPC)         MAXIMUM(6,3)
  SENDSIZE(4096)         RECEIVESIZE(4096)      SESSPRIORITY(0)
  AUTOCONNECT(YES)       BUILDCHAIN(YES)        USERAREALEN(0)
  IOAREALEN(0,0)         RELREQ(NO)             DISCREQ(NO)
  NEPCLASS(0)            RECOVOPTION(SYSDEFAULT)
* ------------------------------------------------------------------*
* ADD LU62 GROUP TO THE STANDARD VSE/ESA LIST.
* ------------------------------------------------------------------*
 ADD GROUP(GRPCPIC) LIST(VSELIST)
/*
/&
* $$ EOJ
```

Figure 57. Auto UNIX System Sample EZMVSC07

```
* $$ JOB JNM=EZMVSV05,CLASS=0,DISP=L
// JOB EZMVSV05  DEFINE RESOURCES TO CICS FOR CPIC SAMPLE
* ---------------------------------------------------------------------*
* UPDATE CICS CSD WITH DEFINITIONS FOR THE HELLO WORLD SAMPLE
* (TRANSACTION HELL)
* ---------------------------------------------------------------------*
// EXEC DFHCSDUP
*
DELETE GROUP(EZMHELLO)
* ---------------------------------------------------------------------*
* DEFINE PROGRAM CICSHELL AS C PROGRAM TO CICS                         *
* ---------------------------------------------------------------------*
DEFINE PROGRAM(CICSHELL) GROUP(EZMHELLO)
       LANGUAGE(C) RSL(PUBLIC)
* ---------------------------------------------------------------------*
* DEFINE CICSHELL TRANSACTION (PARTNER TP NAME OF AUTO UNIX )          *
* ---------------------------------------------------------------------*
DEFINE TRANSACTION(HELL) GROUP(EZMHELLO) PROGRAM(CICSHELL)
* ---------------------------------------------------------------------*
* ADD EZMHELLO GROUP TO THE STANDARD VSE/ESA LIST.                     *
* ---------------------------------------------------------------------*
ADD GROUP(EZMHELLO) LIST(VSELIST)

/*
/&
* $$ EOJ
```

Figure 58. Auto UNIX System Sample EZMVSV05

```
* $$ JOB JNM=CICSHELL,DISP=D,CLASS=8,NTFY=YES
* $$ LST DISP=D,CLASS=Q,PRI=3
* $$ PUN DISP=I,DEST=*,PRI=9,CLASS=8
// JOB CICSHEL TRANSLATE PROGRAM CICSHELL
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
$ $$ LST DISP=D,CLASS=Q,PRI=3
// JOB CICSHELL COMPILE PROGRAM CICSHELL
// DLBL SYSMSGS,'C370.COMP.MSGS',0,VSAM,RECSIZE=3000,RECORDS=35,       X
               CAT=VSESPUC
// LIBDEF *,SEARCH=(PRD2.SCEEBASE,PRD2.DBASE)
// SETPARM CATALOG=1
// IF CATALOG = 1 THEN
// GOTO CAT
// OPTION ERRS,SXREF,SYM,LIST,NODECK
// GOTO ENDCAT
/. CAT
// LIBDEF PHASE,CATALOG=PRD2.UNIX
// OPTION ERRS,SXREF,SYM,CATAL,NODECK
   PHASE CICSHELL,*
   INCLUDE DFHELII
/. ENDCAT
// EXEC EDCCOMP,SIZE=EDCCOMP,PARM='NATLANG(ENU)/LONGNAME'
* $$ END
// ON $CANCEL OR $ABEND GOTO ENDJ2
// OPTION NOLIST,NODUMP,DECK
// EXEC DFHEDP1$,SIZE=512K
 /* START OF SPECIFICATIONS ****************************************/
 /*                                                                */
 /*01* MODULE-NAME: HELLOD (CORRESPONDING PARTNER PROGRAM HELLO.C)  */
 /*                                                                */
 /*02*   DESCRIPTIVE-NAME: HELLOD.C                               */
 /*                                                                */
 /*03* TRANSACTION ID: HELL DEFINED IN EZMVSV05.JCL                */
 /*                                                                */
 /* 5647 - A01                                                     */
 /*                                                                */
 /* (C) COPYRIGHT IBM CORP. 1997,1998                              */
 /*                                                                */
 /* STATUS = HEZ6606                                               */
 /*                                                                */
 /*01* FUNCTION:  THIS VSE/CICS APPC SERVER (INBOUND PROGRAM)      */
 /*               RECEIVES A "HELLO WORLD !" STRING FROM           */
 /*               AUTO UNIX                                        */
 /*               AND RETURNS "HELLO AUTO UNIX !" TO ITS HELLO.C   */
 /*               COUNTERPART RUNNING UNDER AUTO UNIX.             */
 /*                                                                */
 /*      SYNTAX:  WILL NOT BE CALLED BY OPERATOR, BUT FROM CICS    */
 /*               DRIVEN BY AUTO UNIX PARTNER PROGRAM.             */
 /*02*   CALLERS:                                                  */
 /*                                                                */
 /*    THIS MODULE MUST BE COMPILED AND LINKED IN A VSE BATCH      */
 /*    PARTITION. JUST RELEASE THIS JOB.                           */
```

Figure 59 (Part 1 of 3). Auto UNIX System Sample CICSHELL C Program

```
/*                                                                   */
/*03*   INPUT: NO INPUT                                             */
/*                                                                   */
/*                                                                   */
/*05*   OUTPUT:                                                     */
/*                                                                   */
/*      THE STRING "HELLO WORLD !" WILL BE DISPLAYED, AFTER IT       */
/*      HAS BEEN RECEIVED FROM THE CORRESPONDING SERVER PROGRAM      */
/*      AND SEND BACK "HELLO AUTO UNIX !".                           */
/*                                                                   */
/*06    FAILING APIS PRINT RETURNCODE TO VSE CONSOLE.                */
/*07    FROM A PC IT CAN BE SEND TO VSE READER QUEUE:                */
/*      E.G.  SEND CICSHELL A: (FILE=RDR NOUC                        */
/*      MENTION THAT A: IS A PC EMULATOR SESSION AND NOUC MUST       */
/*      BE SPECIFIED FOR NO UPPERCASE TRANSLATION.                   */
/*                                                                   */
/*08    MENTION THAT FOLLOWING MESSAGE CAUSES THE                    */
/*                                                                   */
/*      WARNING EDC4015: UNRESOLVED REFERENCES ARE DETECTED:         */
/*                       @@TRT                                       */
/*      AND THE RESULTING                                            */
/*      1S55I  LAST RETURN CODE WAS 0004                             */
/*      CAN BE IGNORED.                                              */
/*                                                                   */
/*      MENTION THAT THE LAST RETURN CODE IN THE SECOND LINE         */
/*      BEFORE THE LAST ONE (OF THE OUTPUT) MUST BE 2                */
/*      (CAUSED BY ONLY WEAK EXTERNALS).                             */
/*                                                                   */
/*********************************************************************/
#include <string.h>
#include <stdlib.h>                 /* STANDARD LIBRARY              */
main(void)
{
 char achBuffer[100] ;
 short length ;
 signed long int rc = 0;
/* INITIALIZE VARIABLES */
memset(achBuffer,'\0',sizeof(achBuffer));
/* GET ADDRESS OF EIB; NOT USED IN THIS PROGRAM  */
/* BUT JUST ISSUED TO DEMONSTRATE HOW TO GET TO  */
/* DFHEIPTR THAT IS THE POINTER TO EIB           */
/* DFHEIPTR IS DEFINED BY CICS PREPROCESSOR.     */
EXEC CICS ADDRESS EIB(dfheiptr);
length = sizeof(achBuffer);
EXEC CICS RECEIVE
       INTO(achBuffer) LENGTH(length) RESP(rc);
if (rc != DFHRESP(NORMAL) && rc != DFHRESP(EOC)
{
    EXEC CICS WRITE OPERATOR TEXT("RECEIVE failed  ");
    if (rc == DFHRESP(INVREQ))
```

Figure 59 (Part 2 of 3). Auto UNIX System Sample CICSHELL C Program

```
     { EXEC CICS WRITE OPERATOR TEXT("RESP = INVREQ");
     }
     if (rc == DFHRESP(LENGERR))
     { EXEC CICS WRITE OPERATOR TEXT("RESP = LENGERR");
     }
     if (rc == DFHRESP(NOTALLOC))
     { EXEC CICS WRITE OPERATOR TEXT("RESP = NOTALLOC");
     }
     if (rc == DFHRESP(SIGNAL))
     { EXEC CICS WRITE OPERATOR TEXT("RESP = SIGNAL");
     }
      if (rc == DFHRESP(TERMERR))
      { EXEC CICS WRITE OPERATOR TEXT("RESP = TERMERR");
      }
      EXEC CICS RETURN;
  } /* ENDIF */
  /* WRITE RECEIVED STRING TO VSE CONSOLE */
  /* length = strlen(achBuffer);     */
  EXEC CICS WRITE OPERATOR TEXT(achBuffer) TEXTLENGTH(length);
  /* WRITE STRING TO SEND TO VSE CONSOLE */
  strcpy(achBuffer,"Hello Auto UNIX ! ");
  length = strlen(achBuffer);
  EXEC CICS SEND FROM(achBuffer) LENGTH(length) RESP(rc);
  if (rc != DFHRESP(NORMAL))
  { EXEC CICS WRITE OPERATOR
    TEXT("CICS COULD NOT SEND ANSWER TO OPEN EDITION ");
    EXEC CICS RETURN;
  }
  /* WRITE TO CONSOLE WAS HAS BEEN SENT BACK TO PARTNER */
  strcat(achBuffer,"HAS BEEN SENT BACK TO PARTNER.") ;
  length = strlen(achBuffer);
  EXEC CICS WRITE OPERATOR TEXT(achBuffer) TEXTLENGTH(length);
  /* MUST BE REMOVED WHEN RECEIVE WILL BE ACTIVATED [ */
  EXEC CICS RETURN;
 }/* END MAIN */
/*
/. ENDJ2
// EXEC IESINSRT
/*
// IF CATALOG NE 1 OR $MRC GT 4 THEN
// GOTO NOLNK
// EXEC EDCPRLK,SIZE=EDCPRLK,PARM='NATLANG(ENU)/UPCASE'
/*
// EXEC LNKEDT,SIZE=256K
/. NOLNK
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 59 (Part 3 of 3). Auto UNIX System Sample CICSHELL C Program

```
* $$ JOB JNM=AUXCSD,CLASS=O,DISP=D
// JOB AUXCSD  DEFINE RESOURCES TO CICS FOR CPIC SAMPLE
* UPDATE CICS CSD WITH DEFINITIONS FOR THE HELLO WORLD SAMPLE
* (TRANSACTION AUXX)
// EXEC DFHCSDUP
*
DELETE GROUP(AUXHELLO)
*
* DEFINE PROGRAM AUXCOB AS COBOL PROGRAM TO CICS                    X
*
DEFINE PROGRAM(AUXCOB) GROUP(AUXHELLO)
       LANGUAGE(COBOL) RSL(PUBLIC)
*
* DEFINE AUXCOB TRANSACTION (PARTNER TP NAME OF AUTO UNIX )         X
*
DEFINE TRANSACTION(AUXX) GROUP(AUXHELLO) PROGRAM(AUXCOB)
*
* ADD AUXHELLO GROUP TO THE STANDARD VSE/ESA LIST.                  X
*
ADD GROUP(AUXHELLO) LIST(VSELIST)
*   DEFINE THE TCPIP GROUP TO VSE/ESA
MIGRATE TABLE(DFHPPTIP) TOGROUP(TEMP)
COPY GROUP(TEMP) TO(TCPIP) REPLACE
DELETE ALL GROUP(TEMP)
MIGRATE TABLE(DFHPCTIP) TOGROUP(TEMP)
COPY GROUP(TEMP) TO(TCPIP) REPLACE
DELETE ALL GROUP(TEMP)
ADD GROUP(TCPIP) LIST(VSELIST)
/*
/&
* $$ EOJ
```

Figure 60. Define COBOL Sample Program to CSD File

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. AUXCOB.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*****************************************************************
*                                                               *
*       MISCELLANEOUS DEFINITIONS                               *
*                                                               *
*****************************************************************
*
 01  IO-BUFFER PICTURE X(100).
 01  RECLEN PICTURE 9999 COMP VALUE 100.
 01  RC     PICTURE 9999 COMP.
 01  ERR    PICTURE X(80).
 01  OK-MSG.
     05 OK-ONE PIC X(25).
     05 OK-TWO PIC X(55).
*
*
*
*****************************************************************
*                                                               *
*    START OF EXECUTION                                         *
*                                                               *
*****************************************************************
*
*
*
 PROCEDURE DIVISION.
       EXEC CICS RECEIVE
       INTO(IO-BUFFER) LENGTH(RECLEN) RESP(RC) END-EXEC.
       IF RC = DFHRESP(NORMAL) OR RC = DFHRESP(EOC)
            THEN GO TO REC-OK.
       MOVE 'RECEIVE FAILED  ' TO ERR.
       EXEC CICS WRITE OPERATOR TEXT(ERR)
                 END-EXEC.
       IF RC = DFHRESP(INVREQ)
            THEN
            MOVE 'RESP = INVREQ' TO ERR.
            EXEC CICS WRITE OPERATOR TEXT(ERR)
                      END-EXEC.
       IF RC = DFHRESP(LENGERR)
            THEN
            MOVE 'RESP = LENGERR' TO ERR.
            EXEC CICS WRITE OPERATOR TEXT(ERR)
                      END-EXEC.
       IF RC = DFHRESP(NOTALLOC)
            THEN
            MOVE 'RESP = NOTALLOC' TO ERR.
```

Figure 61 (Part 1 of 2). Sample in CICS COBOL Program

```
               EXEC CICS WRITE OPERATOR TEXT(ERR)
                        END-EXEC.
          IF RC = DFHRESP(SIGNAL)
               THEN
               MOVE 'RESP = SIGNAL' TO ERR.
               EXEC CICS WRITE OPERATOR TEXT(ERR)
                        END-EXEC.
          IF RC = DFHRESP(TERMERR)
               THEN
               MOVE 'RESP = TERMERR' TO ERR.
               EXEC CICS WRITE OPERATOR TEXT(ERR)
                        END-EXEC.
               EXEC CICS RETURN END-EXEC.
       REC-OK.
      * WRITE RECEIVED STRING TO VSE CONSOLE
              EXEC CICS WRITE OPERATOR TEXT(IO-BUFFER)
                    TEXTLENGTH(80)
                    END-EXEC.
      * WRITE STRING TO AUTO UNIX USER
           MOVE 18 TO RECLEN.
           MOVE 'HELLO AUTO UNIX | ' TO IO-BUFFER.
           EXEC CICS SEND FROM(IO-BUFFER) LENGTH(RECLEN) RESP(RC)
                    END-EXEC.
             IF RC = DFHRESP(NORMAL)
                THEN GO TO SEND-OK.
                 MOVE 'CICS COULDN''T SEND ANSWER TO OPEN EDITION ' TO
                             IO-BUFFER.
              EXEC CICS WRITE OPERATOR
                  TEXT(IO-BUFFER)
                  TEXTLENGTH(80) END-EXEC.
                  EXEC CICS RETURN
                         END-EXEC.
       SEND-OK.
      * WRITE TO CONSOLE WAS HAS BEEN SENT BACK TO PARTNER
           MOVE IO-BUFFER TO OK-ONE.
           MOVE 'HAS BEEN SENT BACK TO PARTNER.' TO OK-TWO.
            EXEC CICS WRITE OPERATOR TEXT(OK-MSG) TEXTLENGTH(80)
                      END-EXEC.
            EXEC CICS RETURN END-EXEC.
```

Figure 61 (Part 2 of 2). Sample in CICS COBOL Program

```
* $$ JOB JNM=IPINIT00,CLASS=0
* $$ LST CLASS=A
// JOB IPINIT01
// EXEC LIBR
ACC S=PRD2.UNIX
CATALOG IPINIT01.L                          REPLACE=YES
*-----------------------------------------*
*                                         *
*        Define the constants             *
*                                         *
*                                         *
*    The IPADDR and the subsystem MASK    *
*    are dependent on your local          *
*    TCP/IP network configuration.        *
*                                         *
*                                         *
*-----------------------------------------*
SET IPADDR   = 9.164.155.61
SET MASK     = 255.255.255.000
*
SET ALL_BOUND        = 30000
SET WINDOW           = 4096
SET TRANSFER_BUFFERS = 20
SET TELNETD_BUFFERS  = 20
SET RETRANSMIT       = 100
SET DISPATCH_TIME    = 30
SET REDISPATCH       = 10
*
SET GATEWAY          = OFF
*-----------------------------------------*
*                                         *
*        Wait for VTAM Startup            *
*                                         *
*-----------------------------------------*
WAIT     VTAM
*                                         *
*-----------------------------------------*
*                                         *
*        DEFINE TRANSLATION FOR ASCII     *
*                                         *
*-----------------------------------------*
*                                         *
DEFINE TRANSLATION,TYPE=SINGLE,MEMBER=IPXLATE,DEFAULT=DEFAULT
*
*                                         *
*-----------------------------------------*
*                                         *
*        DEFINE NAME OF REMOTE AUTOUNIX   *
*                                         *
*-----------------------------------------*
*                                         *
DEFINE NAME,NAME=AUTOUNIX,IPADDR=9.164.155.62
*
```

Figure 62 (Part 1 of 3). VSE/ESA TCP/IP Configuration

```
*-----------------------------------------*
*                                         *
*        Define the Communication Links   *
*                                         *
*-----------------------------------------*
*                                         *
*        DEFINE LINK TO VM TCPIP          *
*                                         *
*-----------------------------------------*
*
DEFINE LINK,ID=VM_TCPIP,TYPE=CTCA,DEV=630,MTU=1500
*
*-------------------------------------------------*
*                                                 *
*        DEFINE ROUTE TO VM TCPIP                 *
*                                                 *
*    The VM TCPIP subsystem is being defined as   *
*    a gateway.  All IPADDR resolution will be    *
*    handled by VM TCPIP.                         *
*                                                 *
*-------------------------------------------------*
*
*
DEFINE ROUTE,ID=VMESA,LINKID=VM_TCPIP,IPADDR=0.0.0.0,         -
         GATEWAY=9.164.186.250
*
*-----------------------------------------*
*                                         *
*        Define Telnet Daemons            *
*                                         *
*-----------------------------------------*
DEFINE TELNETD,ID=LU,TERMNAME=TELNLU,TARGET=DBDCCICS,PORT=23,COUNT=2
DEFINE TELNETD,ID=LU21,TERMNAME=TELNLU21,TARGET=DBDCCICS,PORT=23
DEFINE TELNETD,ID=LU22,TERMNAME=TELNLU22,TARGET=DBDCCICS,PORT=23
*-----------------------------------------*
*                                         *
*        Define FTP Daemons               *
*                                         *
*-----------------------------------------*
DEFINE FTPD,ID=FTP,PORT=21,COUNT=2
DEFINE FTPD,ID=FTP11,PORT=21
DEFINE FTPD,ID=FTP12,PORT=21
*-----------------------------------------*
*                                         *
*        Line Printer Daemons             *
*                                         *
*-----------------------------------------*
DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'
DEFINE LPD,PRINTER=FASTLIB,QUEUE='PRD2.SAVE'
DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=PRD2,SUBLIB=SAVE
```

Figure 62 (Part 2 of 3). VSE/ESA TCP/IP Configuration

```
*-----------------------------------------*
*                                         *
*        Automated Line Printer Client    *
*                                         *
*-----------------------------------------*
DEFINE EVENT,ID=LST_LISTEN,TYPE=POWER,CLASS=X,QUEUE=LST,ACTION=LPR
*-----------------------------------------*
*                                         *
*        Setup the File System            *
*                                         *
*-----------------------------------------*
DEFINE FILESYS,LOCATION=SYSTEM,TYPE=PERM
*
*  Define sample file for auto unix
*
DEFINE FILE,PUBLIC='VSESPUC',DLBL=VSESPUC,TYPE=VSAMCAT
DEFINE FILE,PUBLIC='AUXFILK',DLBL=AUXFILK,TYPE=KSDS
DEFINE FILE,PUBLIC='AUXFILE',DLBL=AUXFILE,TYPE=ESDS
*
DEFINE FILE,PUBLIC='IJSYSRS',DLBL=IJSYSRS,TYPE=LIBRARY
DEFINE FILE,PUBLIC='PRD1',DLBL=PRD1,TYPE=LIBRARY
DEFINE FILE,PUBLIC='PRD2',DLBL=PRD2,TYPE=LIBRARY
DEFINE FILE,PUBLIC='POWER',DLBL=IJQFILE,TYPE=POWER
*
MODIFY FILE,PUBLIC='VSE.SYSRES.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.PRD1.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.PRD2.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.DUMP.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.PRIMARY.LIBRARY',TYPE=LIBRARY
*
MODIFY FILE,PUBLIC='ICCF.LIBRARY',TYPE=ICCF
MODIFY FILE,PUBLIC='VSE.POWER.QUEUE.FILE',TYPE=POWER
*
*-----------------------------------------*
*                                         *
*        Define NFS Daemon                *
*        The NFS Daemon should be defined *
*        after the file system.           *
*                                         *
*-----------------------------------------*
DEFINE NFSD,ID=NFSD
*-----------------------------------------*
*                                         *
*        Setup member NETWORK.L to        *
*        execute once the engine has      *
*        been activated                   *
*                                         *
*-----------------------------------------*
INCLUDE NETWORK,DELAY
/+
/*
/&
* $$ EOJ
```

Figure 62 (Part 3 of 3). VSE/ESA TCP/IP Configuration

```
* $$ JOB JNM=TCPSTART,CLASS=7,DISP=L
* $$ LST CLASS=A,DISP=D
// JOB TCPSTART START UP TCPIP
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM,CAT=VSESPUC
// LIBDEF *,SEARCH=(PRD2.UNIX,PRD2.CONFIG,PRD1.BASE,PRD2.SCEEBASE)
// EXEC PROC=DTRICCF
// SETPFIX LIMIT=400K
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT01',DSPACE=2M
/&
* $$ EOJ
```

Figure 63. VSE/ESA TPC/IP Startup JCL

```
* $$ JOB JNM=SKAPPN,CLASS=O,DISP=D
// JOB SKAPPN      DEFINE FILES FOR VTAM APPN
// DLBL IJSYSUC,'VSESP.USER.CATALOG',O,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DELETE (VTAM.DSDBCTL) PURGE CL              -
          CATALOG(VSESP.USER.CATALOG)
   DELETE (VTAM.DSDB1) PURGE CL                -
          CATALOG(VSESP.USER.CATALOG)
   DELETE (VTAM.DSDB2) PURGE CL                -
          CATALOG(VSESP.USER.CATALOG)
   DELETE (VTAM.TRSDB) PURGE CL                -
          CATALOG(VSESP.USER.CATALOG)
   DEFINE CLUSTER (NAME(VTAM.DSDBCTL)          -
          RECORDS (1 1)                        -
          FREESPACE (20 10)                    -
          NONINDEXED                           -
          RECORDSIZE (20,20)                   -
          REUSE                                -
          SHAREOPTIONS (2,3)                   -
          VOLUMES (SYSWK1))                    -
          CATALOG (VSESP.USER.CATALOG)
   DEFINE CLUSTER (NAME(VTAM.DSDB1)            -
          RECORDS (1 1)                        -
          FREESPACE (20 10)                    -
          NONINDEXED                           -
          RECORDSIZE (1000,1000)               -
          REUSE                                -
          SHAREOPTIONS (2,3)                   -
          VOLUMES (SYSWK1))                    -
          CATALOG (VSESP.USER.CATALOG)
   DEFINE CLUSTER (NAME(VTAM.DSDB2)            -
          MODEL (VTAM.DSDB1))                  -
          CATALOG (VSESP.USER.CATALOG)
   DEFINE CLUSTER (NAME(VTAM.TRSDB)            -
          MODEL (VTAM.DSDB1))                  -
          CATALOG (VSESP.USER.CATALOG)
   SET MAXCC = 0
/*
// OPTION STDLABEL=DELETE
DSDBCTL
DSDB1
DSDB2
TRSDB
/*
// OPTION STDLABEL=ADD
// DLBL DSDBCTL,'VTAM.DSDBCTL',,VSAM,CAT=VSESPUC
// DLBL DSDB1,'VTAM.DSDB1',,VSAM,CAT=VSESPUC
// DLBL DSDB2,'VTAM.DSDB2',,VSAM,CAT=VSESPUC
// DLBL TRSDB,'VTAM.TRSDB',,VSAM,CAT=VSESPUC
/*
// EXEC IESVCLUP,SIZE=AUTO             ADD LABEL TO STDLABUP PROC
D                                              DSDBCTL
D                                              DSDB1
D                                              DSDB2
D                                              TRSDB
```

Figure 64 (Part 1 of 2). Definition of VTAM Clusters for VSE/ESA APPN

```
A VTAM.DSDBCTL                                          DSDBCTL VSESPUC
A VTAM.DSDB1                                            DSDB1   VSESPUC
A VTAM.DSDB2                                            DSDB2   VSESPUC
A VTAM.TRSDB                                            TRSDB   VSESPUC
/*
// EXEC LIBR,PARM='MSHP'
CONNECT S=PRD1.BASE:PRD2.CONFIG
COPY COSAPPN.Z:COSAPPN.B R=Y
/*
/&
* $$ EOJ
```

Figure 64 (Part 2 of 2). Definition of VTAM Clusters for VSE/ESA APPN

```
* $$ JOB JNM=TCPAPPL,CLASS=O
* $$ LST CLASS=A
// JOB TCPAPPL
// EXEC LIBR,PARM='MSHP'
ACC S=PRD2.UNIX
CATALOG TCPAPPL.B                              REPLACE=YES
TCPAPPL  VBUILD TYPE=APPL
TCPDIAG  APPL  AUTH=(ACQ)
TCPIP    APPL  AUTH=(ACQ)
TELNLU01 APPL  AUTH=(ACQ)
TELNLU02 APPL  AUTH=(ACQ)
TELNLU03 APPL  AUTH=(ACQ)
TELNLU04 APPL  AUTH=(ACQ)
TELNLU05 APPL  AUTH=(ACQ)
TELNLU06 APPL  AUTH=(ACQ)
TELNLU07 APPL  AUTH=(ACQ)
TELNLU08 APPL  AUTH=(ACQ)
TELNLU09 APPL  AUTH=(ACQ)
TELNLU10 APPL  AUTH=(ACQ)
TELNLU11 APPL  AUTH=(ACQ)
TELNLU12 APPL  AUTH=(ACQ)
TELNLU13 APPL  AUTH=(ACQ)
TELNLU14 APPL  AUTH=(ACQ)
TELNLU15 APPL  AUTH=(ACQ)
TELNLU16 APPL  AUTH=(ACQ)
TELNLU21 APPL  AUTH=(ACQ)
TELNLU22 APPL  AUTH=(ACQ)
/+
/*
/&
* $$ EOJ
```

Figure 65. Changed Sample TCPAPPL.B Member

```
* $$ JOB JNM=ATCSTR00,CLASS=O
* $$ LST CLASS=A
// JOB ATCSTR00 CATALOG VTAM STARTUP BOOK
// EXEC LIBR
ACC S=PRD2.UNIX
CATALOG ATCSTR00.B        EOD=XY          REPLACE=YES
SSCPID=1,                                                          X
SSCPNAME=SSCP01,                                                   X
NETID=VTAM1,                                                       X
NODETYPE=NN,              DEFINE APPN NETWORK NODE, MANDATORY      X
HOSTSA=1,                                                          X
HOSTPU=NODE01,                                                     X
APPNCOS=NONE,             APPN                                     X
CONNTYPE=APPN,            APPN                                     X
INITDB=ALL,               APPN                                     X
SORDER=APPN,              APPN                                     X
MAXSUBA=255,                                                       X
CONFIG=00,                                                         X
NOPROMPT,                                                          X
IOINT=0,                                                           X
SGALIMIT=0,                                                        X
BSBUF=(28,,,,1),                                                   X
CRPLBUF=(60,,,,1),                                                 X
LFBUF=(70,,,,11),                                                  X
IOBUF=(70,288,,,11),                                               X
LPBUF=(12,,,,6),                                                   X
SFBUF=(20,,,,20),                                                  X
SPBUF=(210,,,,32),                                                 X
XDBUF=(6,,,,1)
XY
/*
/&
* $$ EOJ
```

Figure 66. Updated VSE/ESA VTAM Startup List

```
* $$ JOB JNM=EZMVSV03,CLASS=O,DISP=D
// JOB EZMVSV03  CATALOG VTAM/APPN  DEFS INTO PRD2.UNIX       (VTMAHHC
* ********************************************************************
*
* CATALOGS VTAM DEFINITIONS FOR APPN CONNECTION INTO
* 'PRD2.UNIX'
*
* MAKE YOUR ADAPTATIONS WHERE NECESSARY
*
* ********************************************************************
// EXEC LIBR,PARM='MSHP'
ACC S=PRD2.UNIX
CATALOG VTMAHHC.B                          REPLACE=YES
AHHC1    VBUILD TYPE=LOCAL
CAHHC1   PU TRLE=TRLE1,XID=YES,VPACING=O,CPCP=YES,DYNLU=YES,           X
              CONNTYPE=APPN,ISTATUS=ACTIVE
/+
/*
/&
* $$ EOJ
```

Figure 67. Auto UNIX System Sample EZMVSV03

```
* $$ JOB JNM=EZMVSV04,CLASS=O,DISP=D
// JOB EZMVSV04  CATALOG VTAM/APPN  DEFS INTO PRD2.UNIX       (VTMTRL
* ********************************************************************
*
* CATALOGS VTAM DEFINITIONS FOR APPN CONNECTION INTO
* 'PRD2.UNIX'
*
* MAKE YOUR ADAPTATIONS WHERE NECESSARY
* - CHANGE THE CTC ADDRESSES OF 'READ=(640),WRITE=(641)'
*   TO THE ADDRESSES MATCHING YOUR ENVIRONMENT
*
* ********************************************************************
// EXEC LIBR,PARM='MSHP'
ACC S=PRD2.UNIX
CATALOG VTMTRL.B                           REPLACE=YES
TRL1     VBUILD TYPE=TRL
TRLE1    TRLE  LNCTL=MPC,MAXBFRU=10,                                   X
              READ=(640),WRITE=(641)
/+
/*
/&
* $$ EOJ
```

Figure 68. Auto UNIX System Sample EZMVMV04

```
* $$ JOB JNM=ATCCON00,CLASS=0
* $$ LST CLASS=A
// JOB ATCCON00 CATALOG VTAM CONFIGURATION DEFINITIONS
// EXEC LIBR
ACC S=PRD2.UNIX
CATALOG ATCCON00.B        EOD=XY          REPLACE=YES
         TCPAPPL,             TCP APPL DEFINITIONS                    X
         VTMAPPL,                                                     X
         VTMTRL,              AUTO UNIX TRL MAJOR NODE                X
         VTMAHHC,             AUTO UNIX APPN HOST TO HOST CONNECTION  X
         VTMSNA,              >>>> TRL SHOULD PRECEDE VTMSNA          X
         VTMNSNA,                                                     X
         VTMCTCA,                                                     X
         VTMPATH,                                                     X
         VTMCA1,                                                      X
         VTMCA2,                                                      X
         VTMCA3,                                                      X
         VTMCDRM,                                                     X
         VTMCDRS,                                                     X
         VTMSW1
XY
/*
/&
* $$ EOJ
```

Figure 69. Updated VSE/ESA VTAM Configuration List

Appendix C. Sample MOUNTPW C Program

The following program (MOUNTPW C) is supplied with 5654-030 for use on NFS client systems. The MOUNTPW C file must be copied to the client system and compiled into an executable program before you can execute the MOUNTPW command. Compile the MOUNTPW program using a C compiler resident on the client system.

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME = MOUNTPW C                                           */
/*                                                                   */
/* COPYRIGHT =                                                       */
/*  Licensed Materials - Property of IBM                             */
/*  This product contains "Restricted Materials of IBM"              */
/*  5654-030 (C) Copyright IBM Corp. - 1990, 1998                    */
/*  All rights reserved.                                             */
/*  US Government Users Restricted Rights -                          */
/*  Use, duplication or disclosure restricted by GSA ADP             */
/*  Schedule                                                         */
/*  Contract with IBM Corp.                                          */
/*  See IBM Copyright Instructions.                                  */
/*                                                                   */
/* STATUS = TCP/IP level 310 for VM/ESA Version 2 Release 3          */
/*                                                                   */
/*********************************************************************/
/*
   This program is supplied with 5654-030 for use on NFS client systems.
   It is possible that modifications will be necessary to adapt this code
   to your NFS client systems.  These modifications are the responsibility
   of the customer.  IBM does not claim this code will be suitable for
   all systems, or that this function can be realized in all NFS client
   environments.  Information about experiences with other NFS client
   environments may be available from IBM Software Support, which should
   be contacted if difficulty is experienced when trying to port this
   program to a local client system.

   IBM grants permission to use and copy this program, to modify it
   for other environments or purposes, without restriction.

*/

#include <stdio.h>
#include <netdb.h>

   /* AIX RISC 6000 uses a slightly different name for the rpc/types header
file. */
#define AIX6000
#if defined(AIX6000)
#include <rpc/types.h>
#else
#include <rpc/rpctypes.h>
#endif

#include <rpc/xdr.h>
```

```
/* Program to supply a password to a VM NFS server prior to
   a mount request.

   The password can be specified as part of the mount argument
   string, but that makes it visible to anyone who can issue
   the query form of the mount command (usually non-privileged),
   or ask the local mount service what has been mounted.

   This program invokes a non-standard mount program procedure,
   defined by the VM NFS implementation.  This procedure remembers
   the target mount information and access authentication
   information for a short while (generally five minutes, but this
   may be varied by a source change incorporated by an individual
   VM installation), and combines it with data in a subsequent,
   ordinary mount request from that client for a mount of the
   same minidisk, SFS or BFS directory. There may be multiple
   mountpw calls to the VM server for different objects, and
   multiple mountpw calls for the same object from different
   clients, before matching mount requests are received.
   If a second mountpw call from the same client is received for an
   object for which a mountpw call has already been received, the
   earlier data is replaced by the new data.

   There is one argument to this program, specifying the host,
   minidisk or directory, and password information. Its syntax
   is similar to that used on a mount command addressed to a
   VM NFS server:

        host:user.vaddr,password=pppp,userid=uuuu,account=aaaa
        host:directory,password=pppp,userid=uuuu,account=aaaa

   where:

     "host" identifies the VM server in a format acceptable to
     callrpc (either symbolic or numeric string).

     "user" and "vaddr" identify the target CMS minidisk.

     "directory" identifies the fully-qualified target SFS or
     BFS directory name.

     "password" is a keyword, and may be shortened to any initial
     abbreviation (minimum is "p").

     "pppp" is the relevant CP link password (if "userid" is not
     specified), or the relevant logon password (if "userid" is
     specified).

     "userid" is a keyword, and may be shortened to any initial
     abbreviation (minimum is "u").

     "uuuu" is a userid known by the VM system identified by "host"
     for which the logon password is "pppp".  For minidisks, this
     information is when an access list scheme (such as RACF --
     -- Resource Access Control Facility) is used by the VM system
     for controlling access. For SFS and BFS directories, this
     information is used for controlling access.
```

```
         "account" is a keyword, and may be shortened to any initial
         abbreviation (minimum is "a").

         "aaaa" is account information (up to 64 bytes), which is not
         used directly by the VM NFS server, but which is passed to
         the nfslink program that performs authentication of a client
         request to mount a CMS minidisk.  This parameter is intended
         to provide a mechanism by which an individual installation may
         acquire additional information from a client in order to decide
         whether to grant or deny access to a minidisk.


      The identification of the object to be mounted (CMS minidisk or
      SFS or BFS directory) is required. All other parts are optional.
      The information required to authenticate a mount request may
      differ from system to system according to the choices made
      during installation of VM NFS.  Any information not supplied
      by a mountpw call may be supplied in the mount call argument string.
      If different information is supplied in the mountpw and mount calls,
      the information in the mount call is used.

      In addition to the fields described above, any of the other fields
      allowed in the VM/CMS mount syntax are allowed:  they will be parsed
      to verify correct syntax, but their value will be ignored.  This is
      to allow the user the convenience of using similar arguments for both
      mountpw and mount commands.

      The parsing of the mount (and mountpw) argument string is performed by
      the parse_mnt function, which may be found in the file reqstart.c.  */


int main(argc,argv)          /* Value is 0 if successful; non-zero if error. */
int                argc;
char              *argv[];

{char          hostname[65];
 register char       *p = hostname, *a = argv[1];
 char                 c, *a_pointer;
 unsigned int         k, rc;

 if (argc < 2) {                 /* Argument missing. */
   fprintf(stderr, "Formats are:\n   %s
host:directory,password=pppp,userid=uuuu\n",
       argv[0]);
   fprintf(stderr, "   %s
host:user.vaddr,password=pppp,userid=uuuu,account=aaaa\n",
       argv[0]);
   fprintf(stderr, "\n\"directory\" identifies an SFS or BFS directory.\
\nFor SFS and BFS directories, use both login password and userid name.\
\n\n\"user.vaddr\" identifies a CMS minidisk.\
\nFor minidisk, use only password for standard VM systems (pppp=read or mult\
\nlink password). Use both login password and userid name for RACF VM systems.\
\n\nUse account if required by local installation.\
\n\nAny initial substring of keyword names is accepted.\n");
   fprintf(stderr, "\nAll authentication data may be part of a mount command.\n\
%s is used to avoid display of sensitive data by mount query.\n\
Data supplied by %s is kept by the VM NFS server only until a matching\n\
mount request is received from this client system, or for five minutes,\n\
whichever occurs first.\n\n", argv[0], argv[0]);
```

```
  return 1;}

for (k=0; k<sizeof(hostname); k++) {
  if ((*p++ = c = *a++) == ':') break;
  if (c == '\000') return 2;}     /* Invalid syntax: colon missing. */

*(--p) = '\0';                  /* Store proper string delimiter. */
a_pointer = a;
if (rc = callrpc(hostname, 100005, 1, 6, xdr_wrapstring,
                 &a_pointer, xdr_u_long, &k)) {
  fprintf(stderr, "Error in %s:  %d returned by callrpc\n", argv[0], rc);
  clnt_perrno(rc);
  return 3;}                    /* RPC error. */

if (k) {
  fprintf(stderr, "Error in %s:  %d returned from mount server at %s\n",
      argv[0], k, hostname);
  return 4;}                    /* Error from remote host. */
return 0;}
```

Appendix D. Samples NFS for VSE/ESA

```
# ------------------------------------------------------------------#
#                       N F S T Y P E S . L                         #
#                                                                   #
# The following is a list of file types that will allow data being #
# written *TO* the VSE operating system to be accessible to the    #
# mainframe-based programs. Any file type not listed will still be #
# usable by the NFS client user, but VSE-based programs will not   #
# be able to use the data if the created file does not have an     #
# extension that is in this list.                                  #
# ------------------------------------------------------------------#
#
# The BINARY entries will not be converted to EBCDIC, nor will
# we worry about eliminating the CR/LF. The difference between BINARY
# and BIN80 has to do with the method that is used to store the data
# in a VSE library. BINARY will use "STRING" mode, while BIN80 will
# write in 80-byte records instead of the continuous byte-string that
# is used by BINARY.
#
BIN      BINARY     <---- Store it in string mode
BJB      BIN80
DUMP     BINARY     <---- Stored in string mode
E        BIN80      <---- Edited macros (Non-HLASM) end in ".E"
JOB      BIN80
OBJ      BIN80
W        BIN80      <---- PRD2.PWS binary entries end in ".W"
#
# The ASCII entries will be converted to EBCDIC and will have their
# CR/LF sequences removed.
#
&BLANK   ASCII
A        ASCII
ASM      ASCII
B        ASCII
C        ASCII
CATALOG  ASCII
D        ASCII
F        ASCII
FILE     ASCII
G        ASCII
H        ASCII
HTM      ASCII
HTML     ASCII
I        ASCII
J        ASCII
JCL      ASCII
K        ASCII
L        ASCII
LST      ASCII
```

Figure 70 (Part 1 of 2). Sample NFS for VSE File Type Translation File NFSTYPES.L

```
M         ASCII
N         ASCII
O         ASCII
OUT       ASCII
OUTPUT    ASCII
P         ASCII
PROC      ASCII
PUN       ASCII
PRT       ASCII
Q         ASCII
R         ASCII
RDR       ASCII
S         ASCII
SLI       ASCII
T         ASCII
TEXT      ASCII
TXT       ASCII
U         ASCII
V         ASCII
VRML      ASCII
X         ASCII
Y         ASCII
```

Figure 70 (Part 2 of 2). Sample NFS for VSE File Type Translation File NFSTYPES.L

```
*
* Sample NFS for VSE configuration file NFSCFG.L
*
*
* The following are options that the typical site may use for tuning
*
DirCacheSize=80k      /* Here goes 40 to 84 byte entries (RECFM=V) */
DirGroupSize=10       /* The number of entries to send during I/O  */
DirlistExit=No        /* DIRLIST exit name, or YES for "NFSEX01"   */
DirlistNullFiles=Yes  /* Yes=Include zero-length files in DirList  */
LimitVSAMToESDS=Yes   /* Yes=Display only ESDS dataset via VSAM    */
LoadPowerClasses=No   /* Yes=Define all classes at startup         */
LoadPowerQueues=No    /* Yes=Define all queue names at startup     */
MaxAllocErrs=3        /* Maximum consecutive ALLOC FRBLOK errors   */
MaxRequests=1000      /* Maximum requests that can be serviced     */
OS2Support=Yes        /* Perform additional routines for OS/2      */
PrintIDCAMS=Yes       /* Yes=Output IDCAMS results to SYSLST       */
ReadCacheSize=128k    /* The total cache size for any READ         */
ReadCacheTime=30      /* Max seconds a recently-read entry remains */
Security=No           /* Yes=Use the same user exit that FTP does  */
ShowActive=No         /* Yes=Show Power queue entries with disp=*  */
ShowNonExported=No    /* Yes=List truncated UDP EXPORT lists       */
ShowPhases=No         /* No=Suppress .PHASE info in DIRLIST        */
ShowRPCError=No       /* Yes=Show info when GETPORT fails via RPC  */
ShowStaleInfo=No      /* Yes=Show info when request goes stale     */
TimeOffset=+0         /* Number of minutes to use for accurate TOD */
Truncate=Yes          /* No=Don't truncate WRITE records to VSE    */
UnixTextMode=Off      /* ON=Omit CR character from end-of-record   */
VSAMTableSize=64k     /* Maximum size for the VSAM attribute table */
WakeUpTime=5          /* # of seconds to check cache usage         */
WriteCacheTime=30     /* Max seconds a written entry remains       */
XDRExit=No            /* Indicate XDR support and possible name    */
*
* The following are options are for internal use and should not be
* used unless requested to do so by Technical Support.
*
DatagramTrace=NO      /* Yes=produce SYSLST output of I/O traffic  */
Debug=No              /* Yes=Output a lot of debugging info        */
ListExports=No        /* Yes=List all mount points that we know of */
MaxExportSize=4K      /* Maximum outgoing EXPORT datagram (MOUNTD) */
MaxPacketSize=9k      /* Defines the largest supported datagram    */
Version3=Yes          /* To allow version 3 protocol               */
WatchDirCache=No      /* Yes=Show DIR cache real time statistics   */
WatchReadCache=No     /* Yes=Show READ cache real time statistics  */
WatchReads=No         /* Yes=Show READ real time activity          */
WatchWrites=No        /* Yes=Show WRITE real time activity         */
WatchWriteCache=No    /* Yes=Show WRITE cache real time statistics */
```

Figure 71. Sample NFS for VSE Configuration File NFSCFG.L

Appendix E. Figures and Samples for Chapter 10

```
* $$ JOB JNM=MQINQ,DISP=D,CLASS=O,NTFY=YES
* $$ LST DISP=D,CLASS=V,PRI=3
* $$ PUN DISP=I,DEST=*,PRI=9,CLASS=O
// JOB MQINQ TRANSLATE PROGRAM MQINQ
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
$ $$ LST DISP=D,CLASS=V,PRI=3
// JOB MQINQ2 COMPILE PROGRAM MQINQ
// LIBDEF *,SEARCH=(PRD2.SCEECICS,PRD2.SCEEBASE)
// SETPARM CATALOG=1
// IF CATALOG = 1 THEN
// GOTO CAT
// OPTION ERRS,SXREF,SYM,LIST,NODECK,SOURCE,FLAG(I)
// GOTO ENDCAT
/. CAT
// LIBDEF PHASE,CATALOG=PRD2.UNIX
// OPTION ERRS,SXREF,SYM,CATAL,NODECK
   PHASE MQINQ,*
   INCLUDE DFHELII
/. ENDCAT
// EXEC IGYCRCTL,SIZE=IGYCRCTL
 CBL LIB,APOST,NOADV,NODYNAM,RENT,BUF(4096)
* $$ END
// ON $CANCEL OR $ABEND GOTO ENDJ2
// OPTION NOLIST,NODUMP,DECK
// EXEC DFHECP1$,SIZE=512K
 CBL XOPTS(COBOL2 CICS DEBUG)
      * ---------------------------------------------------------------*
       IDENTIFICATION DIVISION.
      * ---------------------------------------------------------------*
       PROGRAM-ID. MQINQ.
      *REMARKS
      *****************************************************************
      * §START_COPYRIGHT§                                             *
      *   Statement:     Licensed Materials - Property of IBM         *
      *                  5647 - A01                                   *
      *                                                               *
      *                  (C) Copyright IBM Corp. 1997,1998            *
      *                                                               *
      *                  Status = HEZ6606                             *
      *                                                               *
      * §END_COPYRIGHT§                                               *
      *                                                               *
      * ************************************************************* *
```

Figure 72 (Part 1 of 7). Sample CICS MQSeries Application Program

```
* ************************************************************ *
*                                                              *
*  Product Number   : 5647-A01                                 *
*                                                              *
*  MODULE NAME      : MQINQ                                    *
*                                                              *
*  TRANSACTION ID   : INQK                                     *
*                                                              *
*  Environment      : VSE/CICS COBOL/VSE                       *
*                                                              *
*  Description      : Sample MQI transaction program           *
*                     which can be started from Open Edition   *
*                     (by sending transaction name to          *
*                     VSECICS (MQI Server Program) which       *
*                     in turn does a                           *
*                     EXEC CICS START TRANSID                  *
*                                                              *
*  Function         : This MQI receives its data from          *
*                     VSECICS MQ Server sample                 *
*                     (which received its input from Auto UNIX) *
*                     via EXEC CICS RETRIEVE INTO call         *
*                     The input is a two byte key to a KSDS    *
*                     file, AUXFILK. It uses the received      *
*                     key to read the specified record and     *
*                     sends back the record contents to        *
*                     Auto UNIX by directly putting it to the  *
*                     OE MQI queue (a remote queue for VSE).   *
*                                                              *
*  Return Values    : 0 - Successful completion                *
*                     in case of error write error message     *
*                     to VSE console                           *
*                                                              *
* ************************************************************ *
*                                                              *
*                      Program logic                           *
*                      -------------                           *
*                                                              *
*   - Connect to VSE Queue Manager                             *
*   - write answerstring to VSE console                        *
*   - MQPUT1 answerstring (including inputstring) to OE queue  *
*     in case of error print message to VSE console            *
*   - disconnect from VSE Queue Manager                        *
*                                                              *
* ************************************************************ *
```

Figure 72 (Part 2 of 7). Sample CICS MQSeries Application Program

```
      * ------------------------------------------------------------- *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      * ------------------------------------------------------------- *
      *
      *    General work fields
      *
      * VSE Queue Manager
       01  QUEUE-MANAGER-NAME   PIC X(4)   VALUE
           'VSE1'.
      * Open Edition Queue to inform about errors
      * OEQUEUE is alias for CSQ1.REMOTEQ
       01  OE-QUEUE-NAME       PIC X(30)  VALUE
           'OEQUEUE'.
      * Variables used for APIs
       01  HCONN                PIC S9(9)  BINARY.
       01  BUFFLEN              PIC S9(9)  BINARY.
       01  COMPCODE             PIC S9(9)  BINARY.
       01  REASON               PIC S9(9)  BINARY.
      * Other Variables
       01  TRANSACT-INPUT       PIC X(02)  VALUE SPACES.
       01  WORK-AREA            PIC X(80)  VALUE SPACES.
      * CICS RESP variables
       01  CICS-RESP            PIC S9(8) COMP.
       01  CICSWTO              PIC X(080) VALUE SPACES.
       01  CICS-TO-UNIX         PIC X(080) VALUE SPACES.
       01  TEMPLENGTH           PIC S9(8) COMP VALUE 0.
      * Error Message structure
       01   ERROR-MESSAGE.
        05  FILLER              PIC  X(14) VALUE 'MQINQ FAILED: '.
        05  ERR-FUNCTION        PIC  X(9).
        05  FILLER              PIC  X(13) VALUE ',  COMPCODE = '.
        05  ERR-DISPLAY-CCODE   PIC  9(4) VALUE ZERO.
        05  FILLER              PIC  X(14) VALUE ',  REASONCODE ='.
        05  ERR-DISPLAY-RCODE        PIC  9(4) VALUE ZERO.
      * Include MQOD (object descriptor)
       01 MQM-OBJECT-DESCRIPTOR.
          COPY CMQODV SUPPRESS.
       01 MQM-PUT-MESSAGE-OPTIONS.
          COPY CMQPMOV SUPPRESS.
       01 MQM-MESSAGE-DESCRIPTOR.
          COPY CMQMDV SUPPRESS.
      * Include MQI constants
       01 MQM-CONSTANTS.
          COPY CMQV SUPPRESS.
      * ------------------------------------------------------------- *
       PROCEDURE DIVISION.
          A-MAIN SECTION.
      *
      * Retrieve INPUTSTRING from MQSeries sample server
      *
          EXEC CICS RETRIEVE INTO (TRANSACT-INPUT)
                           LENGTH(LENGTH OF TRANSACT-INPUT)
                           RESP(CICS-RESP)
          END-EXEC.
```

Figure 72 (Part 3 of 7). Sample CICS MQSeries Application Program

```
      * Check Return Codes of RETRIEVE
           EVALUATE TRUE
           WHEN (CICS-RESP = DFHRESP(NOTAUTH))
           MOVE ' could not receive inputstring because RESP=NOAUTH |'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(ENDDATA))
           MOVE ' could not receive inputstring because RESP=ENDDATA |'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(ENVDEFERR))
           MOVE ' could not receive inputstring because RESP=ENDDEFERR|'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(INVREQ))
           MOVE ' could not receive inputstring because RESP=INVREQ |'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(INVTSREQ))
           MOVE ' could not receive inputstring because RESP=INVSREQ |'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(IOERR))
           MOVE ' could not receive inputstring because RESP=IOERR |'
            TO CICSWTO(5:)
           WHEN (CICS-RESP = DFHRESP(NOTFND))
           MOVE ' could not receive inputstring because RESP=NOTFND |'
            TO CICSWTO(5:)
           END-EVALUATE.
      *
      * In Case of not being able to RETRIEVE the data,
      * cancels with OPERATOR WARNING
      *
           IF (CICSWTO(1:) NOT = SPACES) THEN
                                          GO TO CICS-WTO
                                          GO TO A-MAIN-END
                                       END-IF.
      *
      * Retrieved, so:
      * Connect to VSE queue manager
      *
            CALL 'MQCONN' USING QUEUE-MANAGER-NAME
                                HCONN
                                COMPCODE
                                REASON
            END-CALL

            IF (COMPCODE NOT = MQCC-OK) THEN
             MOVE 'MQCONN' TO ERR-FUNCTION
             GO TO ERROR-DISPLAY
             GO TO A-MAIN-END
            END-IF
```

Figure 72 (Part 4 of 7). Sample CICS MQSeries Application Program

```
      *
      * Put answer string to Open Edition queue
      *
           MOVE QUEUE-MANAGER-NAME TO MQOD-OBJECTQMGRNAME.
           MOVE OE-QUEUE-NAME TO MQOD-OBJECTNAME.
           MOVE LENGTH OF CICS-TO-UNIX TO BUFFLEN.
           MOVE SPACES TO MQMD-REPLYTOQ.
           MOVE SPACES TO MQMD-REPLYTOQMGR.
           MOVE SPACES TO MQMD-MSGID.
           MOVE EIBTRNID TO MQMD-MSGID.
           MOVE MQCI-NONE TO MQMD-CORRELID.
           MOVE MQFMT-NONE TO MQMD-FORMAT.
           COMPUTE MQPMO-OPTIONS = MQPMO-SYNCPOINT.
      *
      * Write record retrieved back to AUTO UNIX Queue
      *
      * BECAUSE MQSERIES FOR VSE DOES NOT SUPPORT ANY SYNCPOINTING
      * and does not provide MQSeries APIs, CICS SYNCPOINTS have to
      * be used to ensure that the message will be transferred from
      * the VSE transmission queue to the Auto UNIX queue.
           EXEC CICS HANDLE CONDITION
                                    NOTFND (NOT-FOUND)
                                    END-EXEC.
           EXEC CICS READ DATASET ('AUXFILK')
                          INTO (CICS-TO-UNIX)
                          RIDFLD (TRANSACT-INPUT)
                          EQUAL
                          END-EXEC.
           MOVE CICS-TO-UNIX TO WORK-AREA
           COMPUTE TEMPLENGTH = (LENGTH OF CICS-TO-UNIX - 4)
           MOVE WORK-AREA TO CICS-TO-UNIX(5:TEMPLENGTH)
           MOVE 0 TO TEMPLENGTH
           MOVE EIBTRNID TO CICS-TO-UNIX(1:4).
       SYNCPOINT.
           EXEC CICS SYNCPOINT
           END-EXEC
      * Put message to Auto UNIX queue
      *  if put fails write error message to VSE console
           CALL 'MQPUT1' USING HCONN
                               MQOD
                               MQMD
                               MQPMO
                               BUFFLEN
                               CICS-TO-UNIX
                               COMPCODE
                               REASON.
           IF (COMPCODE NOT = MQCC-OK) THEN
             MOVE 'MQPUT1' TO ERR-FUNCTION
             GO TO ERROR-DISPLAY
           END-IF.
      *    we define MQPUT1 as a single unit of work
           EXEC CICS SYNCPOINT
           END-EXEC.
```

Figure 72 (Part 5 of 7). Sample CICS MQSeries Application Program

```
      *
       CLEANUP.
           CALL 'MQDISC' USING HCONN
                               COMPCODE
                               REASON.
           IF (COMPCODE NOT = MQCC-OK) THEN
             MOVE 'MQDISC' TO ERR-FUNCTION
             GO TO ERROR-DISPLAY
           END-IF.
      *
      *
       A-MAIN-END.
           EXEC CICS RETURN
           END-EXEC.
      *
      * Error Displaying procedure
      *
       CICS-WTO.
           EXEC CICS WRITE OPERATOR TEXT(CICSWTO)
                     TEXTLENGTH(LENGTH OF CICSWTO)
           END-EXEC.
           GOBACK.
      *
        ERROR-DISPLAY.
           MOVE COMPCODE TO  ERR-DISPLAY-CCODE.
           MOVE REASON TO  ERR-DISPLAY-RCODE.
           EXEC CICS  WRITE OPERATOR
                TEXT(ERROR-MESSAGE)
                TEXTLENGTH (LENGTH OF ERROR-MESSAGE)
           END-EXEC.
           GOBACK.
      *
      * Record not found procedure
      *
       NOT-FOUND.
            MOVE 1 TO TEMPLENGTH
            MOVE 'Record: ' TO CICS-TO-UNIX(TEMPLENGTH:)
            ADD 8 TO TEMPLENGTH.
            MOVE TRANSACT-INPUT TO CICS-TO-UNIX(TEMPLENGTH:)
            ADD 3 TO TEMPLENGTH.
            MOVE 'was not found. Sorry, try again'
                                  TO CICS-TO-UNIX(TEMPLENGTH:).
            MOVE 0 TO TEMPLENGTH.
           GO TO SYNCPOINT.
      *
```

Figure 72 (Part 6 of 7). Sample CICS MQSeries Application Program

```
      * ------------------------------------------------------------- *
      *                   End of program                              *
      * ------------------------------------------------------------- *
/*
/. ENDJ2
// EXEC IESINSRT
/*
// IF CATALOG NE 1 OR $MRC GT 4 THEN
// GOTO NOLNK
// EXEC LNKEDT,SIZE=256K
/. NOLNK
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 72 (Part 7 of 7). Sample CICS MQSeries Application Program

```
* $$ JOB JNM=AUXDFILE,DISP=D,CLASS=O
// JOB AUXDFILK - DEFINE VSAM CLUSTER
* ****************************************
*  DEFINE AND INITIALIZE VSAM FILE
* ****************************************
// EXEC IDCAMS,SIZE=AUTO
 /*                                                          */
 /*  DELETE VSAM FILE                                        */
 /*                                                          */
     DELETE (AUTO.UNIX.KSDS) CL NOERASE PURGE  -
     CATALOG(VSESP.USER.CATALOG)
 /*                                                          */
 /*  DEFINE VSAM FILE                                        */
 /*                                                          */
 DEF CLUSTER(NAME(AUTO.UNIX.KSDS)          -
     FILE(AUXFILK)                         -
     VOL(SYSWK1 DOSRES)                    -
     RECORDS (100 100)                     -
     RECORDSIZE (0080 0080)                -
     INDEXED                               -
     KEYS(2 0)                            -
     SHR(2))                               -
     DATA (NAME (AUTO.UNIX.KSDS.§D§) CISZ(4096)) -
     INDEX (NAME (AUTO.UNIX.KSDS.§I§) CISZ(512)) -
     CATALOG(VSESP.USER.CATALOG)
/*
*
*    INITIALIZE THE AUXFILK
*
// DLBL AUXFILK,'AUTO.UNIX.KSDS',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE   -
        (SYSIPT                 -
         ENVIRONMENT            -
          (RECORDFORMAT (FIXUNB) -
           BLOCKSIZE(80)         -
           RECORDSIZE (80)))     -
        OUTFILE (AUXFILK)
01 RECORD ONE
02 RECORD TWO
03 RECORD THREE
  ...
  ...
09 RECORD NINE
10 RECORD TEN
/*
// OPTION STDLABEL=DELETE
AUXFILK
/*
// OPTION STDLABEL=ADD
// DLBL AUXFILK,'AUTO.UNIX.KSDS',,VSAM,                              X
               CAT=VSESPUC
/*
// EXEC IESVCLUP,SIZE=AUTO
D                                          AUXFILK
A AUTO.UNIX.KSDS                           AUXFILK VSESPUC
/*
/&
* $$ EOJ
```

Figure 73. Define and Initialize VSAM File

```
* $$ JOB JNM=MQFCT
// JOB MQFCT    ASSEMBLE FCT MQ
* ********************************************************************
*
* BUILDS A CICS FCT WITH SUFFIX MQ WITH MQSERIES ENTRIES.
*
*
* ********************************************************************
// LIBDEF *,CATALOG=PRD2.UNIX
// LIBDEF SOURCE,SEARCH=(PRD1.BASE,PRD1.MACLIB)
// OPTION CATAL,LIST
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
               -200K,ABOVE)'
         TITLE 'DFHFCTMQ -- TO SUPPORT MQSERIES  '
         PUNCH ' CATALOG DFHFCTMQ.OBJ REP=YES'
         DFHFCT TYPE=INITIAL,SUFFIX=MQ
         SPACE 3
         COPY  DFH$FCT       - ENTRY FOR RDO CSD DATASET         §TTA
         SPACE 3
         COPY  IESZFCTP  - DATASET ENTRIES FOR VSE/ESA - ADMINISTRATIVE
         SPACE 3
*--------------------------------------------------------------------*
*     IF YOU WANT TO ADD THE DATASET WHICH IS REQUIRED FOR           *
*     WORKSTATION FILE TRANSFER SUPPORT, THEN REMOVE THE * IN        *
*     COLUMN 1 OF THE LINE FOLLOWING THIS BOX.                       *
*--------------------------------------------------------------------*
*        COPY  IESWFCT       - ENTRY FOR WORKSTATION FILE TRANSFER SUPP
         SPACE 3
**********************************************************************
*         START OF  MQ/SERIES VSAM CLUSTER DEFINITIONS              *
**********************************************************************
* SYSTEM SSETUP FILE
MQFSSET  DFHFCT TYPE=DATASET,DATASET=MQFSSET,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,BROWSE),                                  *
               LOG=NO,                                                 *
               RSL=PUBLIC,                                             *
               BUFND=5,STRNO=5,                                        *
               RECFORM=(FIXED,BLOCKED)
* CONFIGURATION FILE
MQFCNFG  DFHFCT TYPE=DATASET,DATASET=MQFCNFG,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               LOG=YES,                                                *
               RSL=PUBLIC,                                             *
               BUFND=5,BUFNI=10,STRNO=20,                              *
               RECFORM=(FIXED,BLOCKED)
```

Figure 74 (Part 1 of 3). CICS FCT for MQSeries and Sample VSAM File

```
*--EXAMPLE OF QUEUES (INPUT FOLLOWED BY OUTPUT)
MQFI001  DFHFCT TYPE=DATASET,DATASET=MQFI001,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               RSL=PUBLIC,                                             *
               LOG=YES,                                                *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFO001  DFHFCT TYPE=DATASET,DATASET=MQFO001,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               LOG=YES,                                                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFI002  DFHFCT TYPE=DATASET,DATASET=MQFI002,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               RSL=PUBLIC,                                             *
               LOG=YES,                                                *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFO002  DFHFCT TYPE=DATASET,DATASET=MQFO002,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               LOG=YES,                                                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFI003  DFHFCT TYPE=DATASET,DATASET=MQFI003,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               LOG=YES,                                                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFO003  DFHFCT TYPE=DATASET,DATASET=MQFO003,                          *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               LOG=YES,                                                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFLOG   DFHFCT TYPE=DATASET,DATASET=MQFLOG,                           *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
MQFERR   DFHFCT TYPE=DATASET,DATASET=MQFERR,                           *
               ACCMETH=VSAM,                                           *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                *
               RSL=PUBLIC,                                             *
               BUFND=16,BUFNI=16,STRNO=16,                             *
               RECFORM=(VARIABLE,BLOCKED)
```

Figure 74 (Part 2 of 3). CICS FCT for MQSeries and Sample VSAM File

```
MQFMON   DFHFCT TYPE=DATASET,DATASET=MQFMON,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               RSL=PUBLIC,                                            *
               BUFND=16,BUFNI=16,STRNO=16,                            *
               RECFORM=(VARIABLE,BLOCKED)
*********************************************************************
*             END  OF MQ/SERIES VSAM CLUSTER DEFINITIONS            *
*********************************************************************
*********************************************************************
*         SAMPLE MQ CICS COBOL VSAM INQUIRE FILE DEFINITION         *
*********************************************************************
* MQ SAMPLE KSDS INQUIRE FILE
AUXFILK  DFHFCT TYPE=DATASET,DATASET=AUXFILK,                         *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               RSL=PUBLIC,                                            *
               BUFND=5,BUFNI=10,STRNO=2,                              *
               RECFORM=(FIXED,BLOCKED)
         SPACE 3
*--------------------------------------------------------------------*
*     FOR EXPLICIT SPECIFICATION OF VSAM LSR PARAMETERS USE SKELETON  *
*     SKSHRCTL. COPY IT INTO YOUR PRIMARY LIBRARY AND CHANGE IT AS    *
*     OUTLINED THERE. INCLUDE THE UPDATED SKELETON BELOW THIS BOX     *
*     BY DELETING THE * FOLLOWING THIS BOX.                           *
*--------------------------------------------------------------------*
*/INCLUDE SKSHRCTL
         SPACE 3
         DFHFCT TYPE=FINAL
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Figure 74 (Part 3 of 3). CICS FCT for MQSeries and Sample VSAM File

```
* $$ JOB JNM=MQSIT
// JOB MQSIT ASSEMBLE SIT MQ
* ********************************************************************
*
* BUILDS A CICS SIT WITH SUFFIX MQ.
*
*
* ********************************************************************
// LIBDEF *,CATALOG=PRD2.UNIX
// LIBDEF SOURCE,SEARCH=(PRD1.BASE,PRD1.MACLIB)
// OPTION CATAL,LIST
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
              -200K,ABOVE)'
*************************************************
*                                               *
*   5686-066 (C) COPYRIGHT IBM CORP. 1984, 1995 *
*                                               *
*************************************************
         TITLE 'DFHSITMQ  --  TO SUPPORT MQSERIES  '
         PUNCH ' CATALOG DFHSITMQ.OBJ REP=YES'
         DFHSIT TYPE=CSECT,                                            *
               ABDUMP=NO,               CICS SYSDMP ON ASRB TRANS ABEND*
               AKPFREQ=200,             ACTIVITY KEYPOINTING FREQUENCY *
               ALT=NO,                  NO APPLICATION LOAD TABLE      *
               AMXT=20,                 MAX ACTIVE TASKS               *
               APPLID=DBDCCICS,         CICS APPLICATION NAME          *
               AUTINST=(100,IESZATDX,700,200), AUTO INSTALL TERMINALS  *
               BFP=NO,                  NO BUILT-IN FUNCTIONS          *
               BMS=FULL,                FULL BASIC MAPPING SUPPORT     *
               CMXT=(10,10,10,10,10,10,10,10,10,10), 10 TASKS/ TRANS CL*
               COBOL2=NO,               USAGE OF COBOL II APPL. PRGMS  *
               DATFORM=MMDDYY,          EXTERNAL DATE DISPLAY          *
               DBP=1$,                  DYN. BACKOUT (NO LOCAL DLI I/F)*
               DBUFSZ=2000,             DYN. ADJUSTED BY CICS          *
               DCT=MQ,                                                 *
               DIP=NO,                  NO BATCH DATA INTERCHANGE      *
               DLI=NO,                  NO DL/I SUPPORT                *
               DTB=MAIN,                CHANGE TO AUX TO SAVE VIRT STOR*
               DUMP=YES,                IDUMP IN ABEND SITUATIONS      *
               DUMPDS=AUTO,             AUTO SWITCH DUMP DATA SETS     *
               EXEC=YES,                EXEC LEVEL SUPPORT             *
               EXITS=YES,               USER EXIT INTERFACE            *
               EXTSEC=YES,              CHECKING DONE BY VSE/ESA       *
               FCT=MQ,                                                 *
               GMTEXT='VSE/ESA ONLINE WITH MQSERIES SUPPORT ',         *
               GMTRAN=IEGM,             LOGON TRANSACTION ID           *
               ICP=COLD,                INTERVAL CONTROL PGM           *
               ICV=1000,                INTERVAL CONTROL EXIT TIME-MS  *
               ICVR=20000,              RUNAWAY TASK TIME              *
               ICVS=1000,               DELAY BEFORE STALL PURGE =ICV  *
               ICVTSD=200,              TERMINAL SCAN DELAY            *
               IRCSTRT=NO,              NO INTERREGION COMMUNICATION   *
               ISC=(YES,NOFS),          INTERSYSTEM COMMUNICATION      *
               JCT=MQ,                                                 *
```

Figure 75 (Part 1 of 2). CICS SIT for MQSeries

```
               LESTG=NO,               CHANGE TO 1000 IF ENV B       *
               LGNMSG=YES,             VTAM LOGON DATA               *
               MCT=NO,                 NO MONITOR CONTROL TABLE      *
               MSGLVL=2,               MESSAGES ON BOTH SYSLST/SYSLOG *
               MXT=999,                MAX NO. OF ALL CONCURRENT TASKS*
               NLT=NO,                 DEFAULT LOAD ORDER FOR NUCLEUS *
               OPNDLIM=10,             OPEN/CLOSE DESTINATION LIMIT  *
               PCDUMP=NO,              CICS SYSDMP ON ASRA TRANS ABEND*
               PCT=SP,                 SUPPLIED WITH VSE/ESA         *
               PGCHAIN=X/,             BMS CHAINING COMMAND          *
               PGCOPY=COPY/,           BMS COPY COMMAND              *
               PGPURGE=T/,             BMS PURGE COMMAND             *
               PGRET=P/,               BMS RETRIEVAL COMMAND         *
               PGSIZE=2048,            SIZE OF VIRTUAL PAGING AREA   *
               PLTPI=MI,                                             *
               PLTSD=MS,                                             *
               PL1=NO,                 NO PL/1 SUPPORT               *
               PPT=SP,                 SUPPLIED WITH VSE/ESA         *
               PRGDLAY=100,            ONE HOUR PURGE DELAY          *
               PRINT=PA1,              PRINT WITH PA1 AND TCP PRINT  *
               RAMAX=256,              SIZE OF I/O AREA FOR RA       *
               RAPOOL=10,              NUMBER OF FIXED RPLS          *
               SCS=16384,              STORAGE CUSHION-MIN OF 4 PAGES *
               SPOOL=(YES,A,A),        CICS SPOOLER ACTIVE           *
               SRT=1$,                 DEFAULT SRT                   *
               START=AUTO,             LET CICS DETERMINE STARTUP    *
               SUFFIX=MQ,                                            *
               SVD=YES,                STORAGE VIOLATION DUMP&RECOVERY*
               SYSIDNT=CICA,           CICS SYSTEM IDENTIFIER        *
               TCP=S$,                 TERMINAL CONTROL PROGRAM      *
               TCT=SP,                 BUILT AFTER CONFIGURATION     *
               TD=(3,3),               THREE BUFFERS & THREE STRINGS  *
               TRACE=(800,ON),         TRACE TABLE (FOR PROD. SET OFF)*
               TS=(,8,8),              EIGHT BUFFERS & EIGHT STRINGS  *
               TSMGSET=4,              4 MESSAGE SET ENTRIES         *
               TST=NO,                 NO TEMP STORAGE TABLE INCLUDED *
               VTAM=YES,               SUPPORT FOR VTAM TERMINALS    *
               VTPRFX=\,               CICS CLIENT TERMINAL PREFIX§LTA*
               WRKAREA=512,            COMMON WORK AREA OF THE CSA   *
               XLT=SP,                 SUPPLIED WITH VSE/ESA         *
               XRF=NO,                 NO XRF SUPPORT INCLUDED       *
               ZCP=S$,                 ALL ACCESS METHODS            *
               DUMMY=DUMMY             TO END MACRO
         END   DFHSITBA
/*
// IF $MRC GT 4 THEN
// GOTO NOLINK
// EXEC LNKEDT
/. NOLINK
/*
/&
* $$ EOJ
```

Figure 75 (Part 2 of 2). CICS SIT for MQSeries

```
* $$ JOB JNM=MQCSD,CLASS=O,DISP=D
// JOB MQCSD    DEFINE RESOURCES TO CICS FOR MQ   SAMPLE
* UPDATE CICS CSD WITH DEFINITONS FOR THE MQ SAMPLE
* (TRANSACTION INQK)
// EXEC DFHCSDUP
*
DELETE GROUP(MQINQ   )
*
* DEFINE PROGRAM MQINQ AS COBOL PROGRAM TO CICS                    *
*
DEFINE PROGRAM(MQINQ) GROUP(MQINQ   )
       LANGUAGE(COBOL) RSL(PUBLIC)
*
* DEFINE MQ PLT STARTUP AND SHUTDOWN PLT PROGRAM ENTRIES           *
*
DEFINE PROGRAM(DFHPLTMI) GROUP(MQINQ   ) RSL(PUBLIC)
DEFINE PROGRAM(DFHPLTMS) GROUP(MQINQ   ) RSL(PUBLIC)
*
* DEFINE INQK   TRANSACTION                                        *
*
DEFINE TRANSACTION(INQK) GROUP(MQINQ   ) PROGRAM(MQINQ )
*
* ADD MQINQ    GROUP TO THE STANDARD VSE/ESA LIST.                 *
*
ADD GROUP(MQINQ   ) LIST(VSELIST)
/*
/&
* $$ EOJ
```

Figure 76. CICS Define Programs and Transaction

```
* $$ JOB JNM=CICSMQ,DISP=L,CLASS=2,EOJMSG=YES
* $$ LST CLASS=A,DISP=D,RBS=100
// JOB CICSMQ   STARTUP WITH CONNECTION TO AUTO UNIX AND MQSERIES
// OPTION SADUMP=5
// UPSI 11100000
// LIBDEF *,SEARCH=PRD2.UNIX
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASED,PRD1.BASE,PRD2.PROD,        X
               PRD2.SCEECICD,PRD2.SCEECICS,PRD2.SCEEBASD,            X
               PRD2.SCEEBASE,PRD2.CICSR,PRD2.DBASE,                  X
               PRD1.MACLIBD,PRD1.MACLIB),PERM
// LIBDEF DUMP,CATALOG=SYSDUMP.F2
// SETPARM TPMODE=''
// SETPARM XNCPU=''
// SETPARM XMODEF2=''
// EXEC PROC=$COMVAR,XNCPU
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF2=ACTIVE'
/*
// EXEC PROC=CPUVAR&XNCPU,TPMODE,XMODEF2
// SETPFIX LIMIT=144K
* ------------------------------------------------------------------*
* Licensed Materials - Property of IBM                              *
*                                                                   *
* 5787-ECX                                                          *
* (C) Copyright IBM Corp. 1993, 1996                                *
*                                                                   *
* US Government Users Restricted Rights - Use, duplication or       *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
* ------------------------------------------------------------------*
*   Sample JCL file definition for CICS deck                        *
*   The DLBL statements in this JCL corresponds to entries in CICSFCT
*   therefore if there are any new file ids to be added in here,    *
*   it must also be added into the corresponding JCL                *
*                                                                   *
* ------------------------------------------------------------------*
// DLBL MQFSSET,'MQSERIES.MQFSSET',0,VSAM,CAT=VSESPUC
// DLBL MQFCNFG,'MQSERIES.MQFCNFG',0,VSAM,CAT=VSESPUC
// DLBL MQFIO01,'MQSERIES.MQFIO01',0,VSAM,CAT=VSESPUC
// DLBL MQFIO02,'MQSERIES.MQFIO02',0,VSAM,CAT=VSESPUC
// DLBL MQFIO03,'MQSERIES.MQFIO03',0,VSAM,CAT=VSESPUC
// DLBL MQFOO01,'MQSERIES.MQFOO01',0,VSAM,CAT=VSESPUC
// DLBL MQFOO02,'MQSERIES.MQFOO02',0,VSAM,CAT=VSESPUC
// DLBL MQFOO03,'MQSERIES.MQFOO03',0,VSAM,CAT=VSESPUC
// DLBL MQFERR,'MQSERIES.MQFERR',0,VSAM,CAT=VSESPUC
// DLBL MQFLOG,'MQSERIES.MQFLOG',0,VSAM,CAT=VSESPUC
// DLBL MQFMON,'MQSERIES.MQFMON',0,VSAM,CAT=VSESPUC
* ------------------------------------------------------------------*
*   End of sample jcl file definition for cics deck                 *
* ------------------------------------------------------------------*
// EXEC PROC=DTRCICST                 ASSGNS FOR CICS FILES
// EXEC PROC=DTRINFOA                 ASSGNS FOR INFO ANAL FILES
// EXEC PROC=DTRICCF                  ASSGN FOR DTSFILE
// ASSGN SYS005,UA
// ASSGN SYS006,UA
```

Figure 77 (Part 1 of 2). CICS MQSeries Startup Job Stream

```
// ASSGN SYS007,UA
// ASSGN SYS008,UA
// ASSGN SYS009,SYSLOG
// ASSGN SYS019,DISK,VOL=DOSRES,SHR
LOG
// ID USER=FORSEC,PWD=FORSEC
NOLOG
// EXEC DTSANALS                        RECOVER IF DTSFILE DESTROYED
RECOVER OPT
/*
// IF TPMODE=B THEN
// GOTO BTAM
*   WAITING FOR VTAM TO COME UP
// EXEC IESWAITT
// GOTO START
/. BTAM
// ASSGN SYS023,IGN
// ASSGN SYS024,IGN
// ASSGN SYS025,IGN
// ASSGN SYS026,IGN
// ASSGN SYS027,IGN
// EXEC PROC=DTRIBTAM                   ASSGN BTAM TERMINALS
* AT THIS POINT, YOU MAY OPTIONALLY ASSIGN SYS027 TO
* A 3270 PRINTER SUPPORTED AS A 3286. ENTER:
* "// ASSGN SYS027,CUU"  WHERE CUU IS THE PRINTER ADDRESS
// PAUSE
/. START
*  IF XMODEF2 = COLD THEN
// GOTO COLD
// EXEC DFHSIP,PARM='SIT=MQ,GRPLIST=VSELIST,ISC=YES,$END',DSPACE=2M
/*
// GOTO END
/. COLD
// EXEC DFHSIP,PARM='SIT=MQ,GRPLIST=VSELIST,START=COLD,ISC=YES,$END',  X
               DSPACE=2M
/*
/. END
// EXEC DTRSETP,PARM='CPUVAR&XNCPU;;SET XSTATF2=INACTIVE'
/*
/&
* $$ EOJ
```

Figure 77 (Part 2 of 2). CICS MQSeries Startup Job Stream

Appendix F. Special Notices

This redbook will help you install, tailor and configure the new OS/390 Automated UNIX System Option for VM, VSE and OS/390 (Auto UNIX System).

Guidance and samples are provided to help you with the installation, configuration and connectivity of the Auto UNIX System, as well as showing how you can use the product in a normal installation with VSE/ESA or with VM/ESA.

For communication with other systems, either the Virtual Telecommunications Access Method (VTAM) or the Transmission Control Protocol/Internet Protocol (TCP/IP) communication protocols will be used.

The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 Automated UNIX System Option for VM, VSE and OS/390. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 Automated UNIX System Option for VM, VSE and OS/390 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.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AFP
APPN
CICS
DFSMS
IBM
MQ
NetView
OpenEdition
OS/390
Powered by S/390
RS/6000
System/390
VSE/ESA
WebSphere
AIX
AS/400
DB2
ESCON
IMS
MQSeries
Network Station
OS/2
Parallel Sysplex
RACF
S/390
VM/ESA
VTAM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

C++ is a trademark of American Telephone and Telegraph Company, Incorporated

Intel is a trademark of Intel Corporation.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Lotus is a trademark of Lotus Development Corporation

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Network File System is a trademark of Sun Microsystems, Incorporated

NFS is a trademark of Sun Microsystems, Incorporated

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

POSIX is a trademark of Institute of Electrical and Electronic Engineers.

PostScript is a trademark of Adobe Systems, Incorporated

Sun Microsystems is a trademark of Sun Microsystems, Incorporated.

Tivoli, TME, and TME 10 are trademarks of Tivoli

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

X/Open is a trademark of X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix G. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 International Technical Support Organization Publications

For information on ordering ITSO publications see “How to Get ITSO Redbooks” on page 165.

G.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

G.3 Other Publications

These publications are also relevant as further information sources:

- *OS/390 Automated UNIX System Option for VM, VSE and OS/390 General Information*, GA22-7362
- *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Installation and Setup*, GA22-7363
- *OS/390 Automated UNIX System Option for VM, VSE and OS/390 Licensed Program Specification*, GA22-7364
- *OS/390 UNIX System Services Planning*, SC28-1890-05
- *OS/390 UNIX System Services Command Reference*, SC28-1892
- *OS/390 Application Enabling Technology: Administration and Programming*, GC28-1993
- *VM/ESA V2R3.0 TCP/IP FL310 User*, SC24-5848
- *DFFSMS/MVS Version 1 Release 2 Network File System Customization and Operation*, SC26-7029
- *MQSeries Application Programming Guide*, SC33-0807
- *MQSeries for VSE/ESA User's Guide*, SC33-1142
- *TCP/IP for VSE/ESA User*, SC33-6601

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO 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-ROMs 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 via e-mail including information from the redbook order form to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755	
Canada (toll free)	1-800-IBM-4YOU	
Outside North America	(long distance charges apply)	
(+45) 4810-1320 - Danish	(+45) 4810-1220 - French	(+45) 4810-1270 - Norwegian
(+45) 4810-1420 - Dutch	(+45) 4810-1020 - German	(+45) 4810-1120 - Spanish
(+45) 4810-1540 - English	(+45) 4810-1620 - Italian	(+45) 4810-1170 - Swedish
(+45) 4810-1670 - Finnish		

This information was current at the time of publication, but is continually subject to change. The latest information for customers may be found at http://www.redbooks.ibm.com/ and for IBM employees at http://w3.itso.ibm.com/.

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 also view redbook, residency and workshop announcements at http://inews.ibm.com/.

IBM Redbook Fax Order Form

Fax your redbook orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

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

abbreviation. An ordered and shortened representation of data that retains the identity of the data element that is represented.

access control. In computer security, ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways. See discretionary access control, identity-based access control, information flow control, mandatory access control, resource-based access control.

access list. Synonym for access control list.

access method. (1) A technique to obtain the use of data, storage, or the use of an input/output channel to transfer data; for example, random access method, sequential access method. (2) The technique that is used to locate data stored on a physical medium. (3) A technique for moving data between main storage and input/output devices. (4) The way that a system refers to records in files; the reference can be consecutive (records are referred to one after another in the order in which they appear in the file) or it can be random (the individual records are referred to in any order).

acknowledge. To answer. To respond to a poll, address, or message.

acquire. (1) In VTAM programs, to take over resources that were formerly controlled by an access method in another domain, or to resume control of resources that were controlled by that domain but released. Contrast with release. See also resource takeover. (2) In a VTAM application program, to initiate and establish a session with another logical unit (LU). The acquire process begins when the application program issues a macroinstruction.

activate. (1) To put a device into an operational state. (2) To pass control to a program, procedure, or routine. (3) To make a resource ready to perform its function. Contrast with deactivate.

adapter. (1) A mechanism for attaching parts; for example, parts having different diameters. (2) A part that electrically or physically connects a device to a computer or to another device.

advanced program-to-program communication. The general facility characterizing the LU 6.2 architecture and its various implementations in products.

alias. (1) An alternate label; for example, a label and one or more aliases may be used to refer to the same data element or point in a computer program. (2) An alternate name for a member of a partitioned data set.

append. A function or mode that enables a user to add a new document or character string to the end of previously entered text.

application program. (1) A program that is specific to the solution of an application problem. Synonymous with application software. (2) A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll. (3) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

Application Programming Interface. The Control Program Facility (CPF) graphics routines that perform basic graphics tasks when called by high-level language application programs.

APPN network. Advanced Peer-to-Peer Networking (APPN) network.

APPN network node. Advanced Peer-to-Peer Networking (APPN) network node.

argument. A parameter passed between a calling program and a called program.

attach. (1) In programming, to create a task that can be executed asynchronously with the execution of the mainline code. (2) To connect a device logically to a ring network.

attachment. A device or feature attached to a processing unit, including required adapters. Contrast with adapter.

attention. An occurrence external to an operation that could cause an interruption of the operation.

audible alarm. An alarm that is sounded when designated events occur that require operator attention or intervention before continuing system operation.

audio. Pertaining to the portion of recorded information that can be heard.

authority. The right to access objects, resources, or functions.

auxiliary storage. (1) All addressable storage, other than main storage, that can be accessed by means of an input/output channel; for example, storage on magnetic tape or direct access devices. Synonymous with external storage, secondary storage. (2) Contrast with main storage. (3) Synonym for external storage.

B

batch processing. (1) The processing of data or the accomplishment of jobs accumulated in advance, in such a manner that the user cannot further influence processing while it is in progress. (2) The processing of data accumulated over a period of time.

binary format. Representation of a decimal value in which each field must be 2 or 4 bytes long. The sign (+ or -) is in the far left bit of the field, and the number value is in the remaining bits of the field. Positive numbers have a 0 in the sign bit and are in true form. Negative numbers have a 1 in the sign bit and are in twos complement form.

block size. (1) The number of data elements in a block. (2) A measure of the size of a block, usually specified in units such as records, words, computer words, or characters. (3) Synonymous with block length.

browse. To rapidly scan information on a display screen by scrolling or paging. Synonymous with high-speed scan, high-speed scroll.

button. A graphical mechanism in a window that, when selected, results in an action; for example, a list button produces a list of choices.

byte. A group of 8 adjacent binary digits that represent one EBCDIC character.

C

cache. (1) A special-purpose buffer storage, smaller and faster than main storage, used to hold a copy of instructions and data obtained from main storage and likely to be needed next by the processor. (2) A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time. (3) An optional part of the directory database in network nodes where frequently used directory information may be stored to speed directory searches.

call. The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point.

cancel. To end a task before it is completed.

carry. One or more digits, produced in connection with an arithmetic operation on one digit place of two or more numerals in positional notation, that are forwarded to another digit place for processing there.

catalog. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, and blocking factors.

central site. In a distributed data processing network, the central site is usually defined as the focal point in a communications network for alerts, application design, and remote system management tasks such as problem management.

chain. (1) A group of logically linked user data records processed by LU 6.2. (2) A group of request units delimited by begin-chain and end-chain. Responses are always single-unit chains.

channel. A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment.

channel-to-channel. A method of connecting two computing devices.

class of service. A set of characteristics (such as route security, transmission priority, and bandwidth) used to construct a route between session partners. The class of service is derived from a mode name specified by the initiator of a session.

click. To press and release a button on a pointing device without moving the pointer off the choice. See double-click.

client. (1) A user. (2) A functional unit that receives shared services from a server.

close. A data manipulation function that ends the connection between a file and a program. Contrast with open.

cluster. In systems with VSAM, a named structure consisting of a group of related components; for example, a data component with its index component.

code page. (1) An assignment of graphic characters and control function meanings to all code points; for example, assignment of characters and meanings to 256 code points for an 8-bit code, assignment of characters and meanings to 128 code points for a 7-bit code. (2) In the Print Management Facility, a font library member that associates code points and character identifiers. A code page also identifies invalid code points. (3) A particular assignment of hexadecimal identifiers to graphic characters. (4) In AFP support, a font file that associates code points and graphic character identifiers.

command. A statement used to request a function of the system. A command consists of the command name abbreviation, which identifies the requested function, and its parameters.

command processing. The reading, analyzing, and performing of commands issued via a console or through an input stream.

command prompt. A displayed character or string of characters that indicates that a user may enter a command to be processed.

compact. Synonym for compress.

compiler. A program that translates instructions written in a high-level programming language into machine language.

concatenate. (1) To link together. (2) To join two character strings.

configuration file. A file that specifies the characteristics of a system or subsystem.

connected. In VTAM programs, pertaining to a physical unit (PU) or logical unit (LU) with an active physical path to the host processor containing the system services control point (SSCP) that controls the PU or LU.

connectivity. (1) The capability of a system or device to be attached to other systems or devices without modification. (2) The capability to attach a variety of functional units without modifying them.

consecutive. In a process, pertaining to two events that follow one another without the occurrence of any other event between them. Contrast with sequential.

console. A part of a computer used for communication between the operator or maintenance engineer and the computer.

control statement. In programming languages, a statement that is used to alter the continuous sequential execution of statements; a control statement may be a conditional statement, such as IF, or an imperative statement, such as STOP.

conversion. (1) In programming languages, the transformation between values that represent the same data item but belong to different data types. Information may be lost due to conversion since accuracy of data representation varies among different data types. (2) The process of changing from one method of data processing to another or from one data processing system to another.

Customer Information Control System. An IBM licensed program 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.

customization. The process of designing a data processing installation or network to meet the requirements of particular users.

D

database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users. (2) A collection of interrelated data organized according to a database schema to serve one or more applications. (3) A collection of data fundamental to a system. (4) A collection of data fundamental to an enterprise.

data connection. The interconnection of two data terminal equipment (DTEs) by means of switched tandem data circuits to enable data transmission to take place between DTEs.

data exchange. The use of data by more than one program or system. Data recorded or transmitted in a format is referred to as exchange data.

Data Facility Storage Management Subsystem. An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

data file. A collection of related data records organized in a specific manner; for example, a payroll file (one record for each employee, showing such information as rate of pay and deductions) or an inventory file (one record for each inventory item, showing such information as cost, selling price, and number in stock).

data integrity. (1) The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur. (2) Preservation of data for its intended use.

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 name. The term or phrase used to identify a data set. See also qualified name.

data transfer. The movement, or copying, of data from one location and the storage of the data at another location.

datagram. In TCP/IP, the basic unit of information passed across the Internet environment. A datagram contains a source and destination address along with the data. An Internet Protocol (IP) datagram consists of an IP header followed by the transport layer data.

debugging. Acting to detect and correct errors in software or system configuration.

default. Pertaining to an attribute, condition, value, or option that is assumed when none is explicitly specified.

default value. A value assumed when no value has been specified. Synonymous with assumed value.

definition statement. (1) In VTAM, the statement that describes an element of the network. (2) In NCP, a type of instruction that defines a resource to the NCP.

delimiter. A character used to indicate the beginning and end of a character string.

descriptor. A word or phrase used to categorize or index information. Synonymous with keyword.

desktop. A folder that fills the entire screen and holds all the objects with which the user can interact to perform operations on the system.

destination. An external logical unit (LU) or application program to which messages or other data are directed.

device address. (1) The identification of an in-put/out-put device by its channel and unit number. (2) In data communication, the identification of any device to which data can be sent or from which data can be received.

device type code. The four- or five-digit code to be used for defining an I/O device to a computer system.

dialog. (1) The interaction between a user and a computer. (2) In an interactive system, a series of related inquiries and responses similar to a conversation between two people.

digit. A character that represents a nonnegative integer; for example, one of the characters 0 through F in the hexadecimal numeration system.

direct access. The capability to obtain data from a storage device, or to enter data into a storage device, in a sequence independent from their relative position, by means of addresses indicating the physical position of the data.

directory. A table of identifiers and references to the corresponding items of data.

directory tree. An outline of all the directories and subdirectories on the current drive.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disconnect. (1) To disengage apparatus used in a connection and restore it to its ready condition when not in use. Synonymous with release. (2) To break a connection, physically or electrically.

discrete. (1) Pertaining to data that consists of distinct elements such as characters, or to physical quantities having a finite number of distinctly recognizable values. (2) Contrast with analog.

diskette. A thin, flexible magnetic disk and a semi-rigid protective jacket, in which the disk is permanently enclosed.

disposition. In file processing, the process of specifying whether a file is new, old, or shared, and how the file is to be shared.

domain. That part of a computer network in which the data processing resources are under common control.

domain name. In TCP/IP, a name of a host system in a network. A domain name consists of a sequence of subnames separated by a delimiter character.

domain name server. In TCP/IP, a server program that supplies name-to-address translation by mapping domain names to internet addresses. Synonymous with name server.

drag. In SAA Common User Access, to use a pointing device to move an object; for example, clicking on a window border, and dragging it to make the window larger.

drop. To fix the position of an object that is being dragged by releasing the select button on the pointing device. See also drag.

dump. To record, at a particular instant, the contents of all or part of one storage device in another storage device. Dumping is usually for the purpose of debugging.

duplicate. To copy from a source to a destination that has the same physical form as the source; for example, to punch new punched cards with the same pattern of holes as an original punched card. Synonymous with reproduce.

E

edit. To add, change, delete, or rearrange data and to perform operations such as code conversion and zero suppression.

editor. A computer program designed to perform such functions as rearrangement, modification, and deletion of data in accordance with prescribed rules.

emulator. (1) A combination of programming techniques and special machine features that permits a computing system to execute programs written for a different system. See also integrated emulator, terminal emulator. (2) A program that causes a computer to act as a workstation attached to another system.

enabled. Pertaining to a state of the processing unit that allows the occurrence of certain types of interruptions. Synonymous with interruptible.

encoding. The conversion of data to machine-readable format; the final step in the process of converting an analog signal into a digital signal. The three steps are: sampling, quantizing, and encoding.

encryption. In computer security, the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process.

entry point. (1) In a database, the record that is first accessed upon entry into a database, caused by a user′s command. (2) The address or label of the first instruction executed on entering a computer program, routine, or subroutine. A computer program, routine, or subroutine may have a number of different entry points, each perhaps corresponding to a different function or purpose. Synonymous with entrance, entry. (3) In a routine, any place to which control can be passed.

erase. To remove data from a data medium. Erasing is usually accomplished by overwriting the data or deleting the references.

error message. An indication that an error has been detected.

escape. To return to the original level of a user interface.

escape character. A code extension character used, in some cases with one or more succeeding characters, to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

event. An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an input/output operation.

executable program. (1) A program that has been link-edited and therefore can be run in a processor. (2) The set of machine language instructions that constitute the output from the compilation of a source program.

execution. The process of carrying out an instruction or instructions of a computer program by a computer.

exit. To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on.

export. To copy data onto removable media.

external. In programming languages, pertaining to a language object that has a scope that extends beyond one module; for example, the entry names of a module.

F

failure. The termination of the ability of a functional unit to perform its required function. Synonymous with malfunction. Contrast with error, fault, mistake.

fetch. (1) To obtain load modules from auxiliary storage and load them into main storage. (2) In virtual storage systems, to bring load modules or program phases from auxiliary storage into virtual storage.

field. The smallest identifiable part of a record.

file. A named set of records stored or processed as a unit.

file control table. A table containing the characteristics of files processed by CICS file management.

file definition. (1) Information that describes the contents and characteristics of a file. (2) In VM, equating a CMS file identifier (file name, file type, file mode) with an OS data set name via the FILEDEF command, or equating a VSE file-id with a CMS file identifier via the DLBL command.

file management. (1) Creation and maintenance of files by means of a computer. (2) In personal computers, the use of application software to access, create, modify, store, and retrieve files and to obtain documents such as reports and mailing lists.

file name. (1) A name assigned or declared for a file. (2) The name used by a program to identify a file.

file transfer. In remote communications, the transfer of one or more files from one system to another over a communications link.

File Transfer Protocol. In TCP/IP, an application protocol used for transferring files to and from host computers. FTP requires a user ID and possibly a password to allow access to files on a remote host system. FTP assumes that the Transmission Control Protocol is the underlying protocol.

fill. In computer graphics, a designated area of the screen that is flooded with a particular color.

fixed length. A specified length for a record or field that cannot be changed.

folder. A file used to store and organize documents or electronic mail.

foreign host. Synonym for remote host.

function. A mathematical entity whose value, that is, the value of the dependent variable, depends in a specified manner on the values of one or more independent variables, not more than one value of the dependent variable corresponding to each permissible combination of values from the respective ranges of the independent variables.

G

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.

generate. To produce a computer program by selecting subsets from skeletal code under the control of parameters.

get. To obtain a record from an input file.

grant. A DB2 process that authorizes users to access data.

graphics. (1) The making of charts and pictures. (2) Pertaining to charts, tables, and their creation.

group. A set of related records that have the same value for a particular field in all records.

H

hardware. (1) All or part of the physical components of an information processing system, such as computers or peripheral devices. (2) The equipment, as opposed to the programming, of a system. (3) Contrast with software.

header. (1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

header file. Synonym for include file.

header record. A record containing common, constant, or identifying information for a group of records that follows. Synonymous with header table, heading record. Contrast with detail record.

hexadecimal. Pertaining to a system of numbers to the base 16; hexadecimal digits range from 0 through 9 and A through F, where A represents 10 and F represents 15.

host system. (1) A data processing system used to prepare programs and operating environments for use on another computer or controller. (2) The data processing system to which a network is connected and with which the

I

icon. A graphic symbol, displayed on a screen, that a user can point to with a device such as a mouse in order to select a particular function or software application. Synonymous with pictogram.

identifier. One or more characters used to identify or name a data element and possibly to indicate certain properties of that data element.

image. (1) An electronic representation of a picture produced by means of sensing light, sound, electron radiation, or other emanations coming from the picture or reflected by the picture. An image can also be generated directly by software without reference to an existing picture. (2) An electronic representation of an original document recorded by a scanning device.

implementation. The system development phase at the end of which the hardware, software and procedures of the system considered become operational.

index. (1) A list of the contents of a file or of a document, together with keys or references for locating the contents. (2) In programming, an integer that identifies the position of a data item in a sequence of data items.

information management. In an information processing system, the functions of controlling the acquisition, analysis, retention, retrieval, and distribution of information.

initialization. (1) The operations required for setting a device to a starting state, before the use of a data medium, or before implementation of a process. (2) Preparation of a system, device, or program for operation.

input. (1) Pertaining to a device, process, or channel involved in an input process, or to the associated data or states. The word ″input″ may be used in place of ″input data,″ ″input signal″, ″input process″, when such a usage is clear in a given context. (2) Pertaining to a functional unit or channel involved in an input process or to the data involved in such a process.

input stream. (1) A sequence of control statements and data submitted to a system from an input unit. Synonymous with input job stream, job input stream. (2) Synonym for job stream. Contrast with output stream.

insert. A function or mode that enables the introduction of additional characters within previously entered text.

installation. In system development, preparing and placing a functional unit in position for use.

integrated. Pertaining to a feature that is part of a device. Synonymous with built-in.

integrity. The protection of systems, programs, and data from inadvertent or malicious destruction or alteration. See application integrity, data integrity, system integrity.

interactive. (1) Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational, that is, a continuous dialog exists between user and system. Contrast with batch. (2) Pertaining to the exchange of information between a user and a computer.

interface. (1) A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions. (2) Hardware, software, or both, that links systems, programs, or devices.

Internet. A wide area network connecting thousands of disparate networks in industry, education, government, and research. The Internet network uses TCP/IP as the standard for transmitting information.

interrupt. A suspension of a process, such as execution of a computer program caused by an external event, and performed in such a way that the process can be resumed.

invocation. (1) The activation of a program or procedure. (2) An execution of a program.

J

job stream. The sequence of representation of jobs or parts of jobs to be performed, as submitted to an operating system. Synonymous with input stream, run stream.

K

key field. (1) In VSAM, a field, located in the same position in each record of a file or data set, whose content is used for the key of a record. (2) A field in a record whose contents are used to sequence records of a particular type within a file member.

keyword. (1) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier. (2) One of the predefined words of an artificial language.

L

label. (1) In programming languages, a language construction naming a statement and including an identifier. (2) An identifier within or attached to a set of data elements. (3) A record that identifies a volume on tape, disk, or diskette or that identifies a file on the volume. (4) An identifier of a command generally used for branching.

label area. Synonym for label information area.

librarian. In VSE, the set of programs that maintains, services, and organizes the system and private libraries.

library member. (1) A named collection of records or statements in a library. (2) In VSE, the smallest unit of data that can be stored into and retrieved from a sublibrary.

line. The portion of a data circuit external to data circuit-terminating equipment (DCE), that connects the DCE to a data switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE.

link. (1) In computer programming, the part of a program, in some cases a single instruction or an address, that passes control and parameters between separate portions of the computer program. Synonymous with linkage. (2) The combination of the link connection (the transmission medium) and two link stations, one at each end of the link connection. A link connection can be shared among multiple links in a multipoint or token-ring configuration. (3) In TCP/IP, a communications line. A TCP/IP link may share the use of a communications line with SNA.

load. To bring all or part of a computer program into memory from auxiliary storage so that the computer can run the program.

local terminal. Synonym for channel-attached terminal.

log in. (1) To begin a session at a display station. (2) To begin a session with a remote resource. (3) The act of identifying oneself as authorized to use a resource. Often the system requires a user ID and password to check authorization to use the resource.

logging. The recording of data about specific events.

logical. Pertaining to content or meaning as opposed to location or actual implementation.

logical record. (1) A set of related data or words considered to be a record from a logical viewpoint. (2) A record from the standpoint of its content, function,

and use rather than its physical attributes, that is, a record defined in terms of the information it contains. (3) In CICS/VS, a data record sent by one transaction program to another. The length of the record is contained in a two-byte field immediately preceding the record. (4) In VSAM, a unit of information normally pertaining to a single subject; a logical record is the user record requested of or given to the data management function.

logon. (1) The procedure by which a user begins a terminal session. (2) In VTAM, an unformatted session-initiation request for a session between two logical units.

M

macro. (1) A possibly parameterized specification for a statement sequence to replace a macro call. A macrodefinition may be considered as a procedure to be executed by a macrogenerator yielding the statement sequence. (2) A set of statements defining the name of, format of, and conditions for generating a sequence of assembler statements from a single source statement.

main storage. (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) That part of internal storage into which instructions and other data must be loaded for subsequent execution or processing. Synonymous with main memory. (3) The part of a processing unit where programs are run. Contrast with control storage. (4) Synonymous with memory. (5) Contrast with auxiliary storage.

mainframe. (1) A computer, usually in a computer center, with extensive capabilities and resources to which other computers may be connected so that they can share facilities. (2) A large computer, in particular one to which other computers can be connected so that they can share facilities the mainframe provides; for example, a System/390 computing system to which personal computers are attached so that they can upload and download programs and data.

major node. In VTAM, a set of resources that can be activated and deactivated as a group.

mask. A pattern of characters used to control retention or elimination of portions of another pattern of characters.

master file. A file that is used as an authority in a given job and that is relatively permanent, even though its contents may change.

matching. The technique of comparing the keys of two or more records to select items for a particular stage of processing or to reject invalid records.

member. (1) A partition of a partitioned data set. (2) In VSE, the smallest unit of data that can be stored in and retrieved from a sublibrary. See also library member. (3) A data object in a structure, a union, or a library.

memory. (1) All of the addressable storage space in a processing unit and other internal storages that is used to execute instructions. (2) Synonymous with main storage.

menu. A list of options displayed to the user by a data processing system, from which the user can select an action to be initiated.

menu bar. The area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus.

merge. To combine the items of two or more sets that are each in the same given order into one set in that order.

message. A communication sent from a person or program to another person or program.

message box. A type of window that shows messages to users.

message queue. (1) A list of messages awaiting processing or waiting to be sent to a terminal. (2) A queue of messages within a message data set waiting to be transmitted to the host system or to a particular terminal operator.

migration. (1) The process of moving data from one computer system to another without converting the data. (2) Installation of a new version or release of a program to replace an earlier version or release.

minidisk. Synonym for virtual disk

monitor. (1) A device that observes and records selected activities within a data processing system for analysis. Possible uses are to indicate significant departure from the norm, or to determine levels of utilization of particular functional units. (2) Software or hardware that observes, supervises, controls, or verifies operations of a system.

mount. To place a data medium in a position to operate.

N

name server. In TCP/IP, synonym for domain name server.

native. Deprecated term for IBM-supplied, basic, required, or stand-alone.

network. (1) A configuration of data processing devices and software connected for information

interchange. (2) A group of nodes and the links interconnecting them.

network application. The use to which a network is put, such as data collection or inquiry/update.

network node. Synonym for Advanced Peer-to-Peer Networking (APPN) network node.

node. (1) In a network, a point at which one or more functional units connect channels or data circuits. (2) In network topology, the point at an end of a branch. (3) An endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. (4) In VTAM, a point in a network defined by a symbolic name.

numeric. Pertaining to any of the digits 0 through 9.

O

object. (1) A passive entity that contains or receives data; for example, bytes, blocks, clocks, fields, files, directories, displays, keyboards, network nodes, pages, printers, processors, programs, records, segments, words. Access to an object implies access to the information it contains. (2) Something that a user works with to perform a task. Text and graphics are examples of objects.

object name. The name of an object. Contrast with qualified object name.

offset. (1) The number of measuring units from an arbitrary starting point in a record, area, or control block, to some other point. (2) The distance from the beginning of an object to the beginning of a particular field.

online. (1) Pertaining to the operation of a functional unit when under the direct control of the computer. (2) Pertaining to a user′s ability to interact with a computer. (3) Pertaining to a user′s access to a computer via a terminal.

open. The function that connects a file to a program for processing.

operating system. 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.

operator. (1) A symbol that represents an operation to be done. (2) In a language statement, the lexical entity that indicates the action to be performed on operands. See also definition statement.

operator console. A functional unit containing devices that are used for communications between a computer operator and a computer.

option. A specification in a statement that may be used to influence the execution of the statement.

output. (1) Pertaining to a device, process, or channel involved in an output process, or to the associated data or states. The word ″output″ may be used in place of ″output data,″ ″output signal″, ″output process″, when such a usage is clear in a given context. (2) Data that has been processed. (3) Data transferred from storage to an output device.

overhead. In a computer system, the time, operations, and resources used for operating system functions, rather than for application programs.

overlay. (1) A program segment that is loaded into main storage and replaces all or part of a previously loaded program segment. (2) To write over existing data in storage. (3) A collection of predefined data such as lines, shading, text, boxes, or logos, that can be merged with variable data on a page while printing.

override. (1) A parameter or value that replaces a previous parameter or value. (2) The attributes specified at run time that change the attributes specified in the file description or in the program. (3) To specify attributes at run time that change the attributes specified in the file description or in the program. (4) To replace a parameter or value.

P

pad. To fill unused portions of a field with dummy data, usually zeros or blanks.

paging. In System/390 virtual storage systems, the process of transferring pages between real storage and external page storage.

panel. A set of logically related information displayed on the screen for the purpose of communicating information to or from a computer user.

parameter. A variable that is given a constant value for a specified application and that may denote the application.

partition. (1) A fixed-size division of storage. See main storage partition, virtual partition. (2) In VSE, a division of the virtual address area that is available for program execution. (3) On an IBM personal computer hard disk, one of four possible storage areas of variable size; one may be accessed by DOS and each of the others may be assigned to another operating system.

partner. In data communications, the remote application program or the remote computer.

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.

path. (1) The route used to locate files; the storage location of a file. A fully qualified path lists the drive identifier, directory name, subdirectory name (if any), and file name with the associated extension. (2) In a network, any route between any two nodes. A path may include more than one branch. (3) The series of transport network components (path control and data link control) that are traversed by the information exchanged between two network accessible units. See also explicit route (ER), route extension, and virtual route (VR).

performance. One of the two major factors, together with facility, on which the total productivity of a system depends. Performance is largely determined by a combination of throughput, response time, and availability.

phase. In VSE, the smallest complete unit of executable code that can be loaded into virtual storage.

physical. Pertaining to actual implementation or location as opposed to conceptual content or meaning.

platform. (1)The operating system environment in which a program runs. (2) In computer technology, the principles on which an operating system is based.

pointer. A data element that indicates the location of another data element.

pop-up. A box on the display screen that displays information or asks the user to make choices.

port. A specific communications end point within a host.A port is identified by a port number.

port number. In TCP/IP, a 16-bit number used to communicate between TCP and a higher-level protocol or application. Some protocols, such as the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP), use the same port number in all TCP/IP implementations.

prefix. A code at the beginning of a message or record.

print queue. A list of items waiting to be printed.

priority. (1) A rank assigned to a task that determines its precedence in receiving system resources. (2) The relative significance of one job to other jobs in competing for allocation of resources.

procedure. (1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call. (2) The description of the course of action taken for the solution of a problem. (3) A set of related control statements that cause one or more programs to be performed.

process. The performance of logical operations and calculations on data, including temporary retention of data in processor storage while the data is being operated on.

processor. In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit.

profile. (1)Data that describes the significant characteristics of a user, a group of users, or one or more computer resources. (2) In computer security, a description of the characteristics of an entity to which access is controlled. (3) A description of the control available to a particular network operator.

PROFILE EXEC. In VM, a special EXEC procedure with a filename of PROFILE. The procedure is normally executed immediately after CMS is loaded into a virtual machine. It contains CP and CMS commands that are to be issued at the start of every terminal session.

programming. The design, writing, modifying, and testing of programs.

prompt. A visual or audible message sent by a program to request the user's response.

protection. An arrangement for restricting access to or use of all or part of a computer system.

protocol. (1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) In Open Systems Interconnection architecture, a set of semantic and syntactic rules that determine the behavior of entities in the same layer in performing communication functions. (3) In SNA, the meanings of and the sequencing rules for requests and responses used for managing the network, transferring data, and synchronizing the states of network components.

push button. In SAA Advanced Common User Access architecture, a rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected.

put. To place a single data record into an output file.

Q

qualifier. (1) A modifier that makes a name unique. (2) All names in a qualified name other than the right-most, which is called the simple name.

query. (1) A request for data from a database, based on specified conditions; for example, a request for availability of a seat on a flight reservation system. (2) The process by which a master station asks a slave station to identify itself and to give its status. (3) In interactive systems, an operation at a terminal that elicits a response from the system.

queue. (1) A list constructed and maintained so that the next data element to be retrieved is the one stored first. (2) A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed.

queuing. The programming technique used to handle messages awaiting delivery. See also queue.

quit. A key, command, or action that tells a system to return to a previous state or stop a process.

R

radio button. In SAA Advanced Common User Access architecture, a circle with text beside it. Radio buttons are combined to show a user a fixed set of choices from which only one can be selected. The circle is partially filled when a choice is selected.

range. The set of values that a quantity or function may take.

read access. In computer security, permission to read information.

receive. (1) To obtain and store data. (2) In systems with VTAM, to obtain a message transmitted from a terminal to the computer over a line. Contrast with send.

receiver. A person or thing that receives something. See also addressee. Contrast with sender.

reconfiguration. (1) A change made to a given configuration of a computer system; for example, isolating and bypassing a defective functional unit, connecting two functional units by an alternative path. Reconfiguration is effected automatically or manually and can be used to maintain system integrity. (2) The process of placing a processing unit, main storage, and channels offline for maintenance, and adding or removing components.

record format. The definition of how data are structured in the records contained in a file. The definition includes record name, field names, and field descriptions, such as length and data type. The record formats used in a file are contained in the file description.

record length. Synonym for record size.

recovery procedure. (1) A process in which a specified data station attempts to resolve conflicting or erroneous conditions arising during the transfer of data. (2) An action performed by the operator when an error occurs to permit processing to continue.

refresh. The process of repeatedly producing a display image on a display surface so that the image remains visible.

register. A part of internal storage having a specified storage capacity and usually intended for a specific purpose.

remote. (1) Pertaining to a system, program, or device that is accessed through a telecommunication line. Contrast with local. (2) Synonym for link-attached.

remote host. Any host on a network except the one at which a particular operator is working. Synonymous with foreign host.

remote system. Any other system in a network with which a system can communicate.

remote workstation. A workstation that is connected to a system by means of data transmission facilities. Contrast with local workstation.

replace. A function or mode that enables the user to substitute text for a specified part of previously entered text.

request. A directive, by means of a basic transmission unit, from an access method that causes the network control program to perform a data-transfer operation or auxiliary operation.

reset. In SAA Advanced Common User Access architecture, a push button that resets changed fields to their initial values and displays them in the window.

resident. Pertaining to computer programs or data while they remain on a particular storage device.

resolution. In computer graphics, a measure of the sharpness of an image, expressed as the number of lines and columns on the display screen or the number of pels per unit of area.

resource. (1) Any of the data processing system elements needed to perform required operations, including storage, input/output units, one or more processing units, data, files, and programs. Synonymous with computer resource. (2) Any facility of a computing system or operating system required by a job or task, and including main storage,

input/output devices, processing unit, data sets, and control or processing programs.

Resource Access Control Facility. An IBM-licensed program that provides for access control by identifying and by 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.

resource definition. A library member containing the set of records that collectively define a resource.

restart. To resume the execution of a computer program using the data recorded at a checkpoint.

restore. (1) To return to an original value or image; for example, to restore data in main storage from auxiliary storage. See also save. (2) In VSE, to write back onto disk data that was previously written from disk onto an intermediate storage medium such as tape. (3) To return a backup copy to the active storage location for use.

retrieve. To locate data in storage and read it so that it can be processed, printed, or displayed. Contrast with store.

return. (1) Within a subroutine, to effect a link to the computer program that called the subroutine. (2) In programming languages, a language construct within a procedure designating an end of an execution sequence in the procedure.

root. The highest level of a hierarchy.

router. In OSI terminology, a function that determines a path by which an entity can be reached.

routing. (1) The process of determining the path to be used for transmission of a message over a network. (2) The assignment of the path by which a message is to reach its destination. (3) In SNA, the forwarding of a message unit along a particular path through a network, as determined by parameters carried in the message unit, such as the destination network address in a transmission header.

S

semantics. (1) The relationships of characters or groups of characters to their meanings, independent of the manner of their interpretation and use. (2) The relationships between symbols and their meanings.

sender. A person or thing that sends something. Contrast with receiver.

sending. In VTAM, the process by which the host processor places a message on a line for transmission to a station. Contrast with receiving.

server. (1) A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server. (2) In a network, a data station that provides facilities to other stations; for example, a file server, a print server, a mail server. (3) In TCP/IP, a system in a network that handles the requests of a system at another site, called a client-server.

service level. In SNADS, one of the four levels of service (fast, status, data high, or data low) that determines if a distribution is put on the normal or priority distribution queue.

session. (1) In network architecture, for the purpose of data communication between functional units, all the activities which take place during the establishment, maintenance, and release of the connection. (2) A logical connection between two network accessible units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. Each session is uniquely identified in a transmission header (TH) accompanying any transmissions exchanged during the session. (3) The period of time during which a user of a terminal can communicate with an interactive system, usually, elapsed time between logon and logoff.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of connections between the units, and the adjustments needed for the computer to operate on a problem. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels and loading card decks.

shared. Pertaining to the availability of a resource for more than one use at the same time.

shell. (1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens and communicate them to the operating system. Shells simplify user interactions by eliminating the user′s concern with operating system requirements. A computer may have several layers of shells for various levels of user interaction. (2) Software that allows a kernel program to run under different operating system environments.

shift. The concerted movement of some or all of the characters of a word each by the same number of character places in the direction of a specified end of the word.

shutdown. The process of ending operation of a system or a subsystem, following a defined procedure.

simulation. The use of a data processing system to represent selected behavioral characteristics of a physical or abstract system; for example, the representation of air streams around airfoils at various velocities, temperatures, and air pressures.

softcopy. One or more files that can be electronically distributed, manipulated, and printed by a user. Contrast with hardcopy.

software. (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. Software is an intellectual creation that is independent of the medium on which it is recorded. (2) Contrast with hardware.

source. A system, a program within a system, or a device that makes a request to a target. Contrast with target.

source code. The input to a compiler or assembler, written in a source language. Contrast with object code.

spreadsheet. A worksheet arranged in rows and columns, in which a change in the contents of one cell can cause electronic recomputation of one or more cells, based on user defined relations among the cells.

standard label. A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

startup. See system startup.

statement. In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.

static. (1) In programming languages, pertaining to properties that can be established before execution of a program; for example, the length of a fixed length variable is static. (2) Pertaining to an operation that occurs at a predetermined or fixed time. (3) Contrast with dynamic.

storage group. A named collection of physical devices to be managed as a single object storage area. It consists of an object directory (DB2 table space), and object storage on DASD (DB2 table spaces) with optional library-resident optical volumes.

Storage Management Subsystem. A component of MVS/DFP that is used to automate and centralize the management of storage by providing the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

string. (1) A sequence of elements of the same nature, such as characters considered as a whole. (2) In programming languages, the form of data used for storing and manipulating text.

structure. A variable that contains an ordered group of data objects. Unlike an array, the data objects within a structure can have varied data types.

subdirectory. In an IBM personal computer, a file referred to in a root directory that contains the names of other files stored on the diskette.

sublibrary. In VSE, a subdivision of a library.

subnet. In TCP/IP, a part of a network that is identified by a portion of the Internet address.

substring. A part of a character string.

subsystem. (1) A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system. (2) In Open Systems Interconnection architecture, an element in a hierarchical division of an open system that directly interacts only with elements in the next higher division or the next lower division of that open system. A hierarchical division of an open system may be either a layer or a sublayer.

supervisor. The part of a control program that coordinates the use of resources and maintains the flow of processing unit operations.

symbolic name. In a programming language, a unique name used to represent an entity such as a field, file, data structure, or label.

synchronization. In a programming language, a unique name used to represent an entity such as a field, file, data structure, or label.

syntax. The rules governing the structure of a language.

sysplex. A multiple-MVS system environment that allows MCS consoles or extended MCS consoles to receive messages and send commands across systems.

system console. A console, usually equipped with a keyboard and display screen, that is used by an operator to control and communicate with a system.

system image. The representation of a program and its related data as it exists in main storage.

system object. The system profile, system maps, and system JCL masks used by SDF/CICS for communication with users. System objects are kept in the map specification library and can be maintained by the SDF/CICS master operator.

system startup. Synonym for initial program load.

T

target directory. The directory to which information is written. Contrast with source directory.

Telnet. In TCP/IP, an application protocol that allows a user at one site to access a remote system as if the user′s display station were locally attached. Telnet uses the Transmission Control Protocol as the underlying protocol.

temporary file. A file that can be erased or overwritten when it is no longer needed. Contrast with permanent file.

terminate. (1) In SNA products, a request unit that is sent by a logical unit (LU) to its system services control point (SSCP) to cause the SSCP to start a procedure for ending one or more designated LU-LU sessions. (2) To stop the operation of a system or device. (3) To stop execution of a program.

text editor. (1) A computer program that enables a user to create and revise text. (2) A program used to create, modify, and print or display text files.

thread. In the OS/2 operating system, the smallest unit of operation to be performed within a process.

throughput. (1) A measure of the amount of work performed by a computer system over a period of time, for example, number of jobs per day. (2) In data communication, the total traffic between stations per unit of time.

Time Sharing Option. An operating system option; for the System/390 system, the option provides interactive time sharing from remote terminals.

timeout. (1) An event that occurs at the end of a predetermined period of time that began at the occurrence of another specified event. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

timer. A register whose contents are changed at regular intervals in such a manner as to measure time. Synonymous with clock register, time register.

tool. Software that permits the development of an application program without using a traditional programming language.

transaction. (1) In a batch or remote batch entry, a job or job step. (2) An exchange between a workstation and another device that accomplishes a particular action or result; for example, the entry of a customer′s deposit and the updating of the customer′s balance. (3) An item of business; for example, the handling of customer orders and customer billing. (4) A specific set of input data that triggers execution of a specific process or job; a message destined for an application program. (5) In communications, an exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result.

transaction program. (1) A program that processes transactions in an SNA network. There are two kinds of transaction programs: application transaction programs and service transaction programs. See also conversation. (2) In VTAM, a program that performs services related to the processing of a transaction. One or more transaction programs may operate within a VTAM application program that is using the VTAM application program interface (API). In that situation, the transaction program would request services from the application program, using protocols defined by that application program. The application program, in turn, could request services from the VTAM program by issuing the APPCCMD macroinstruction.

transfer. To send data from one place and receive the data at another place.

transmission. A table used to replace one or more characters with alternative characters; for example, to translate characters representing a virtual address to those representing a real address, characters representing an event to those representing a procedure call, characters of a national character set to those of another national language, or characters representing a relocated address to those representing an absolute address.

transmit. (1) To send data from one place for reception elsewhere. (2) To move an entity from one place to another; for example, to broadcast radio waves, to dispatch data via a transmission medium, or to transfer data from one data station to another via a line.

truncate. (1) To terminate a computational process in accordance with some rule; for example, to end the evaluation of a power series at a specified term. (2) To remove the beginning or ending elements of a string. (3) To drop data that cannot be printed or displayed in the line width specified or available. Contrast with fold. (4) To shorten a field or statement to a specified length.

tuning. The process of adjusting an application or a system to operate in a more efficient manner in the work environment of a particular installation.

U

unit of work. In advanced program-to-program communications, the amount of processing that is started directly or indirectly by a program on the source system.

unpack. To recover the original form of the data from packed data.

update. To add, change, or delete items.

user. (1) A person who requires the services of a computing system. (2) Any person or any thing that may issue or receive commands and messages to or from the information processing system.

user data. Data transferred between entities in a layer on behalf of the entities in the next higher layer for which the former entities are providing services.

user exit. A programming service provided by an IBM software product that may be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

user ID. User identification.

user name. In RACF, one to twenty alphanumeric characters that represent a RACF-defined user.

utility. (1) A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. Synonymous with service program. (2) A program designed to perform an everyday task such as copying data from one storage device to another.

V

variable. In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type.

virtual. Pertaining to a functional unit that appears to be real, but whose functions are accomplished by other means.

virtual file system. In the AIX operating system, a remote file system that has been mounted so that it is accessible to the local user.

virtual machine. (1) A virtual data processing system that appears to be at the exclusive disposal of a particular user, but whose functions are accomplished by sharing the resources of a real data processing system. (2) A system in which each user appears to have his own computer and input/output devices.

virtual reality. A computer-generated simulation of reality with which users can interact using specialized peripherals such as data gloves and head-mounted computer graphic displays. Synonymous with artificial reality.

Virtual Telecommunications Access Method. 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.

virtual terminal. (1) A generalized logical model of different terminals of a certain class, describing how terminals of that class will perform in the OSI environment. (2) In the AIX operating system, any of several logical equivalents of a display station. A virtual terminal supports the illusion that more devices exist than are physically present. Virtual terminals are logically independent of each other, but share physical resources over time. (3) In TCP/IP, a system object, created and controlled by an application program that provides a functional representation or simulation of a physical display station.

volume label. An area on a standard label tape used to identify the tape volume and its owner. This area is the first 80 bytes and contains VOL 1 in the first four positions.

W

window. A portion of a display surface in which display images pertaining to a particular application can be presented. Different applications can be displayed simultaneously in different windows.

working directory. Synonym for current directory.

workstation. (1) One or more programmable or nonprogrammable devices that allow a user to do work. See also programmable workstation, nonprogrammable workstation. (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

List of Abbreviations

ABEND	ABnormal END
ACF/VTAM	Advanced Communications Facility/Virtual Telecommunications Access Method
AFP	Advanced Function Printing
AHHC	APPC Host-to-HostConnection
AIX	Advanced Interactive Executive
APAR	Authorized Program Analysis Report
API	Application Programming Interface
APPC	Advanced Program-to-Program Communication
APPL	APPLication
APPN	Advanced Peer-to-Peer Networking
AR	Attention Routine
ASCII	American National Standard Code for Information Interchange
BFS	Byte File System
BG	BackGround
BMS	Basic Mapping Support
BTAM	Basic Telecommunications Access Method
C + +	A Preprocessor to C
CD	Change Directory
CDRM	Cross-Domain Resource Manager
CEMT	Master Terminal Transaction
CET	Central European Time
CICS	Customer Information Control System
CMS	Conversational Monitor System
COBOL	COmmon Business Oriented Language
COS	Class of Service
CP	Control Program
CPC	Central Processing Complex
CPIC	Common Programming Interface for Communications
CPU	Central Processing Unit
CSA	Common Storage Area
CSD	CICS System Definition
CTC	Channel To Channel
CTCA	Channel To Channel Adapter
DAM	Direct Access Method
DASD	Direct Access Storage Device
DAT	Digital Audio Tape
DB2	DataBase 2
DFSMS	Data Facility Storage Management Subsystem
DITTO	Data Interfile Transfer, Testing & Operations utility
DL/I	Data Language 1
DLBL	Disk LaBeL
DLI	dd.Data Language 1
DNS	Domain Name Support
DOS	Disk Operating System
EBCDIC	Extended Binary Coded Decimal Interchange Code
EOJ	End Of Job
ESA	Enterprise Systems Architecture
ESCON	Enterprise Systems CONnection
ESDS	Entry Sequenced Data Set
ESM	External Security Manager
EXEC	EXECute/EXECution
FB	Fixed Block
FCT	File Control Table
FIFO	First In/first Out
FTP	File Transfer Program
GB	GigaByte
GID	Group IDentifier
GIF	Graphical Image Format
GUI	Graphical User Interface
HFS	Hierarchical File System
HMC	Hardware Management Console
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output

IBM	International Business Machines
ICCF	Interactive Computing and Control Facility
ICSF	Integrated Cryptographic Service Facility
ICSS	Internet Connection Secure Server
ID	IDentification/IDentifier
IDCAMS	The program name for Access Method Services
II	Interactive Interface
IML	Initial Microprogram Load
INET	Integrated NETwork
IOCDS	I/O Configuration Data Set
IODF	Input/Output Definition File
IP	Internet Protocol
IPL	Initial Program Load
ITSO	International Technical Support Organization
IUI	Interactive User Interface
JCL	Job Control Language
JES	Job Entry Subsystem
JPG	Joint Photographic Experts Group
KSDS	Key Sequenced Data Set
LAN	Local Area Network
LDL	Library Directory List
LPAR	Logically Partitioned mode
LSR	Local Shared Resources
LU	Logical Unit
MB	MegaByte
MIH	Missing Interruption Handler
MQ	Message Queue
MQI	Message Queue Interface
MQSeries	Message Queuing Series
MR	Modified Read
MVS	Multiple Virtual Storage
N/A	Not Applicable
NCP	Network Control Program
NETID	NETwork IDentification
NFS	Network File System
NFSD	Network File Server Daemon
OS/2	Operating System/2
OS/390	IBM System/390 Operating System
OSA	Open System Architecture
PC	Personal Computer
PCOM	Personal COMmunication
PL/1	Programming Language 1
PL/I	Programming Language 1
POSIX	Portable Operating System Interface for Computer Environments
POWER	Priority Output Writers, Execution processor, and input Readers
PU	Physical Unit
RACF	Resource Access Control Facility
RBA	Relative Byte Address
RC	Return Code
RDO	Resource Definition On-line
RISC	Reduced Instruction-Set Computer
RPC	Remote Procedure Call
RR	Repeatable Read
S/390	IBM System/390
SA-SE	StandAlone Support Element
SAM	Sequential Access Method
SFS	Shared File System
SIT	System Initialization Table
SLI	Single Line Interface
SMF	System Management Facility
SNA	Systems Network Architecture
SQL	Structured Query Language
SSCP	System Services Control Point
SSL	Secure Sockets Layer
SYSRES	SYStem RESidence file
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TELNET	U.S. Dept. of Defense's Virtual Terminal Protocol
TGN	Transmission Group Number
TOD	Time Of Day
TP	TeleProcessing
TRL	Transport Resource List
TSO	Time Sharing Option
UDP	User Datagram Protocol

UNIX	An Operating System Developed at Bell Laboratories
UPSI	Use Program Switch Indicator
URL	Universal Resource Locator
VM	Virtual Machine
VM/CMS	Virtual Machine/Conversational Monitor System
VM/ESA	Virtual Machine/Enterprise Systems Architecture
VRML	Virtual Reality Markup Language
VSAM	Virtual Storage Access Method
VSE	Virtual Storage Extended
VSE/ESA	Virtual Storage Extended/Enterprise Systems Architecture
VSE/POWER	Virtual Storage Extended/Priority Output Writers, Execution processor, and input Readers
VSE/VSAM	Virtual Storage Extended/Virtual Storage Access Method
VTAM	Virtual Telecommunications Access Method
VTOC	Volume Table of Contents
WLM	Workload Manager
XDR	eXternal Data Representation
XRF	eXtended Recovery Facility

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

ITSO Redbook Evaluation

Connecting Auto UNIX System to VM and VSE
SG24-5377-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and Fax it to: USA International Access Code + 1 914 432 8264 or:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- 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!)

_____

_____

_____

_____

_____

SG24-5377-00
Printed in the U.S.A.